

Final Year Project

Facial Recognition Attendance Monitor

Darragh Freeley

Bachelor of Software & Electronic Engineering

Galway-Mayo Institute of Technology

2019/2020

A screenshot of a cell phone

Description automatically generated

**Declaration**

This project is presented in partial fulfilment of the requirements for the degree of Bachelor of Engineering in Software & Electronic Engineering at Galway-Mayo Institute of Technology.

This project is my own work, except where otherwise accredited. Where the work of others has been used or incorporated during this project, this is acknowledged and referenced.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Acknowledgements**

I would like to thank my supervisor Brian O’Shea for his consistent support and guidance during the building of this project. Furthermore, I would like to thank my friends, family and all staff members of GMIT for their support throughout my college education.

**Table of Contents**

[1 Summary 6](#_Toc40806806)

[2 Introduction 8](#_Toc40806807)

[3 Project Architecture 9](#_Toc40806808)

[4 Development Platform and Tools 10](#_Toc40806809)

[5 Sensors 11](#_Toc40806810)

[6 Facial Recognition 13](#_Toc40806811)

[7 Microsoft’s Face API 16](#_Toc40806812)

[8 Machine Learning 19](#_Toc40806813)

[9 Web server 21](#_Toc40806814)

[10 Problem Solving 24](#_Toc40806815)

[11 Conclusion 25](#_Toc40806816)

[12 References 26](#_Toc40806817)

[13 Code 27](#_Toc40806818)

# Summary

The goal of my final year project was to build a small device that could be placed at the front of a lecture hall. This device would recognize and identify students faces, along with recognizing perceived emotions and transcribing the lectures audio. All data of the lecture would be accessible to lecturers and students online.

The core features of my project include,

* Identifying and recognizing faces using computer vision.
* Perceived emotion recognition.
* Storing recognized faces and emotions to a database via RESTful server.

My approach to this project utilized the Waterfall approach. The waterfall approach involves treating steps as separate phases. Firstly, I researched what was needed to get a project like this up and going. I had to decide on a development board, what services to use for facial recognition and how I would display the data on a GUI. After deciding on what services I would use, I planned out a timeline of when to have different aspects of my project completed. Once I had planned my timeline I began developing.

Technologies used in this project,

* Microsoft Face API handles face/emotion recognition.
* The Raspberry Pi 4 & Pi cam is where our code runs.
* OpenCV is used to capture images
* Django Rest server is used to save the data to a MySQL database.
* Python is the programming language used in developing this project.
* Html is used to display the returned results in a table on a webpage.
* Amazon S3 Cloud service stores the captured images.

My project currently captures an image every 15 seconds and recognizes trained faces along with attributes such as age, gender and emotion. This information is posted to a Django REST server and saved in a MySQL database. The image captured is uploaded to an AWS S3 bucket. Unfortunately, all I set out to accomplish was not achieved. I would’ve liked to implement audio transcription, but as of now, this is not a feature.

Overall, I feel this project went reasonably well. Most features are working. I’ve improved my Python programming skills and I have a greater understanding of how facial recognition and REST servers work.

# Introduction

Maintaining attendance in educational institutions is very important. Often attendance is taken using the traditional paper-based approach. This can be time consuming and is not very efficient, it can also be easily manipulated by other students signing for each other.

As humans we distinguish one another using our sight. We can have an idea who someone is if they’re facing away from us, but to confirm an identity we look for a face. Thanks to advancements in the field of computer vision, we can now use machines to identify faces. Under ideal conditions this can be more accurate than a human’s ability to recognise faces.

Another interesting development in the field of facial recognition is perceived emotion recognition. This is a machines ability to recognise a human’s emotional state. This can be useful data in a classroom as the emotional state of a person may affect concentration, task solving and decision-making skills. Identifying students who are experiencing these difficulties can be beneficial, as they can then be addressed. This can lead to enhanced productivity and higher grades overall.

We can combine the technology of a facial recognition system with perceived emotion analysis to automate the tedious task of attendance taking while also monitoring students emotional state.

The goal of my final year project was to build a small device that could be placed at the front of a lecture hall. This device would recognize and identify students faces, along with recognizing perceived emotions and transcribing the lectures audio. All information would be accessible to lectures and students online.

I chose to pursue a facial recognition project because I was interested in Computer Vision, Facial Recognition and Machine learning, so I tried to find a useful application to fit around it. Being in classes and either signing a role sheet or having a role call at the start of lectures, I decided to apply the technology to attendance taking, to automate the task. Then I began to expand on the idea.

