

# Table of Contents

[Table of Contents 2](#_Toc34572706)

[1 Introduction 5](#_Toc34572707)

[2 Planning and collecting Resources 6](#_Toc34572708)

[2.1 Prototype roadmap 6](#_Toc34572709)

[2.2 Research on technology 6](#_Toc34572710)

[2.3 Why OpenCV is better for face – recognition 8](#_Toc34572711)

[2.4 Haar Cascade algorithm to recognize the face 8](#_Toc34572712)

[3 Methodology/Design 10](#_Toc34572713)

[3.1 Circuit diagram 10](#_Toc34572714)

[3.2 Software tools 10](#_Toc34572715)

[3.3 Hardware tools 12](#_Toc34572716)

[3.3.1 Raspberry Pi 12](#_Toc34572717)

[3.3.2 USB Camera 12](#_Toc34572718)

[3.3.3 Connecting Cable 13](#_Toc34572719)

[3.4 Network protocol 13](#_Toc34572720)

[3.5 Important Libraries 13](#_Toc34572721)

[4 Implementation of prototype 14](#_Toc34572722)

[4.1 Raspberry pi issue 14](#_Toc34572723)

[4.2 PyCharm installation issue with raspberry pi 15](#_Toc34572724)

[4.3 Switch to windows platform 15](#_Toc34572725)

[4.4 Phase 1 Data Gathering 18](#_Toc34572726)

[4.4.1 Code to detect face 19](#_Toc34572727)

[4.5 Phase 2 Train the recognizer 20](#_Toc34572728)

[4.5.1 Code to train the dataset 22](#_Toc34572729)

[4.6 Phase 3 Recognition 23](#_Toc34572730)

[4.6.1 Code to recognize the image 24](#_Toc34572731)

[4.7 Upload data on cloud service ThingsPeak 26](#_Toc34572732)

[4.7.1 Code to upload data on ThingsPeak 26](#_Toc34572733)

[5 Testing 27](#_Toc34572734)

[5.1 Troubleshoot 28](#_Toc34572735)

[6 Deploying prototype 30](#_Toc34572736)

[6.1 Run program 30](#_Toc34572737)

[6.1.1 Screen shot of the application 31](#_Toc34572738)

[6.2 Security 36](#_Toc34572739)

[7 Conclusion 36](#_Toc34572740)

[8 Future work 37](#_Toc34572741)

[9 Reference List 37](#_Toc34572742)

**List of Figures**

[Figure 1 Roadmap of prototype 6](#_Toc34572743)

[Figure 2 Specification of Arduino and Raspberry pi. 7](#_Toc34572744)

[Figure 3 suitable for a different project 7](#_Toc34572745)

[Figure 4 Choice of system 8](#_Toc34572746)

[Figure 5 Block Diagram 10](#_Toc34572747)

[Figure 6 Raspberry pi 3 b+ model [] 12](#_Toc34572748)

[Figure 7 Raspberry pi OpenCV installation error -1 14](#_Toc34572749)

[Figure 8 Library installation issue 14](#_Toc34572750)

[Figure 9 USB camera issue with raspberry pi 15](#_Toc34572751)

[Figure 10 PyCharm IDE installation on Raspberry pi 3 15](#_Toc34572752)

[Figure 11 Project structure 16](#_Toc34572753)

[Figure 12 Phase 1: Data Gathering 18](file:///D:\Level%207%20GDIT\Block%203\IOT\Assignment%20Research\Portfolio_Assignment.docx#_Toc34572754)

[Figure 13 face-detection code 19](#_Toc34572755)

[Figure 14 Phase 2 Train the recognizer 20](file:///D:\Level%207%20GDIT\Block%203\IOT\Assignment%20Research\Portfolio_Assignment.docx#_Toc34572756)

[Figure 15 face-trainner.yml file 21](#_Toc34572757)

[Figure 16 Training dataset code 22](#_Toc34572758)

[Figure 17 Recognition 23](file:///D:\Level%207%20GDIT\Block%203\IOT\Assignment%20Research\Portfolio_Assignment.docx#_Toc34572759)

[Figure 18 Code to recognize student's face 24](file:///D:\Level%207%20GDIT\Block%203\IOT\Assignment%20Research\Portfolio_Assignment.docx#_Toc34572760)

[Figure 19 output array with data 25](#_Toc34572761)

[Figure 20 Output when the record is not in the dataset 25](#_Toc34572762)

[Figure 21 Upload data on ThingsPeak 26](#_Toc34572763)

[Figure 22 Output of uploaded data on ThingsPeak 27](#_Toc34572764)

[Figure 23 Record is not stored in the Data set 29](#_Toc34572765)

[Figure 24 Test JSON data result in Postman 30](#_Toc34572766)

[Figure 25 Testing 1 31](#_Toc34572767)

[Figure 26 Testing 2 32](#_Toc34572768)

[Figure 27 Testing 3 32](#_Toc34572769)

[Figure 28 Testing 4 33](#_Toc34572770)

[Figure 29 Testing 5 33](#_Toc34572771)

[Figure 30 Testing 6 34](#_Toc34572772)

[Figure 31 Testing 6.1 34](#_Toc34572773)

[Figure 32 Testing 7 35](#_Toc34572774)

[Figure 33 Visualize ThingsPeak uploaded data in a CSV file 35](#_Toc34572775)

**List of Tables**

[Table 1 Libraries used in Prototype 13](#_Toc34571952)

# Introduction

This use case is implemented to introduce an IoT (Internet of Things) based face recognition attendance system to resolve written presence of documents, address the problem of the fake involvement of the student, and update student attendance every day.

IoT based face recognition attendance systems were then introduced for the attendance system. The problem here was that to fix the student's fake participation in attendance . As a result, Face recognition based automated identification system based are gaining popularity due to the unique nature of detecting student’s face by web-cam in a class room.

Through this project we are developing an intelligent attendance system which uses a face detection module to identify the face of the student in order to store their details. The attendance is planned and scheduled once the engagement is attended and recorded on the ThingsPeak cloud server. This database is requested for the student's information. The collected data will be displayed in CSV format from the ThingsPeak data.

# Planning and collecting Resources

## Prototype roadmap

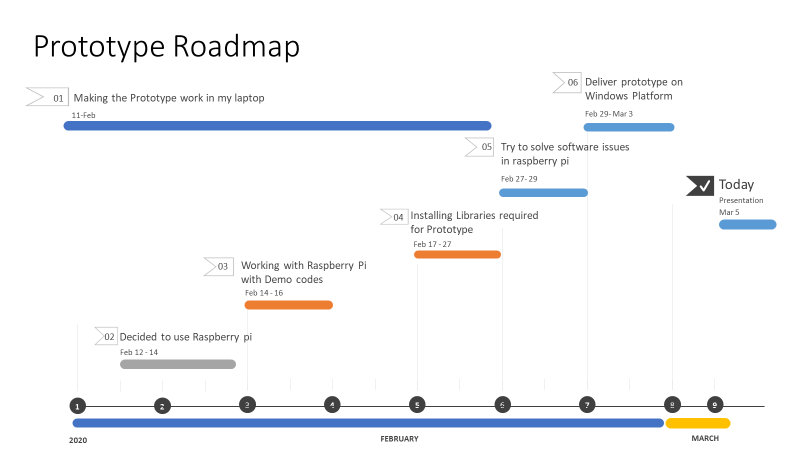


Figure 1 Roadmap of prototype

Above figure describe the working scenario while implementing a prototype. It illustrates the timeline chart of the project. It defines from the beginning try sample demo example on the laptop to work with the raspberry pi and at the end present the prototype demo during the presentation.

## Research on technology

IoT (Internet of Things) application is based on micro-controller and microprocessor. Arduino and raspberry pi both are IoT platforms, which are used to make a different prototype. Arduino is based on micro-controller and Raspberry pi is work on micro-processor.

The microcontroller has strong i/o (input/output) capability, therefore it can drive external hardware directly. However, Microprocessor has weak i/o capability which needs transistor to operate most device. It is suitable for processing has brainier work as a minicomputer. It consumes more power than a microcontroller.



Figure 2 Specification of Arduino and Raspberry pi.

Above figure describe the specification of both systems which define their capacity, size and reliability.



Figure 3 suitable for a different project

Figure 3 is about which system is best. That depends on the project prototype. In this figure it says that Arduino is best for the project which based on sensor, display or motor. However, microprocessor-based Raspberry pi is best for the video streaming camera, complex maths or advance graphics-based prototype.

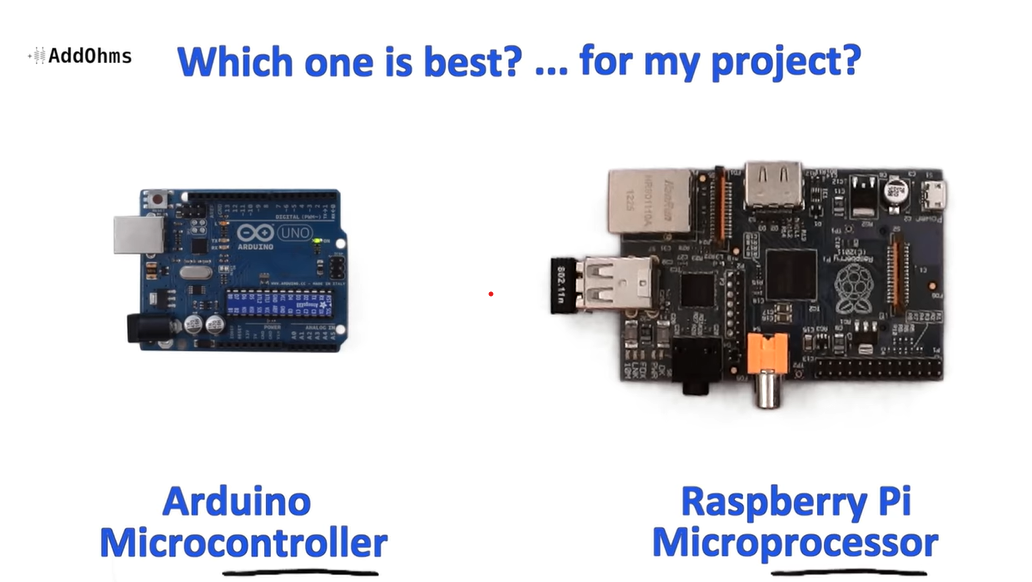


Figure 4 Choice of system

Figure 4 describes that neither is going to be perfect for every single application in electronics word. If the application is more about controlling things however Arduino is a better choice or if process lots of data, then probably raspberry pi is a better choice.

Based on this research, I decide to use the raspberry pi to made intelligent attendance system which recognizes the image of the student using a USB camera.