# Project Architecture

For this project I used a Raspberry Pi 4 as my development board along with a Pi cam. I used many cloud-based services such as Microsoft’s Face API for facial recognition, AWS S3 storage to store any images captured., and a Django REST server to save any data returned to a database. I mainly used the Thonny IDE as my development environment.

A close up of a logo

Description automatically generated

Figure 4‑1 Architecture Diagram

# Development Platform and Tools

* The main IDE I used to create my project was ‘Thonny’, this is a Python IDE that comes pre-installed on the Raspbian operating system onboard the Raspberry Pi. It’s a very lightweight and basic IDE. I also used ‘Pycharm’ when working off my Windows machine.
* The Raspberry Pi 4 Model B is a small low-cost single-board computer. It has a 1.5GHz 64-bit quad core ARM Cortex-A72 processor along with 4GB of RAM. Raspberry Pi computers are a great way for people to explore computing and learn how to program in languages like Scratch and Python.
* The Raspberry Pi Camera is a module that can be attached to the Pi board. This allows the user to have camera capabilities.
* Amazon S3 is a scalable, high-speed, web-based cloud storage service designed for online backup and archiving of data and applications on Amazon Web Services.
* Django is a Python-based free and open-source web framework. Frameworks are a collection of modules that make development easier. They allow you to create applications or websites from an existing source, instead of from scratch. Django REST framework is a toolkit for buildings Web APIs
* OpenCV is a library of programming functions mainly aimed as real-time computer vision.
* The Face API is part of Microsoft cognitive services set of AI cloud services. Its main features are
  + Detect human faces
  + Find similar faces
  + Group faces based on similarity
  + Identify trained faces
  + Recognise the perceived emotion by facial expressions (anger, happiness, sadness)

# Sensors

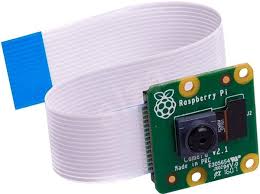
This project only used 1 sensor. This was the Raspberry Pi camera V2 module. The Raspberry Pi camera module can be attached to any Raspberry Pi board. This allows the user to implement camera features to their project.

Figure 5-1 Raspberry Pi Camera module V2

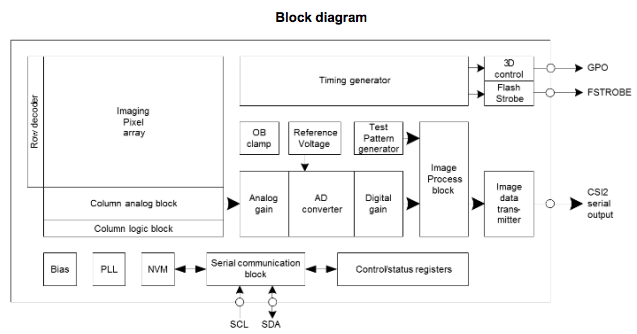


Figure 5-2 Sony IMX219 block diagram

* This module has an 8-megapixel Sony IMX219 sensor and can be used to take high-definition video, as well as still images. It supports 1080p at 30 fps and 720p at 60 fps. It attaches via a 15cm ribbon cable to the CSI port on the Raspberry PI. [1]
* It works with all models of the Raspberry Pi. It can be accessed through MMAL and V4L APIs, and there are numerous third-party libraries built for it, including the Pi-camera Python library.
* This module is very popular in-home security applications, and in wildlife camera traps.



Figure 5-3 Simple Code that Streams real-time video

# Facial Recognition

Facial recognition is a way of recognising human faces through technology. Face recognition systems can be used to identify people in photos, video, or in real-time. A facial recognition system uses biometrics to map facial features from an image or video. [2]

**How it works**

A face is detected in an image or video. Facial detection algorithms typically start by searching for human eyes, these are one of the easiest features to detect. The algorithm then moves on and tries to detect other facial features such as eyebrows, mouth, nose and nostrils.

Once a face is detected, facial recognition software reads the geometry of the face. Some of the key factors are the distance between the eyes and the distance between the chin and forehead. The software identifies facial landmarks. Examples of facial landmarks are the tip of the nose, corners of the mouth and pupils in the eyes. These landmarks are key to identifying your face and results in a facial signature.

A facial signature is a mathematical formula of your face. This signature is then compared to a database of other known facial signatures.

A result is returned when a facial signature matches with a face in the database.

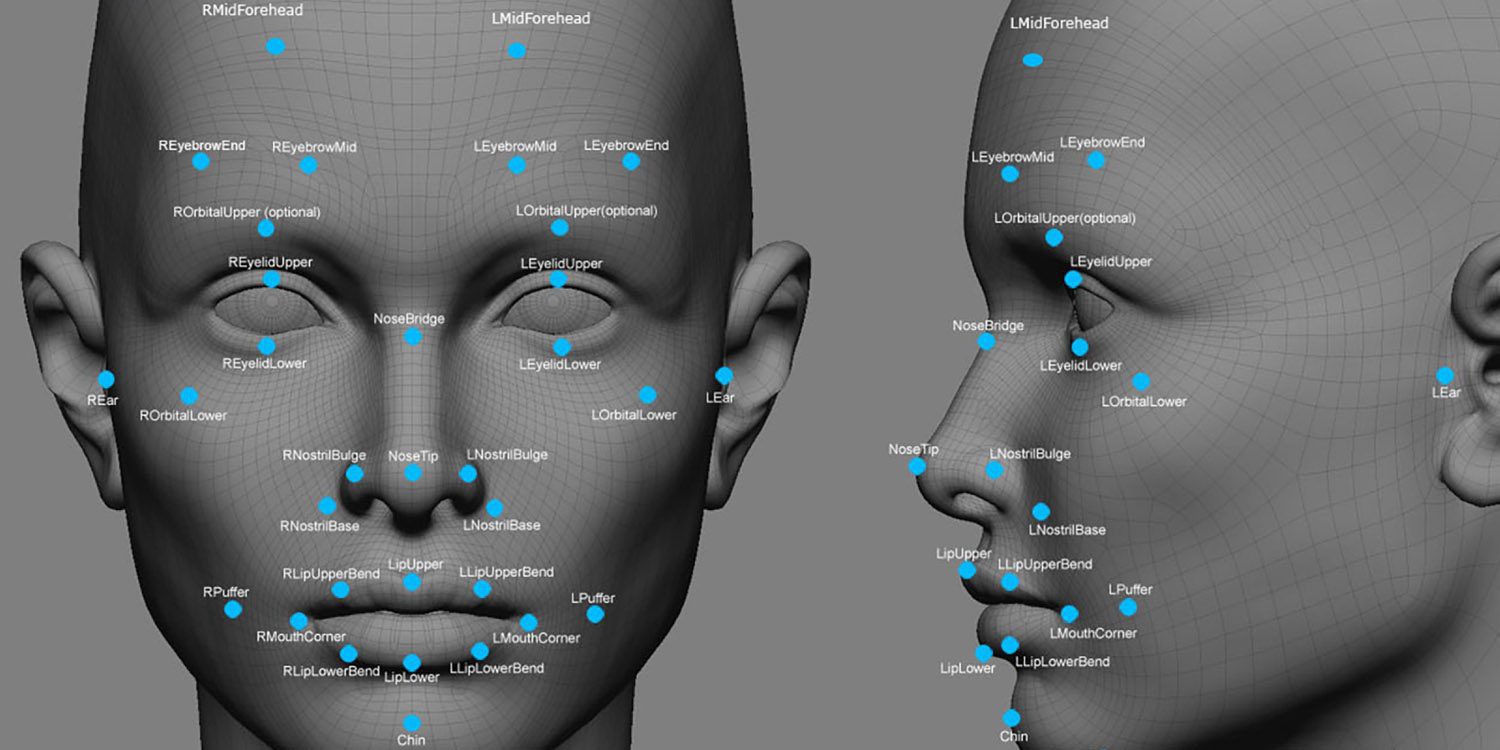


Figure 6-1 Facial landmark points.

Face recognition systems vary in their ability to identify people under challenging conditions such as poor lighting, low quality image resolution, and suboptimal angle of view (such as in a photograph taken from above looking down on an unknown person). Other problems in facial recognition are identifying a face that has aged and a face that has a different expression. These problems are also known as A-PIE, Age, Pose, Illumination and Expression.

When it comes to errors, there are two key concepts to understand:

A “false negative” is when the face recognition system fails to match a person’s face to an image that is, in fact, contained in a database. In other words, the system will return zero results in response to a query.

A “false positive” is when the face recognition system does match a person’s face to an image in a database, but the match is incorrect.

When researching a face recognition system, it is important to look closely at the “false positive” rate and the “false negative” rate. For example, if you are using face recognition to unlock your phone, it is better if the system fails to identify you a few times (false negative) than it is for the system to misidentify other people as you and lets those people unlock your phone (false positive). [3]

**Applications**

Facial recognition is becoming increasingly popular. Some of us even use this technology daily when unlocking our phones. Other applications of facial recognition are,

* Preventing retail crime.
* Targeted advertising.
* Validating ATM transactions.
* Tracking attendance.
* Diagnosing Diseases.