## Why OpenCV is better for face – recognition

For Face recognition there are several API such as OpenCV, TensorFlow object detection API or PyTorch.

For the smart attendance prototype, I choose OpenCV for face recognition because it is better than other API. According to the research by Roy in 2018, “OpenCV DNN runs faster inference than the [TensorFlow object detection API](https://github.com/tensorflow/models/tree/master/research/object_detection)with higher speed and low computational power” (Roy,2018). As per the article, using TensorFlow or PyTorch is difficult with deep learning libraries to make a portable solution using computer vision model on edge devices like Raspberry pi (Roy,2018).

## Haar Cascade algorithm to recognize the face

I used to implementation of my use case using Haar cascade classifier. Haar Cascade Classifier is a powerful detection technique proposed in their 2001 paper "Rapid Detection of Objects through Boosted Cascades of Simple Features" by Paul Viola and Michael Jones (Khan,2019).

Haar Cascade is a machine-based learning approach in which a cascade method is trained in many positive and negative images. It is used to identify objects in other pictures based on the training.

This algorithm includes a Haar feature selection process. Each human face has similar characteristics. Such regularities can be associated with Haar features.

A few properties common to human faces:

a) The surface of the eye is darker than the upper cheeks.

b) The area of the nose bridge is darker than the eyes.

Composition of properties that suit facial characteristics:

a) Location and size: eyes, mouth, nose bridge.

b) Value: pixel intensity dependent gradients (Soewito,2015).

# Methodology/Design

## Circuit diagram

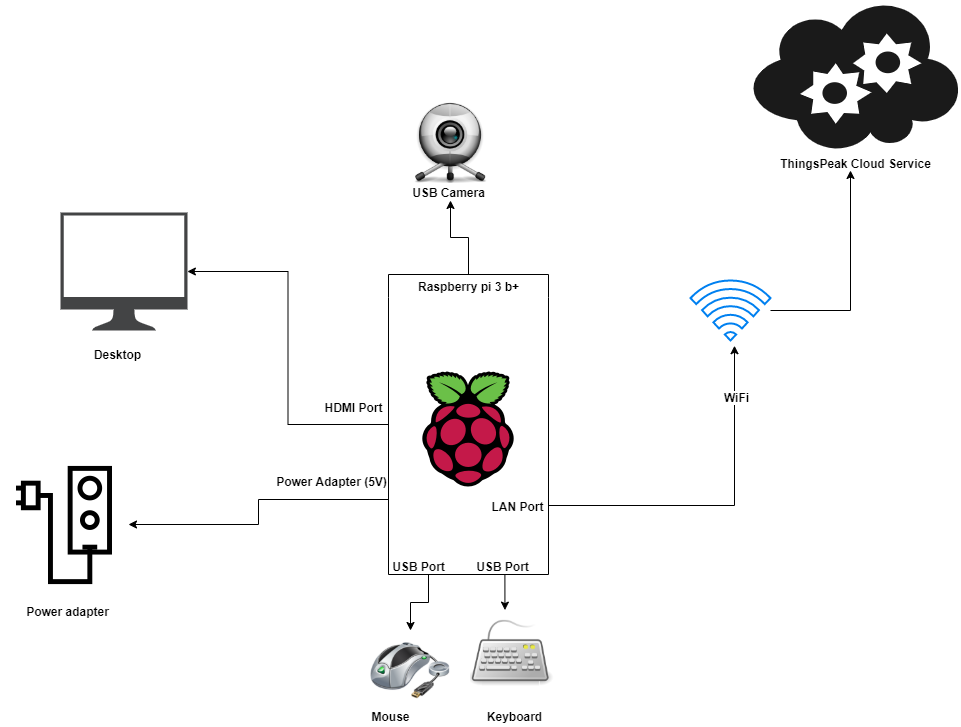


Figure 5 Block Diagram

Above figure represents the block diagram of intelligent attendance system. Which shows that connection of different input output devices, cables with raspberry pi.

## Software tools

* Python IDLE : IDLE is an integrated development environment for editing and running python2.x or python 3 programs. Where we can see or check the output. I used python 3.8.0 version for my prototype.
* Raspbian OS : Raspbian is an Open source operating system which is used run the applications. To run our applications, install the Raspbian OS. Raspbian OS is best for raspberry pi 3 b+ due to many advantages such as, compatibility, reliability and adaptability to most of the beginners' project .[[[1]](#footnote-1)]
* Python : Python is a programming language. Which has easy syntaxes to read that allows fewer lines of code to the programmers. This language is also suitable for other customized applications.
* PyCharm : PyCharm is an IDE (Integrated development environment) which supports different web development tools such as JavaScript, HTML/CSS and also compatible with various libraries platform path such as it supports anaconda libraries [[[2]](#footnote-2)].
* OpenCV : I used OpenCV (Open Computer Vison library) because it is faster than other libraries such as TensorFlow object detection. “OpenCV DNN runs faster inference than the [TensorFlow object detection API](https://github.com/tensorflow/models/tree/master/research/object_detection)with higher speed and low computational power” (Roy,2018).
* Windows OS : I mainly used my windows laptop to make my prototype. And then upload it on raspberry pi.
* ThingsPeak Cloud Service : I used ThingsPeak service to upload my real time data on the cloud. There has been may IoT clod platforms such as Microsoft Azure, AWS , and others. However, I chose ThingsPeak because it’s a free cloud service and has user-friendly interface than other platforms.

## Hardware tools

### Raspberry Pi

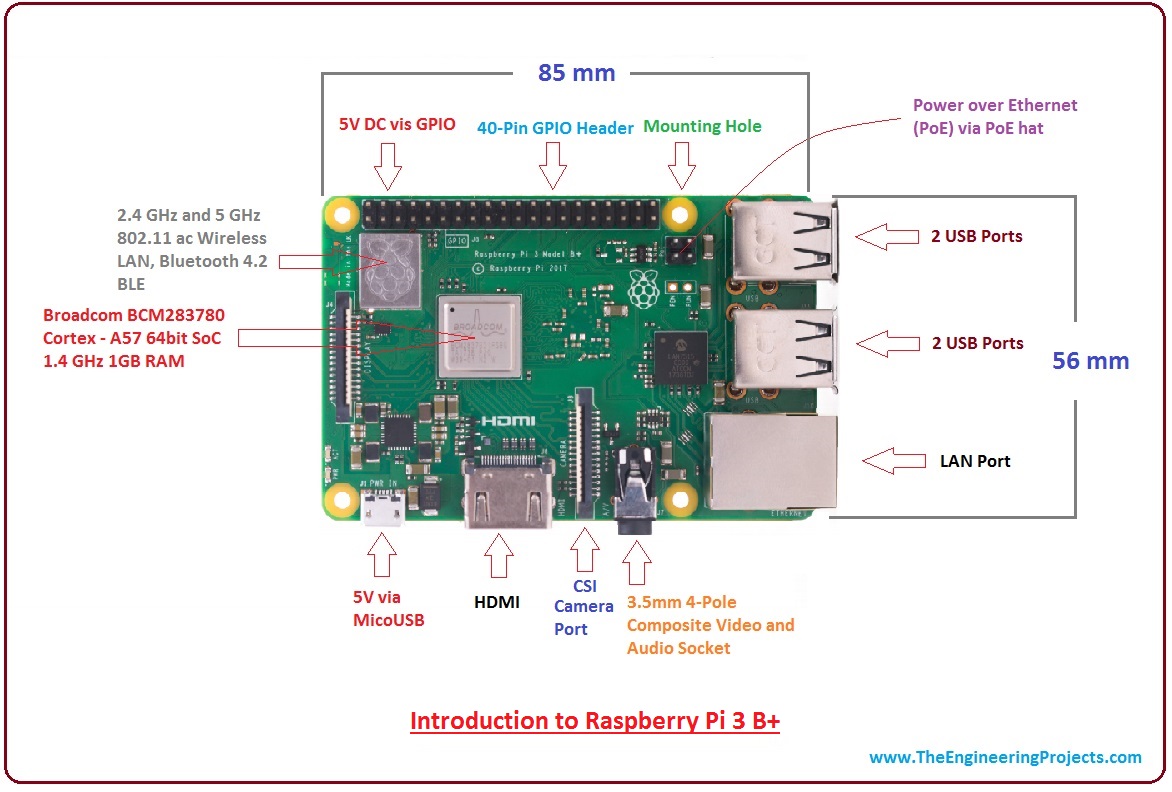


Figure 6 Raspberry pi 3 b+ model [[[3]](#footnote-3)]

The Raspberry Pi 3 B+ Platform was released on 14 March 2018. The Raspberry 3 B, published in 2016, is an advanced version [3].

This small computer board comes with the following features: CPUs, GPU, USB ports, I/ O connectors, Wi-Fi, Bluetooth, USB and network booting.

### USB Camera

For my prototype I used a USB camera to detect a face. A camera is a digital tool for capturing photographs that can be locally stored, transferred to a different location or both. The camera is a remote sensing tool, because without any touch it detects subjects.

### Connecting Cable

To create a prototype environment, I used different cables with a different purpose, such as power cable for raspberry pi, HDMI cable to connect raspberry pi with desktop, ethernet cable to connect raspberry pi with a laptop.

## Network protocol

There are a wide variety of network protocols such as Wi-Fi, MQTT, Bluetooth to transmit data from sensors to cloud service. Diverse range of networks for the communication of IoT devices to choose from, each with a set of features, advantages and disadvantages and ideal scenarios.

For my use case I used Wi-Fi network protocol to transfer data on cloud service. Because Wi-Fi is an easy IoT option, considering that Wi-Fi is almost ubiquitous in building, college or any organization (Parekh, 2017). It is not always the right choice, but it depends on the prototype.

## Important Libraries

Table 1 Libraries used in Prototype

|  |  |
| --- | --- |
| Libraries | Use |
| import tkinter as tk | using Tk and is Python's standard GUI framework. It is used for its simplicity and graphical user interface. It is open source. |
| import datetime  import time | Used to generate system data and time. |
| import numpy as np  from PIL import Image, ImageTk  import pandas as pd | PIL is used to perform several operations on image for detection and recognition. |
| import urllib3  import json | To upload data on the cloud. |
| import csv | Create and perform the operation on csv file |
| import cv2,os | Cv2 library is used for to use face detection function. |

# Implementation of prototype

## Raspberry pi issue

There are two ways to install OpenCV in raspberry pi.

1. Compile OpenCV from source

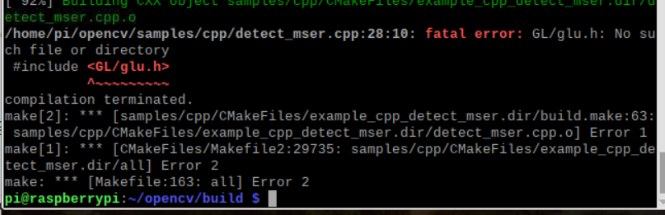


Figure 7 Raspberry pi OpenCV installation error -1

This cmake installation takes almost 2-3 hours. However, due to this error, I could not make it possible to install OpenCV successfully by using this method.

1. pip install OpenCV

Using this method, I almost reach to the installation, but it does not activate the virtual environment script. However, with the help of Andrej we activate script by our self when I start.

Then, I tried to install important libraries required for the prototype but I got this error.

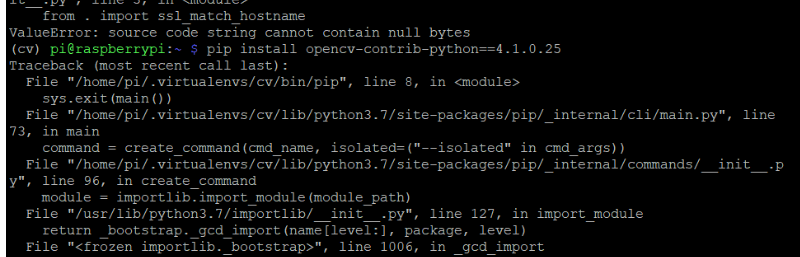


Figure 8 Library installation issue

1. USB Camera

I tried to run my prototype on the raspberry pi to check my USB camera is connect or not correctly. I run the face detection demo example to check that, however it gives me an error “cannot connect to x server”.

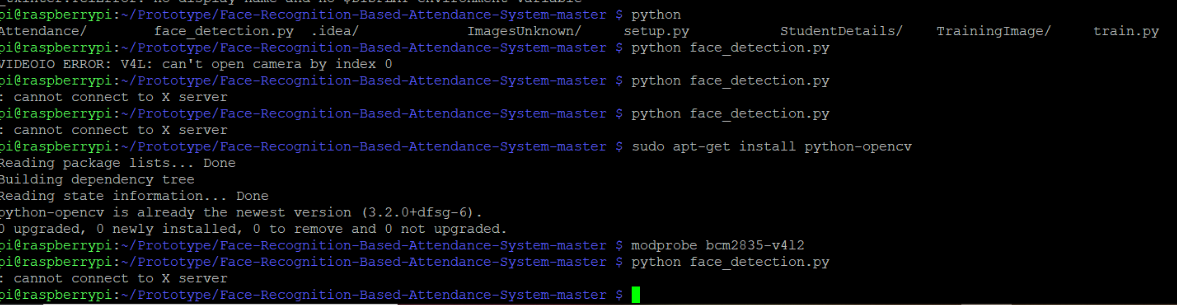


Figure 9 USB camera issue with raspberry pi

## PyCharm installation issue with raspberry pi

After made demo prototype on PyCharm IDE on windows platform, I tried to install PyCharm on raspberry pi. Because, PyCharm allows to create its own virtual environment than using default one. But due to memory issue and raspberry pi 3 processor take a long time to load IDE.

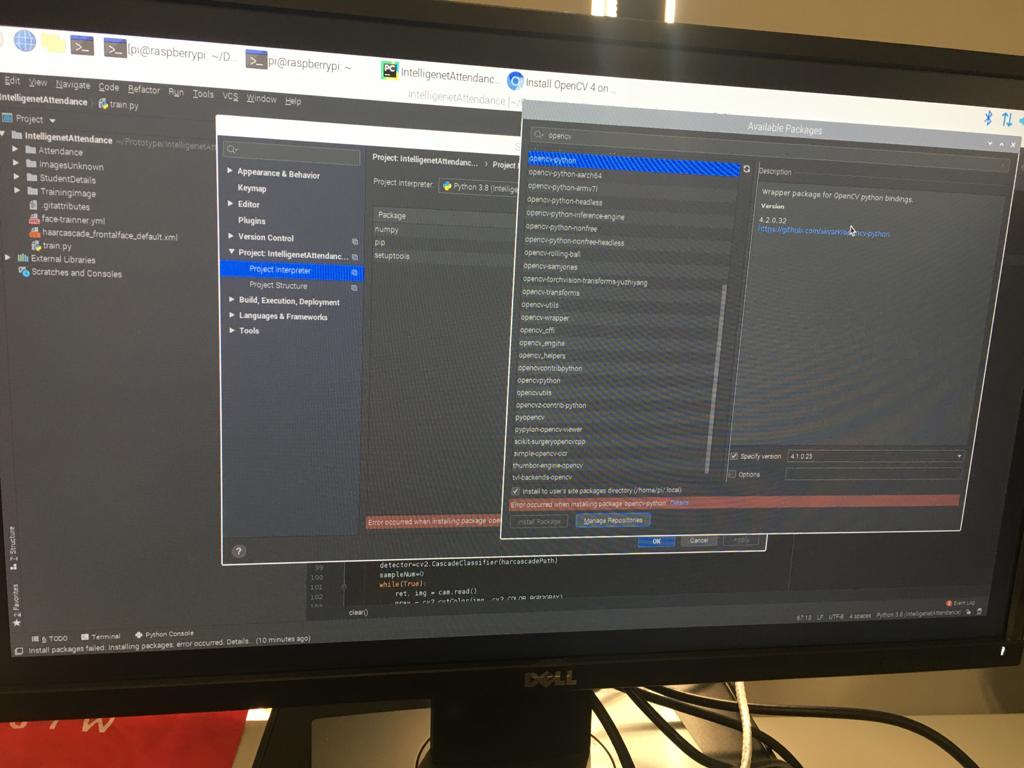


Figure 10 PyCharm IDE installation on Raspberry pi 3

## Switch to windows platform

Due to the above mention issues, I choose to make a prototype in windows platform.

To implement the prototype, I divide my prototype into 4 different part.

* Phase 1: Data Gathering and Face detection
* Phase 2: Train the recognizer
* Phase 3: Face recognition
* Phase 4: Upload real time data on ThingsPeak

Project structure,

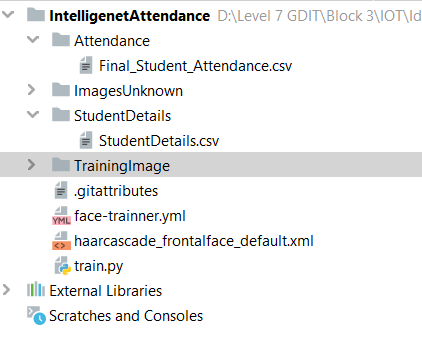


Figure 11 Project structure

As is the above figure shows the structure of the prototype.

Attendance : is a directory which stores the csv file name “Final\_Student\_Attendance.csv” which stores final attendance data. That can be upload on ThingsPeak at the end.

ImagesUnknown: is a directory which stores unknown image which is detected by the recognizer.

StudentDetails: is a directory file which stores students details with their given ID, and name in csv format.

TrainningImage: is a folder where trained images are store.

Face-trainner.yml – file which is used by recognizer to train the data.

Haarcascde\_frontal\_face\_default.xml – file is a Haar Cascade algorithm which is used to detect a face.

Train.py – file is a python file which includes all logic and function to implement a prototype.

## Phase 1 Data Gathering

A screenshot of a cell phone

Description automatically generated

Figure 12 Phase 1: Data Gathering

Above image describes the face detection via USB camera and store data in data set for training recognizer.

### Code to detect face

A screenshot of a cell phone

Description automatically generated

Figure 13 face-detection code

Figure 13 describes the implementation of face detection. For face detection we must have saved on it the Facial Classifier. You can download it from GitHub: [haarcascade\_frontalface\_default.xml](https://github.com/Mjrovai/OpenCV-Face-Recognition/blob/master/FacialRecognition/haarcascade_frontalface_default.xml)

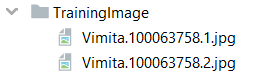
To get user Id and name, I used “input command” to capture student id, that should be an integer number (e.g., 1,2,3.) and student name should be the alphabetical name.

And for each one of the captured frames, it should save as a file in dataset directory by following line of code.

*#saving the captured face in the dataset folder TrainingImage*cv2.imwrite(**"TrainingImage\ "**+name +**"."**+Id +**'.'**+ str(sampleNum) + **".jpg"**, gray[y:y+h,x:x+w])

For saving the above file, it must have the imported library “os”. Each file’s name will follow the structure:

**User\_name.face\_id.count.jpg**



## Phase 2 Train the recognizer

A screenshot of a cell phone

Description automatically generated

Figure 14 Phase 2 Train the recognizer

In phase 2, we must take all user data from dataset and trainer the OpenCV Recognizer. This is happening by a specific OpenCV function. The restult will be a .yml file. As seen in the above figure face-trainner.yml file.

The YML file is primarily related to YAML JavaScript. YAML stands for "YAML Ain't Markup Language;" It uses a text file and organises it in a human-readable format.

YAML can be used on many programming language platforms, such as PHP, Python, Ruby, Perl, and JavaScript [[[4]](#footnote-4)]. This is also used for configuration and for applications that store or transmit data [[[5]](#footnote-5)].

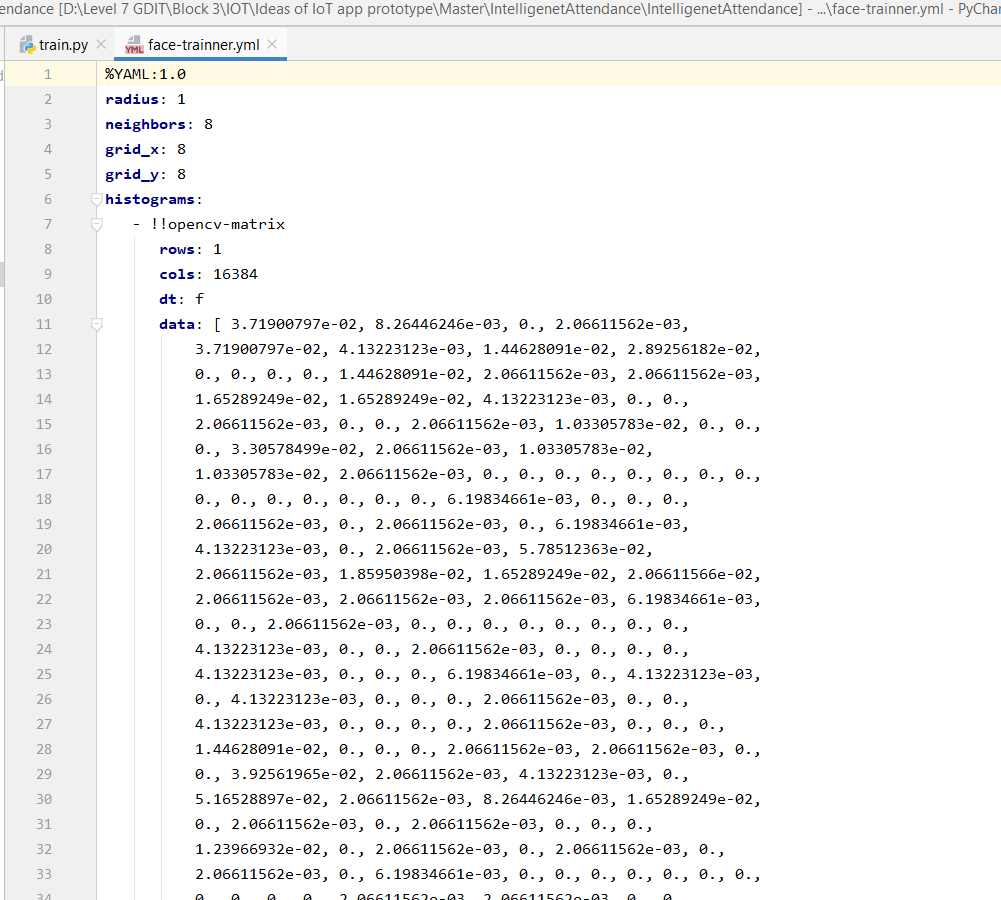


Figure 15 face-trainner.yml file

As seen in the above figure, the face-tranner.yml file defines the number of cols, row , and data where user can understand what the file includes.