[4]

**Concerns**

Facial recognition has triggered serious concerns over the impact facial recognition could have on human rights. Some of these concerns are,

* It operates without a clear legal or regulatory framework.
* It violates our right to privacy.
* It is often inaccurate.
* It can lead to automation bias.
* It can be used to target already venerable groups.

[5]

# Microsoft’s Face API

The Face API is very well documented which was very beneficial when developing my project.

The Face API is part of Microsoft’s Cognitive Services set of AI cloud services. It gives you access to some of the most advanced face recognition algorithms. The main features of Face API are facial detection, facial recognition and emotion recognition. This allows you to   
Detect human faces  
• Find similar faces  
• Group faces based on similarity  
• Identify previously tagged people using their faces  
• Identify the underlying emotion for each face in an image (anger, content, fear, happiness etc.)

**Facial Detection**

This feature enables you to detect a face in an image along with attributes of the face, such as Age, Emotion, Gender, Pose, Smile and Facial hair. In this image, we can see the API has detected a face and outlined it with a purple rectangle. [6]



Figure 7-1 Detected Face highlighted by purple rectangle

The below JSON has the data returned from this image.

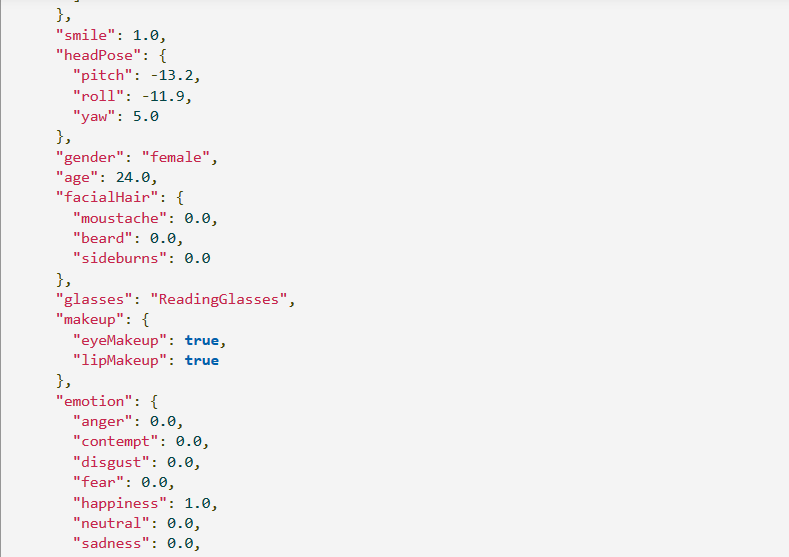


Figure 7-2 JSON returned from image 7-1

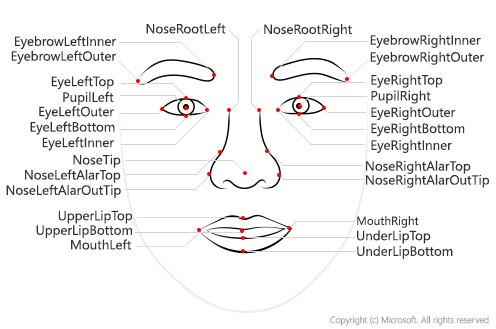
There are 27 face landmarks the Face API uses. [7]

Figure 7-3 MS Face Facial landmark points.

**Emotion recognition**

This feature detects perceived facial expressions such as anger, contempt, disgust, fear, happiness, neutral, sadness, and surprise. Given an image it will return a JSON response with an accompanying set of confidence scores for each emotion. The confidence score ranges from 0 to 1. Where 0 is a low confidence and 1 is a high confidence.

# Machine Learning

Machine learning is an application of artificial intelligence that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly. [8]

**How it works**

Machine learning uses two types of techniques,

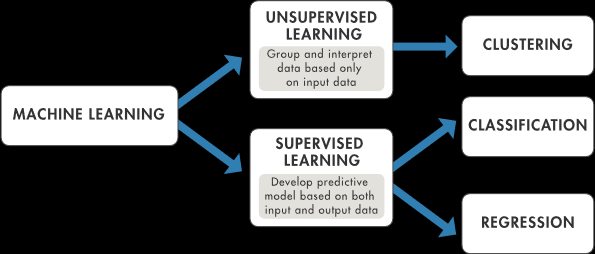
* Supervised learning, which trains a model on known input and output data so that it can predict future outputs.
* Unsupervised learning, which finds hidden patterns or intrinsic structures in input data.