### Code to train the dataset

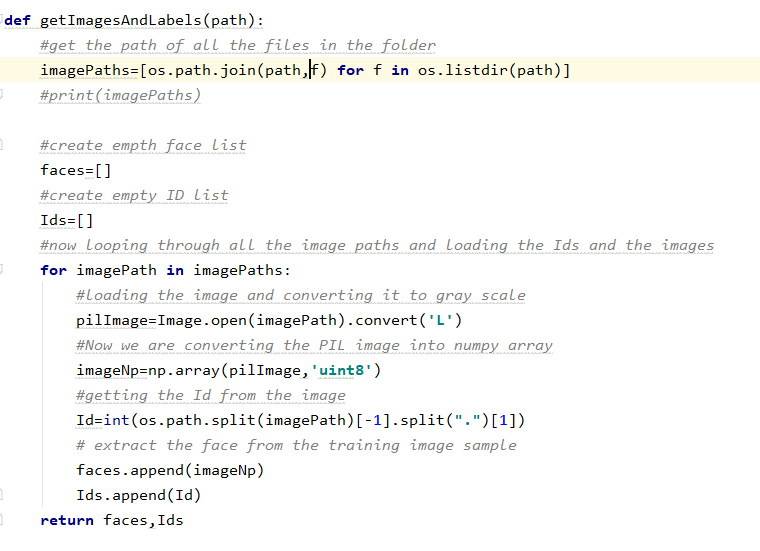


Figure 16 Training dataset code

To perform the above code, we need to install a pillow library if we do not have.

**pip install pillow,** using this command install library.

To train dataset I used, I used LBPH (Local Binary pattern Histograms) face recognizer, included on OpenCV package by following line of code.

recognizer = cv2.face\_LBPHFaceRecognizer.create()

To train the recognizer, I used

faces,Id = getImagesAndLabels(**"TrainingImage"**)

The function getImagesAndLabels(path), where the path is “**TrainingImage**” will take photos from the directory and returning 2 arrays, faces and Id by the following line.

recognizer.train(faces, np.array(Id))

As a result, the face-trainner.yml file will be saved data in the directory.

## Phase 3 Recognition

A screenshot of a cell phone

Description automatically generated

Figure 17 Recognition

Now, we reached the final phase of a prototype implementation. Here, we will capture a fresh face on our camera and if this person had his face captured and trained before, our recognizer would make a "prediction" returning its id and a name, shown how confident the recognizer is with this match as seen in above figure 17.

### Code to recognize the image

A screenshot of a cell phone

Description automatically generatedAbove code describe the track the image from the dataset. It will recognize the face and give the desired id and name once the face is detected.

Figure 18 Code to recognize student's face

As a result, it gives an array of data which include id, name, date, and timestamp to fill the attendance. Attendance array is store in csv file.

Final output array looks like this,



Figure 19 output array with data

And at last, if the recognizer could not predict a face, I put “unknown” label on face. This will identify that in the dataset have no record therefore recognizer could not find that data in the dataset.

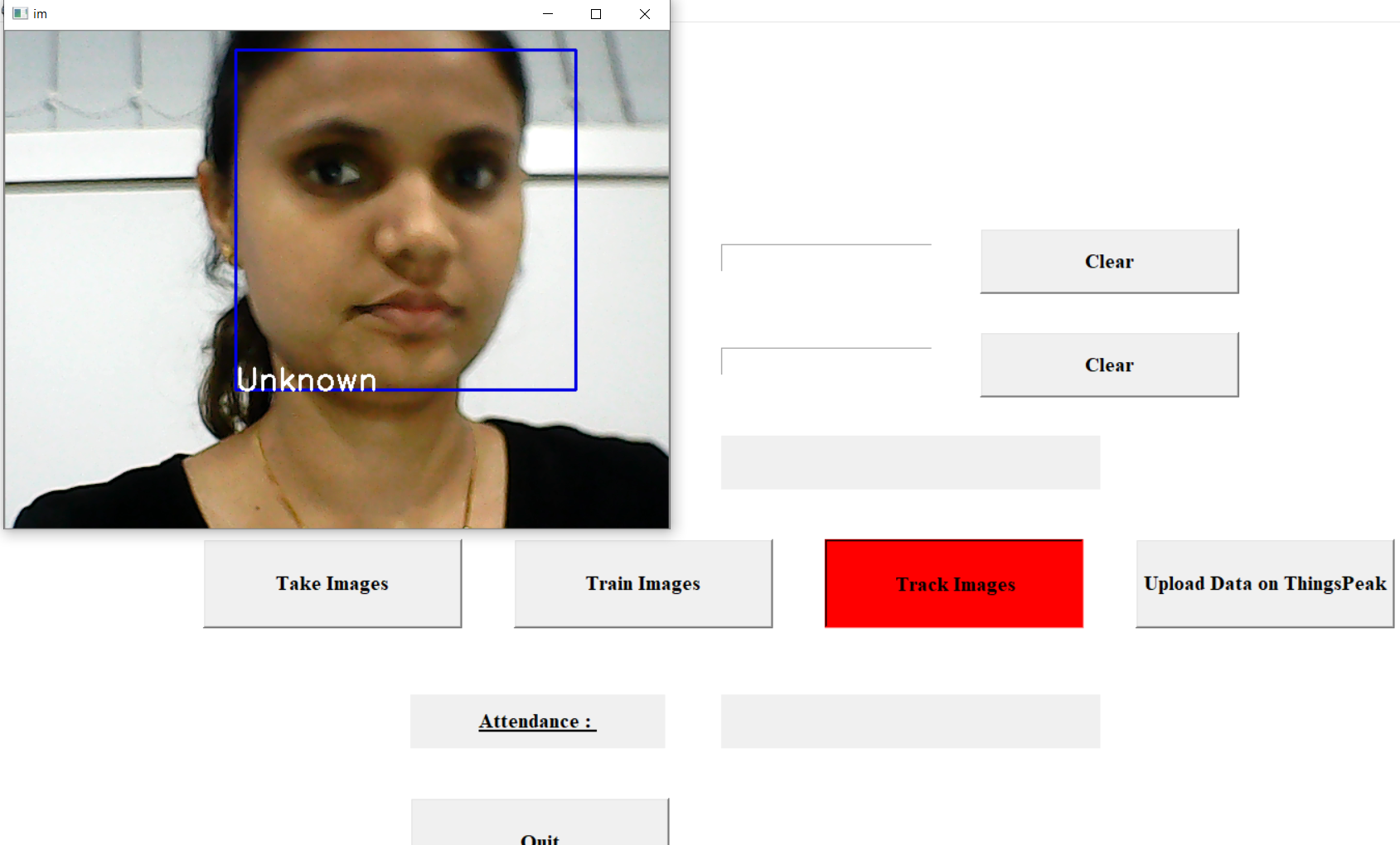


Figure 20 Output when the record is not in the dataset

## Upload data on cloud service ThingsPeak

### Code to upload data on ThingsPeak



Figure 21 Upload data on ThingsPeak

In the above code, at first, we must need api key. I got API key once I created my account on ThingsPeak and created channel.

To upload all wrapped csv file data on the cloud, we must need to pip install

* urllib3 - library to send the request
* pandas.Dataframe - to read csv and generate json

I used the following code format to upload my data,

*#pandas code*

*import pandas as pd*

*colnames=['delta\_t', 'field1','field2','field3','field4']*

*dfForJson = pd.read\_csv('my\_data.csv', names=colnames, header=None, skiprows=1)*

*encodedBody = dfForJson.to\_json(orient ='records')*

where, colnames specify the field name that I created while creating a channel.

To upload my all data which are wrapped in a csv file on ThingsPeak as a JSON format I used following URL pattern,

[https://api.thingspeak.com/channels/*<channel\_id>*/bulk\_update.json](https://api.thingspeak.com/channels/%3cchannel_id%3e/bulk_update.json)

where, <channel\_id> - is my channel id

HTTP method – POST.

write\_api\_key – is object parameter which required private channel and specified the Write API key for the specific channel. This parameter returns the string type.[[[6]](#footnote-6)]

As an output, it uploads data on things peak. And we can visualize data in a graph format.

A screenshot of a computer

Description automatically generated

Figure 22 Output of uploaded data on ThingsPeak

# Testing

This section includes the troubleshooting of application and tests the system that it is working based on requirements or not. It will check the different phase while running the application.

## Troubleshoot

|  |  |  |
| --- | --- | --- |
| Image/Screen short | Issues | Solution |
| N/A | At first, I use the Image file to install Raspbian os on raspberry pi. Which is not supported on Raspberry pi. | Then, I install rasbian operating system on raspeberry pi. And after successfully install the os, continue installation of opencv by following the instruction or steps from a website tutorial.  <https://www.pyimagesearch.com/2018/09/26/install-opencv-4-on-your-raspberry-pi/> |
| Cv2 package not found | Import cv2 | **pip install opencv-python**  <https://www.youtube.com/watch?v=d3AT9EGp4iw> |
|  | AttributeError: module 'cv2.cv2' has no attribute 'createLBPHFaceRecognizer' | Instead of cv2.createLBPHFaceRecognizer()  Use : recognizer = \*\*cv2.face.LBPHFaceRecognizer\_create()\*\*  Install: pip install opencv-contrib-python  <https://stackoverflow.com/questions/44633378/attributeerror-module-cv2-cv2-has-no-attribute-createlbphfacerecognizer> |

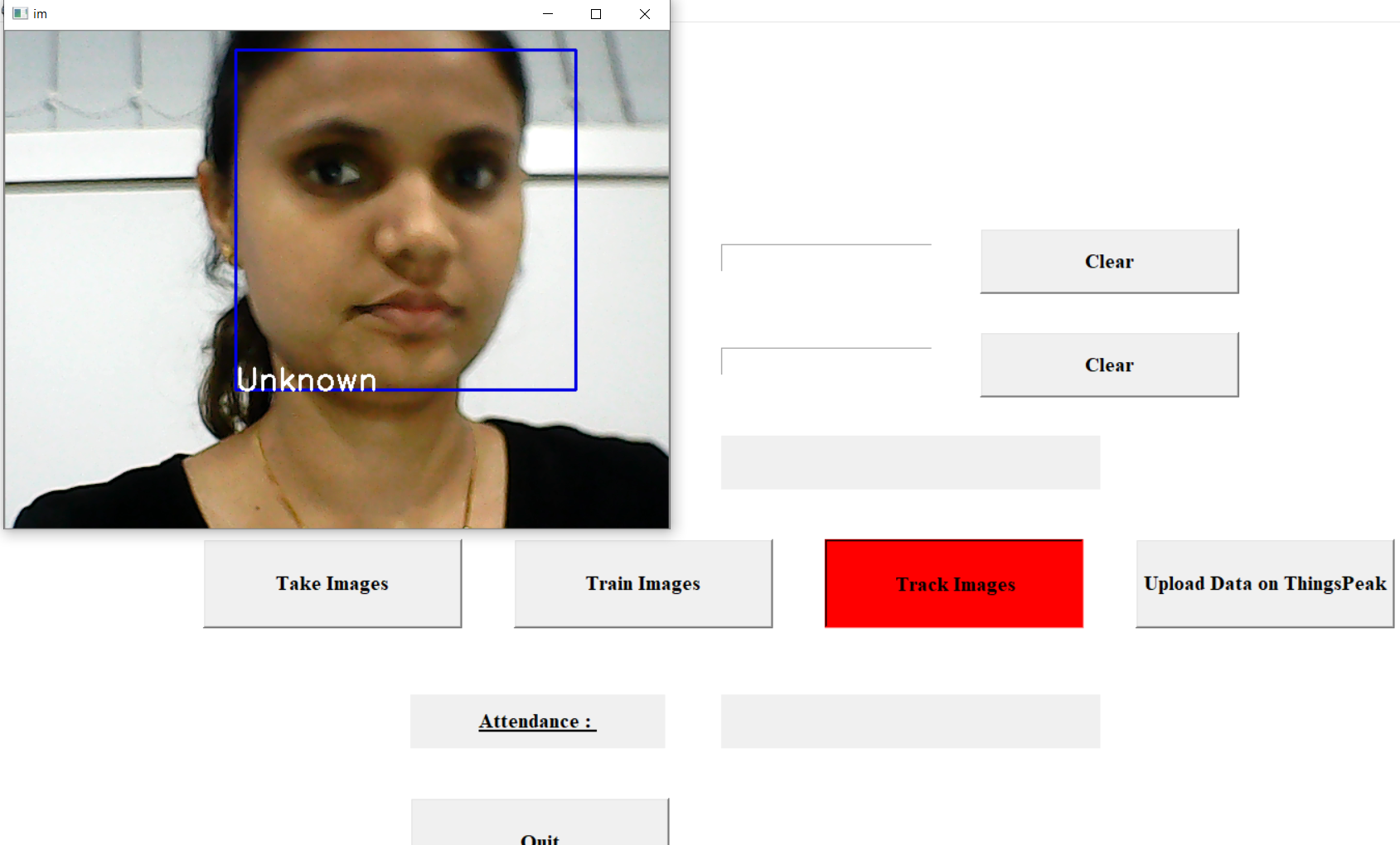


Figure 23 Record is not stored in the Data set

If student’s details are not saved in the data set, then during face recognition it will detect face and recognized as an unknown student.

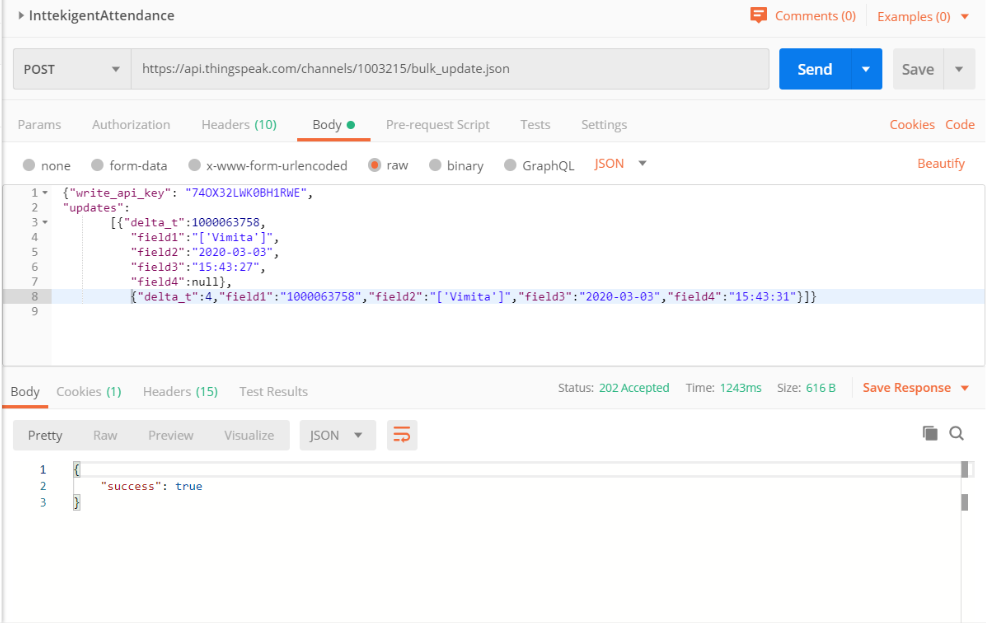


Figure 24 Test JSON data result in Postman

After upload data on ThingsPeak, using JSON URL we can test and check the status on Postman. Status can be true if data is successfully uploaded or else false if not upload.

# Deploying prototype

## Run program

Testing Demo and video is on this following link,

<https://otagopoly-my.sharepoint.com/:f:/g/personal/vaidv1_student_op_ac_nz/ErI-0qRDpHRGv2vlzs5QGzwB7BIsSCn34mjCj705PeJ-Ow?e=dWmmLt>

### Screen shot of the application

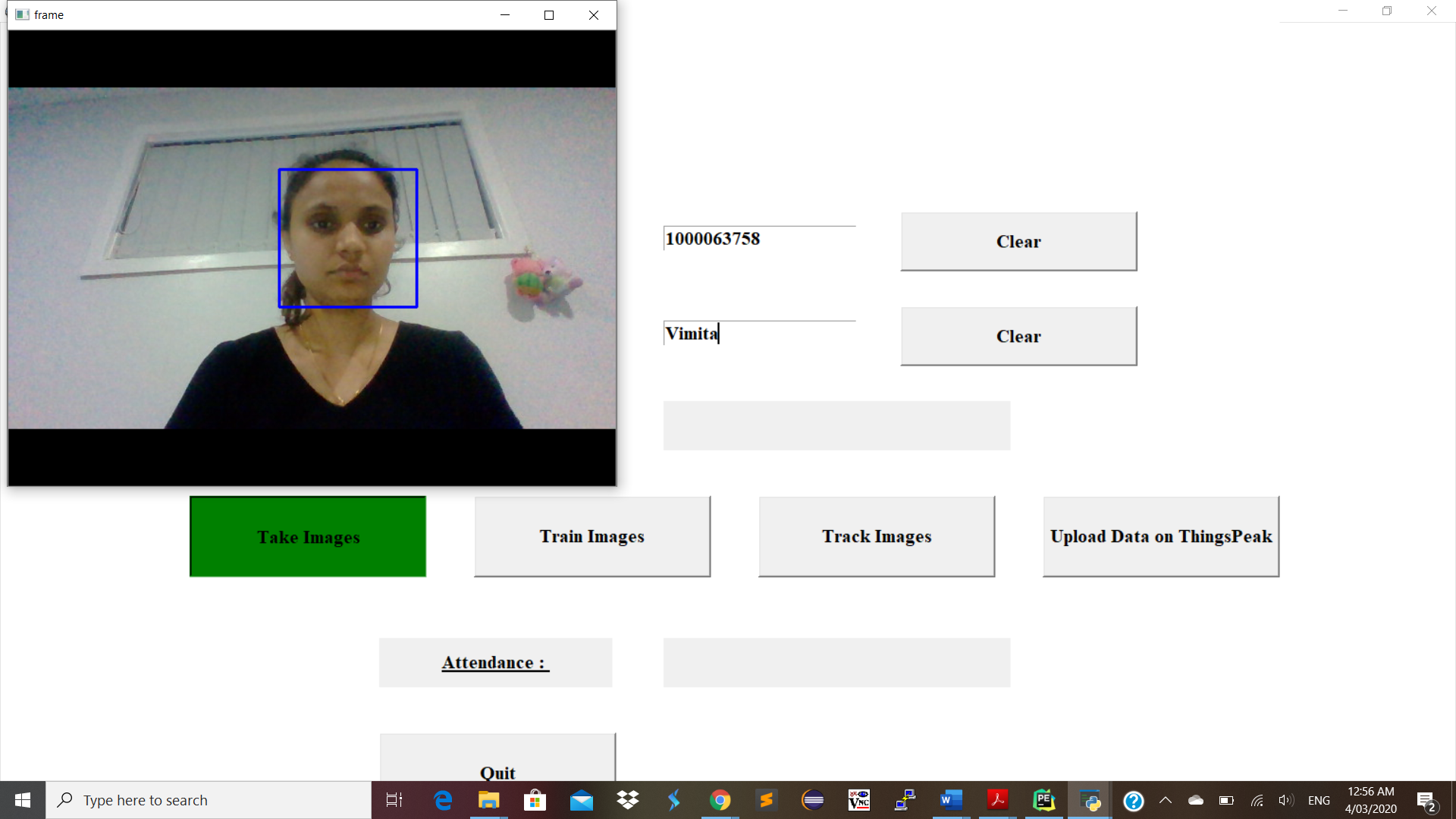


Figure 25 Testing 1

Entering student ID and name in the input box and by pressing Take Images button take photos for detection.

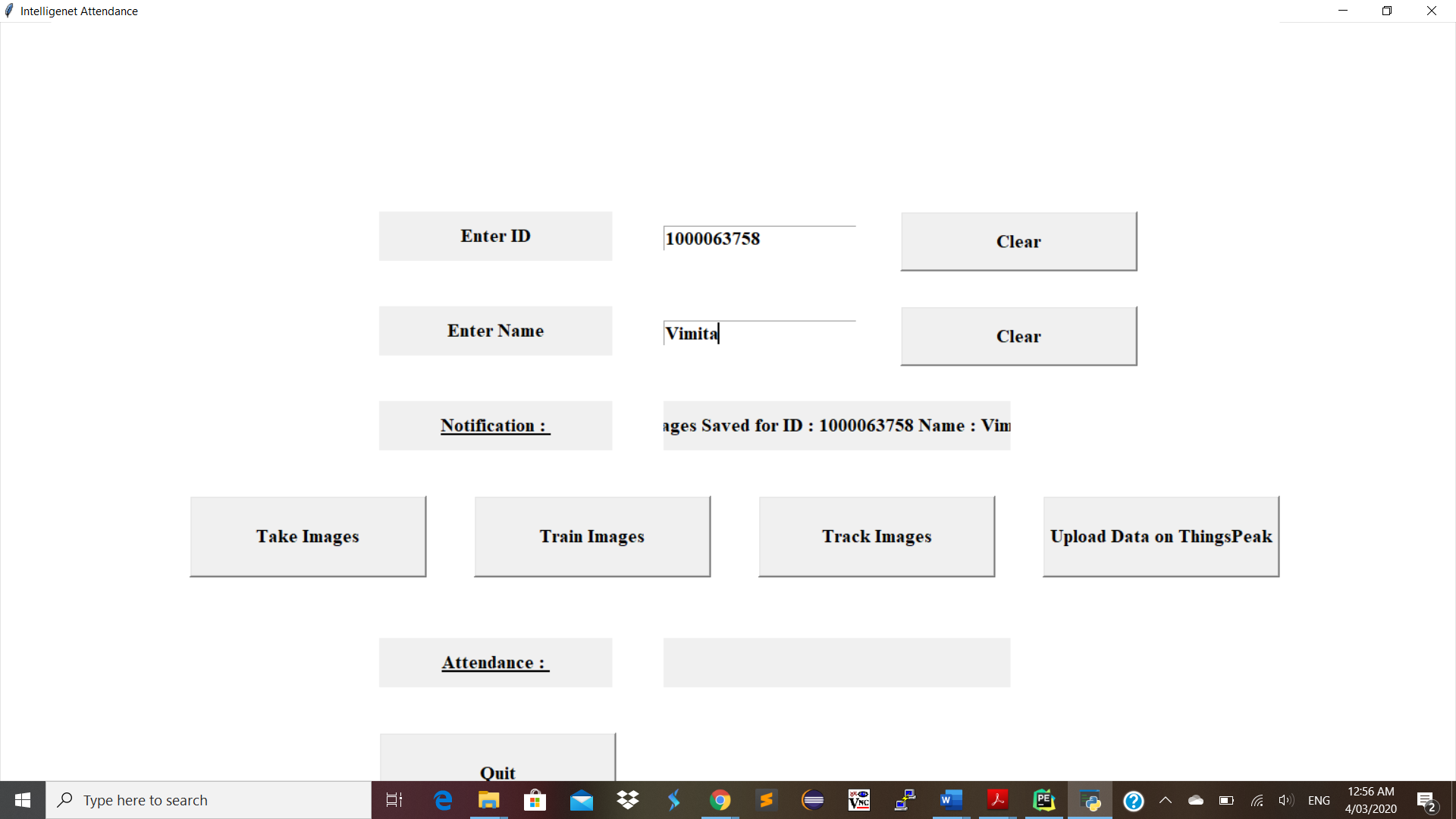


Figure 26 Testing 2

After entered ID, name and images need to store that image with student’s “name.ID.counter.jpeg “ format and trained system by clicking on Train Images button.

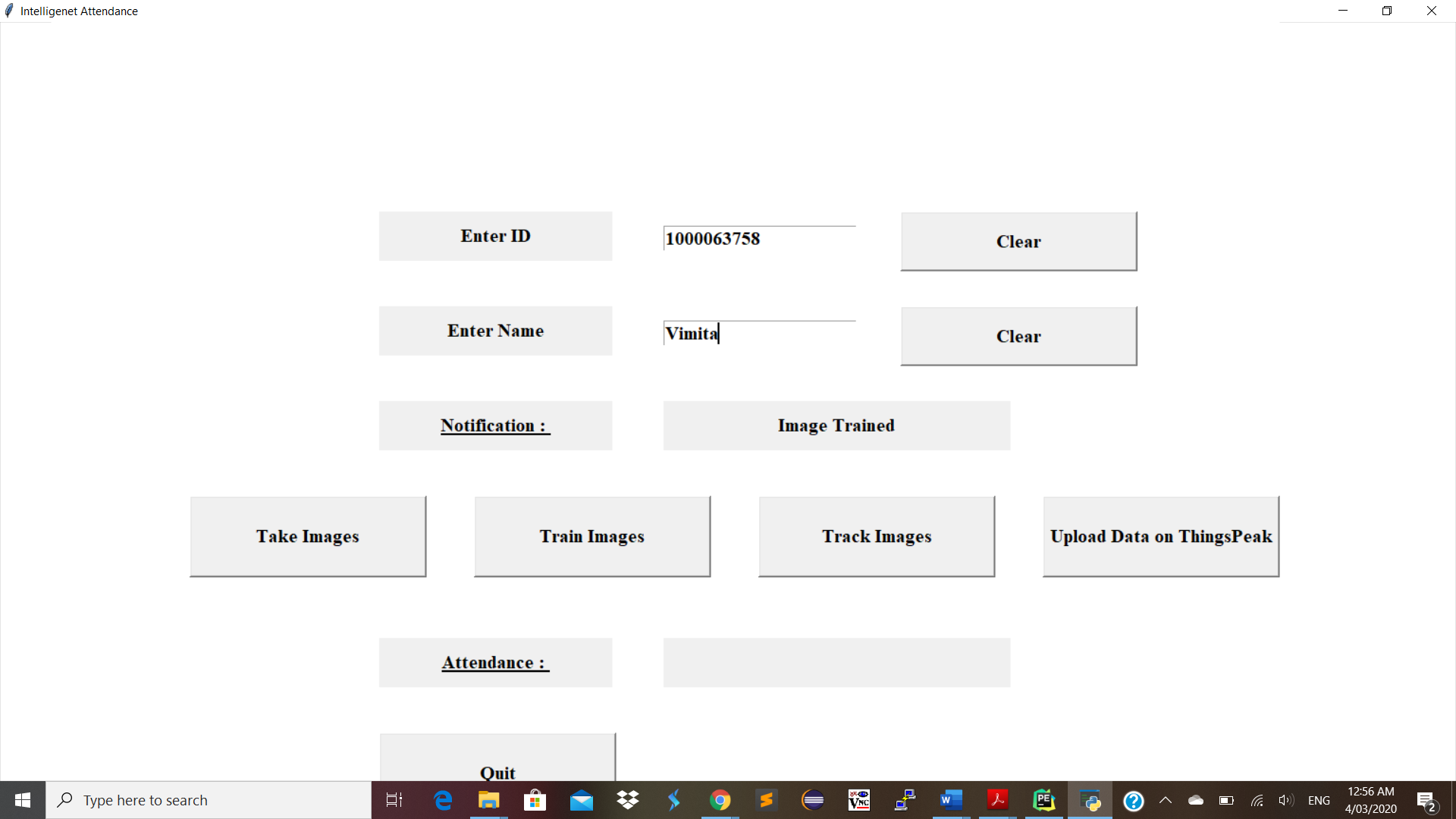


Figure 27 Testing 3

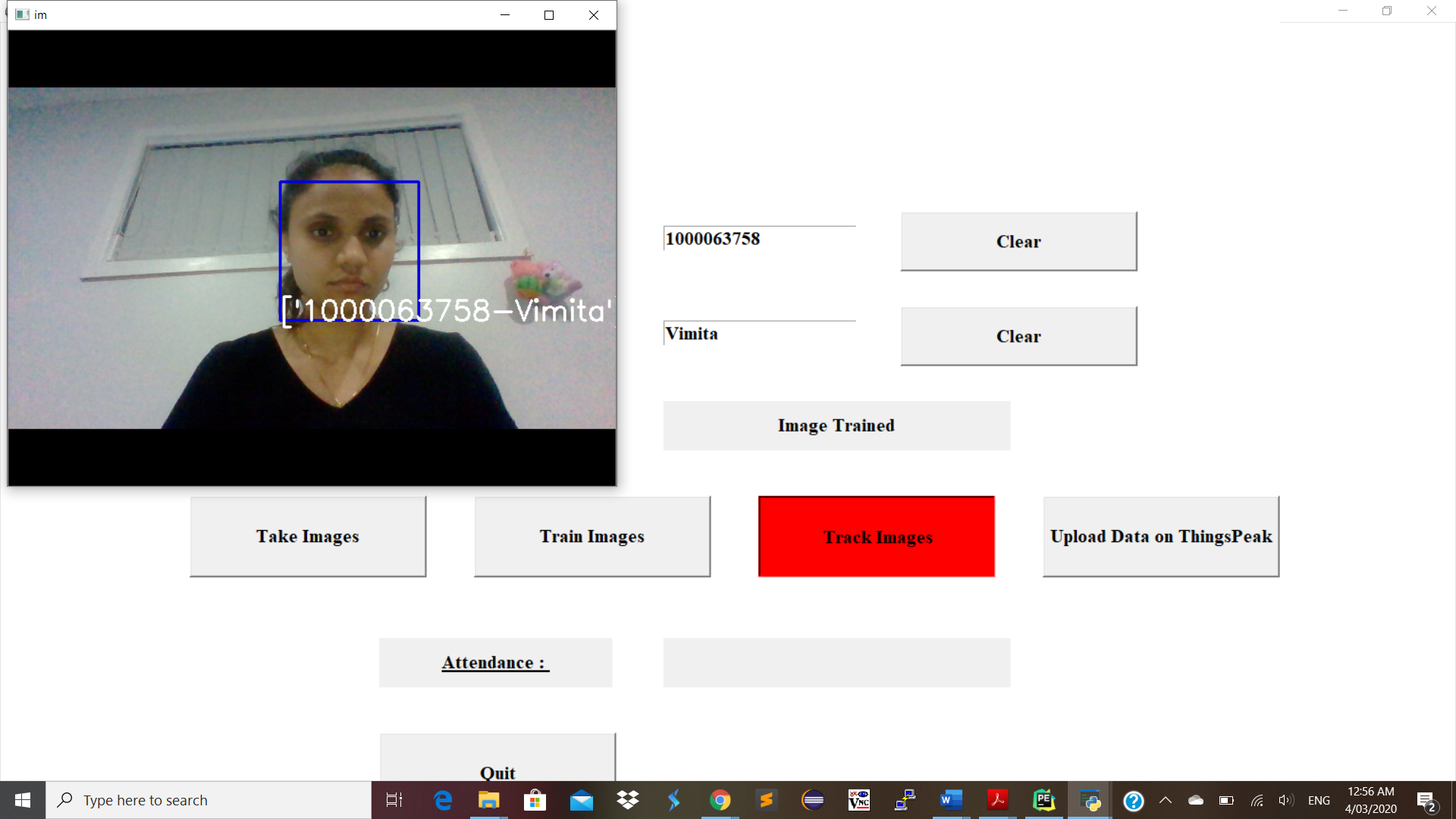


Figure 28 Testing 4

After a prepared system, need to check whether it identifies the image or not from the trained dataset by clicking on the “Track Images” button.

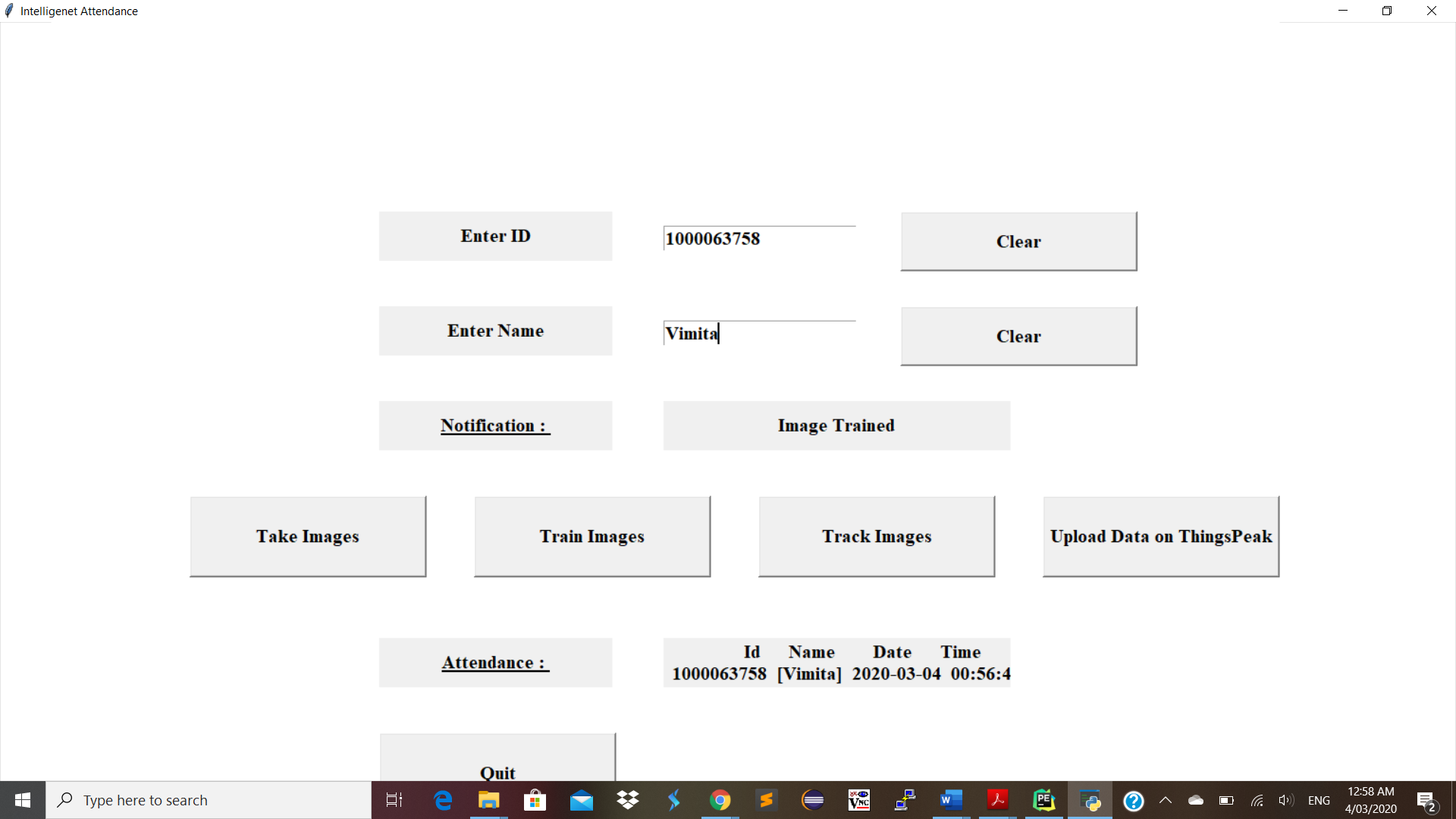


Figure 29 Testing 5

After successfully recognized face it stores an array of data including ID, Name, Date and Time in a CSV file to store student’s attendance.

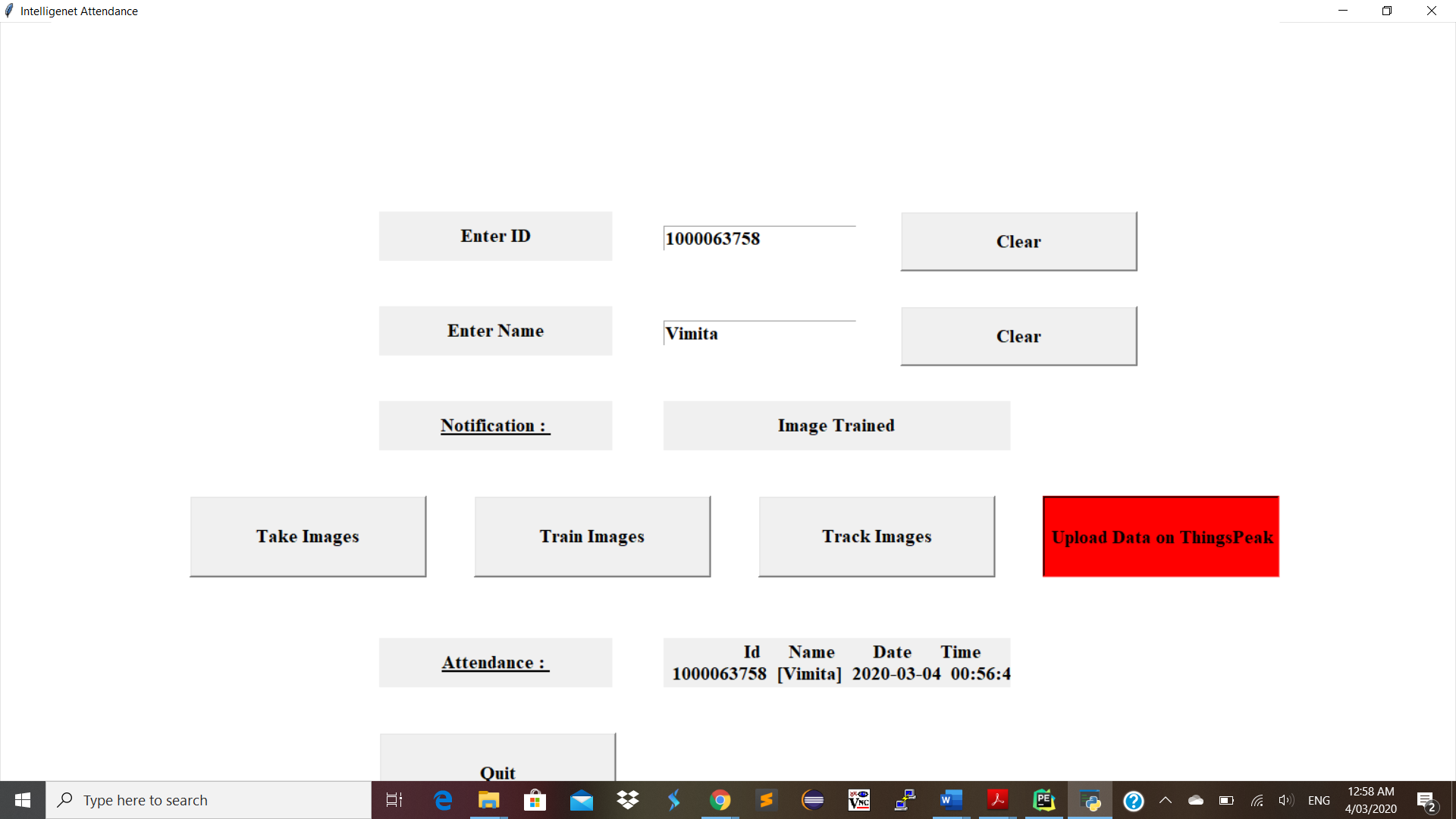


Figure 30 Testing 6

By clicking on Upload Data on ThingsPeak button it transfers data on the cloud.

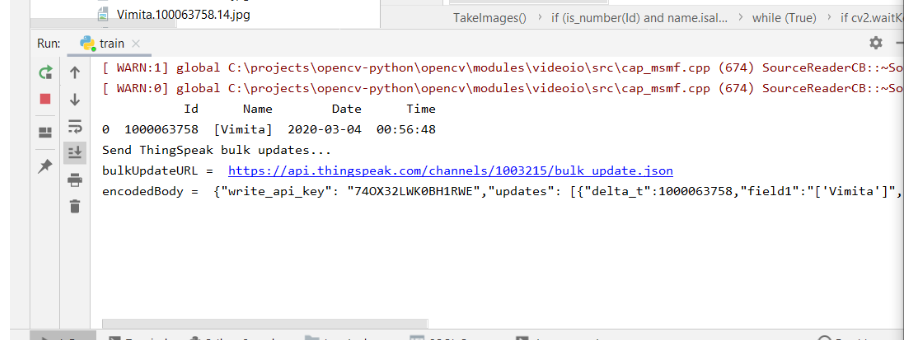


Figure 31 Testing 6.1

As a result, the data is transfer in a JSON format on the ThingsPeak.

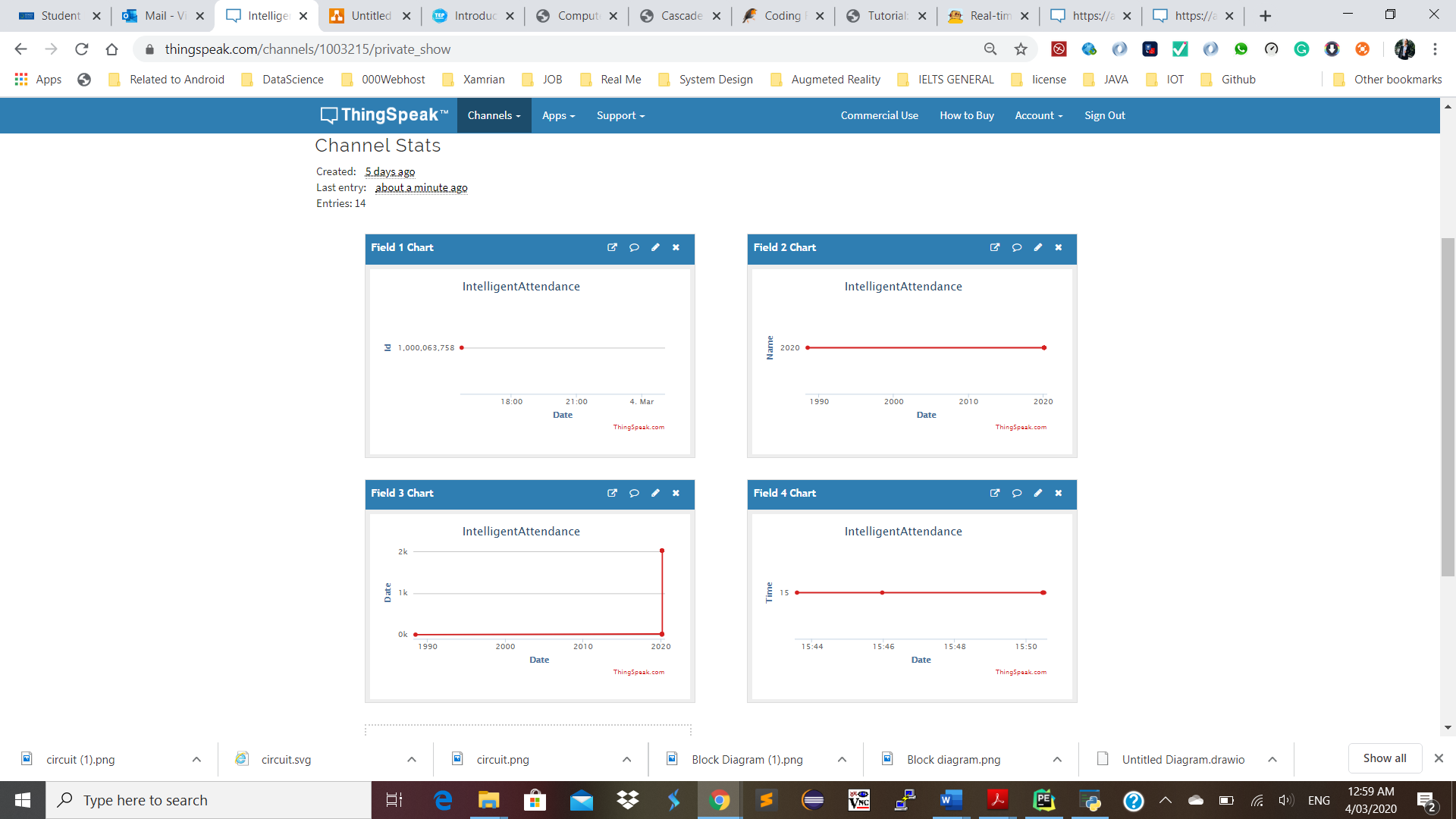


Figure 32 Testing 7

To visualize data on ThingsPeak, In my channel there is 4 field, which defines as ID, Name, Date and Time. ThingsPeak created a sperate chart for each field to visualize data.

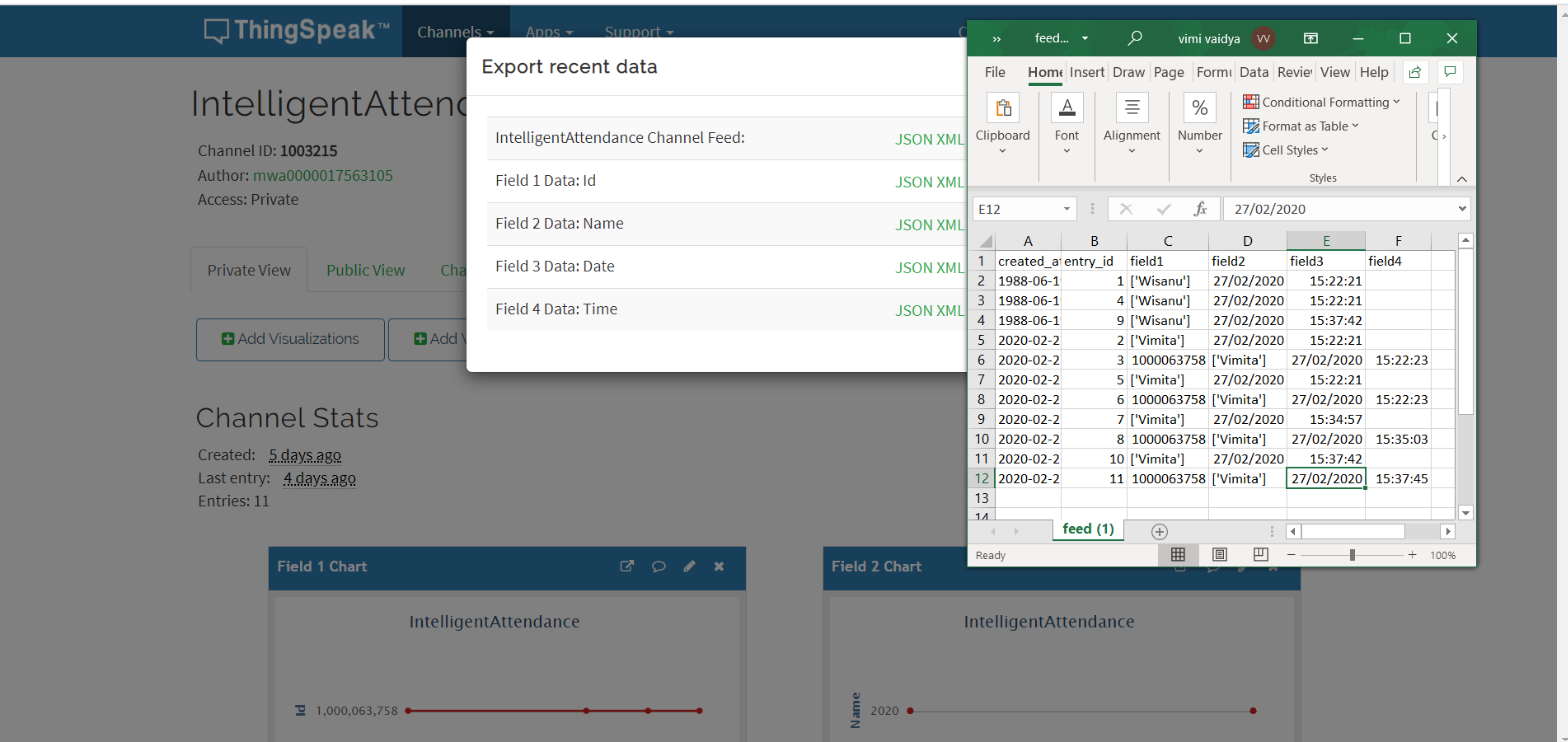


Figure 33 Visualize ThingsPeak uploaded data in a CSV file

To check student’s attendance by their ID, Name, Date and Time stamp by retrieving uploaded data from ThingsPeak in a CSV file.

## Security

Secure data on IoT device by creating extra network for internet of things device only. Because, Having a separate connection to act like a buffer will help to ensure that no outside entity is allowed to access your shared files and other kinds of encrypted data (Viswanathan, 2019).Therefore, the device's cyber security is safeguarded through the use of the private internet network that is only open to that specific classroom.

Reliability: The device improves performance and reduces the amount of time needed to take the attendance of the student.

The USB camera covers the entire corner and identifies the faces and enhances the student's profile.

Data is being transmitted safely by uploading data on cloud using HTTP POST request.

Wi-fi is suitable for small organisations or schools as per network protocol. However, different protocols can be alternatives to allow protected environments to transfer cloud data depending on the requirements of the application case.

# Conclusion

In this established system ten faces were identified and recognized, with each participant stored in ThingsPeak cloud service on the basis of a time stamp and date. In addition, the Raspberry Pi development board is an economically efficient, fully functional computing system that can be used in many different application areas. Through Raspberry Pi's use of Python and Open CV the project was versatile and adaptable for any future changes required. However raspberry pi's performance in some cases is not sufficient for multi-tasking.

# Future work

The same project can be used for multiple security applications, where authentication is required to access the respective system credentials.

* Allow application access for students and faculty to visualize their attendance report.
* In terms of resource utilization, a face recognition algorithm can be developed so that the software can identify more faces at a time and will enhance the system.
* Many variants can be created and used for home protection and personal or corporate benefits. With this framework we can also quickly monitor a specific student in an organisation.

# Reference List

B. Soewito, F. L. Gaol, E. Simanjuntak and F. E. Gunawan, "Attendance system on Android smartphone," 2015 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC), Bandung, 2015, pp. 208-211.

Khan, T. (2019, December 19). Computer Vision - Detecting objects using Haar Cascade Classifier. Retrieved from <https://towardsdatascience.com/computer-vision-detecting-objects-using-haar-cascade-classifier-4585472829a9>

Parekh, J. (2017, May 11). WiFi's evolving role in IoT. Retrieved from <https://www.networkworld.com/article/3196191/wifi-s-evolving-role-in-iot.html>

Roy. S,(2018), “Real-Time Object Detection on Raspberry Pi Using OpenCV DNN and MobileNet-SSD.” Retrieved from <https://heartbeat.fritz.ai/real-time-object-detection-on-raspberry-pi-using-opencv-dnn-98827255fa60>

Viswanathan, V. (2019, October 14). Eight ways to secure your data on IoT devices. Retrieved from <https://www.itproportal.com/features/eight-ways-to-secure-your-data-on-iot-devices/>

1. <https://raspberrytips.com/best-os-for-raspberry-pi/> [↑](#footnote-ref-1)
2. <https://medium.com/@mindfiresolutions.usa/what-is-pycharm-ide-cc0735784f64> [↑](#footnote-ref-2)
3. <https://www.theengineeringprojects.com/2018/07/introduction-to-raspberry-pi-3-b-plus.html> [↑](#footnote-ref-3)
4. <https://filext.com/file-extension/YML> [↑](#footnote-ref-4)
5. <https://en.wikipedia.org/wiki/YAML> [↑](#footnote-ref-5)
6. <https://au.mathworks.com/help/thingspeak/bulkwritejsondata.html> [↑](#footnote-ref-6)