Figure 8-1 Machine Learning Paths

A Supervised machine learning algorithm takes a known set of input data and known responses to the data and trains a model to generate reasonable predictions for the response to new data.

An Unsupervised learning finds hidden patterns or intrinsic structures in data. It is used to draw inferences from datasets consisting of input data without labeled responses.[9]

**Applications**

Machine learning is being used in various industries and professions. Some of the applications of Machine Learning are,

* Image Recognition
* Speech Recognition
* Medical Diagnoses
* Information Extraction

# Web server

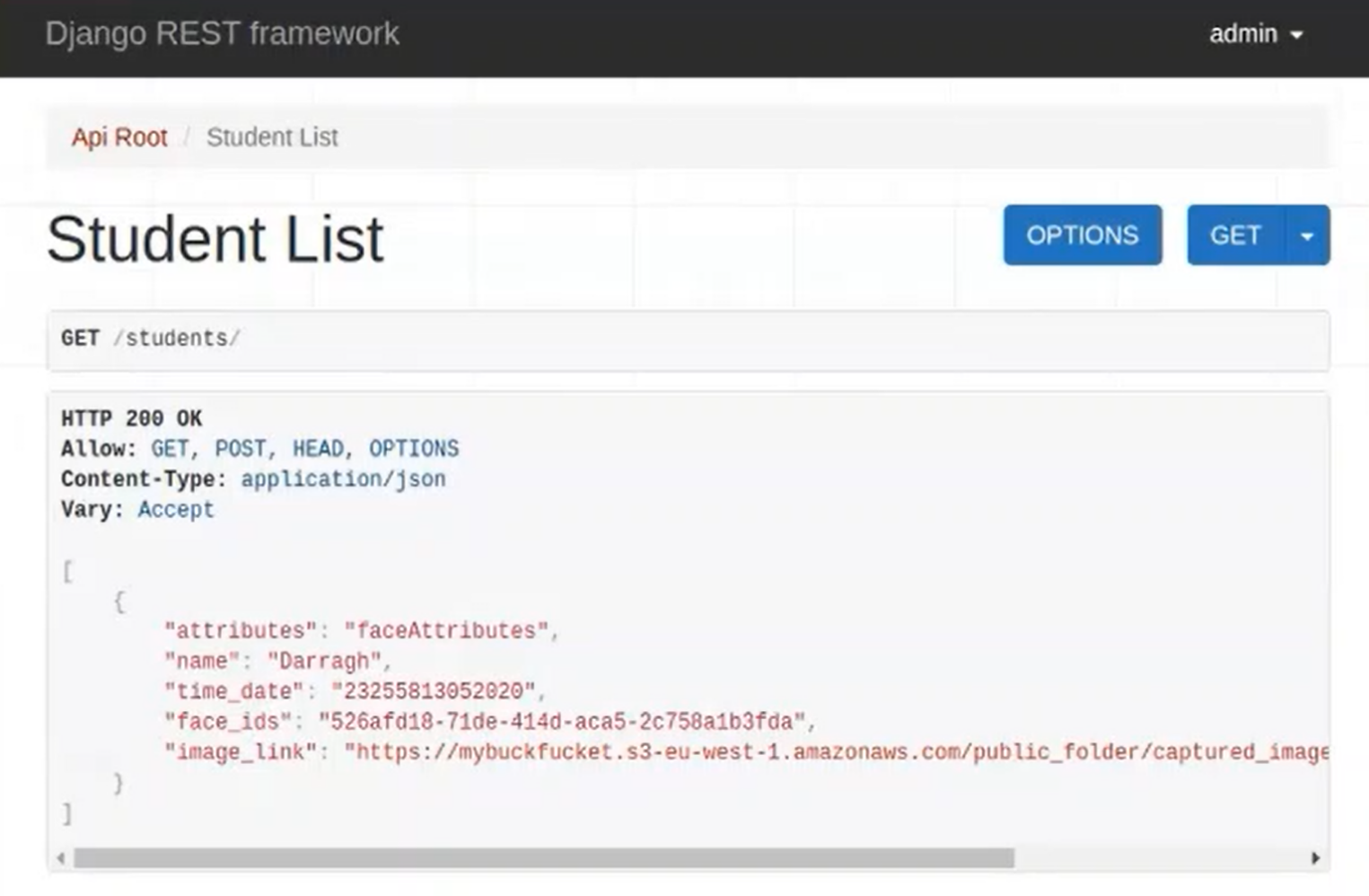
I used a Django REST server to store the data returned from the Face API to a MySQL database.

Figure 9-1 JSON saved in MySQL database on Django Server.

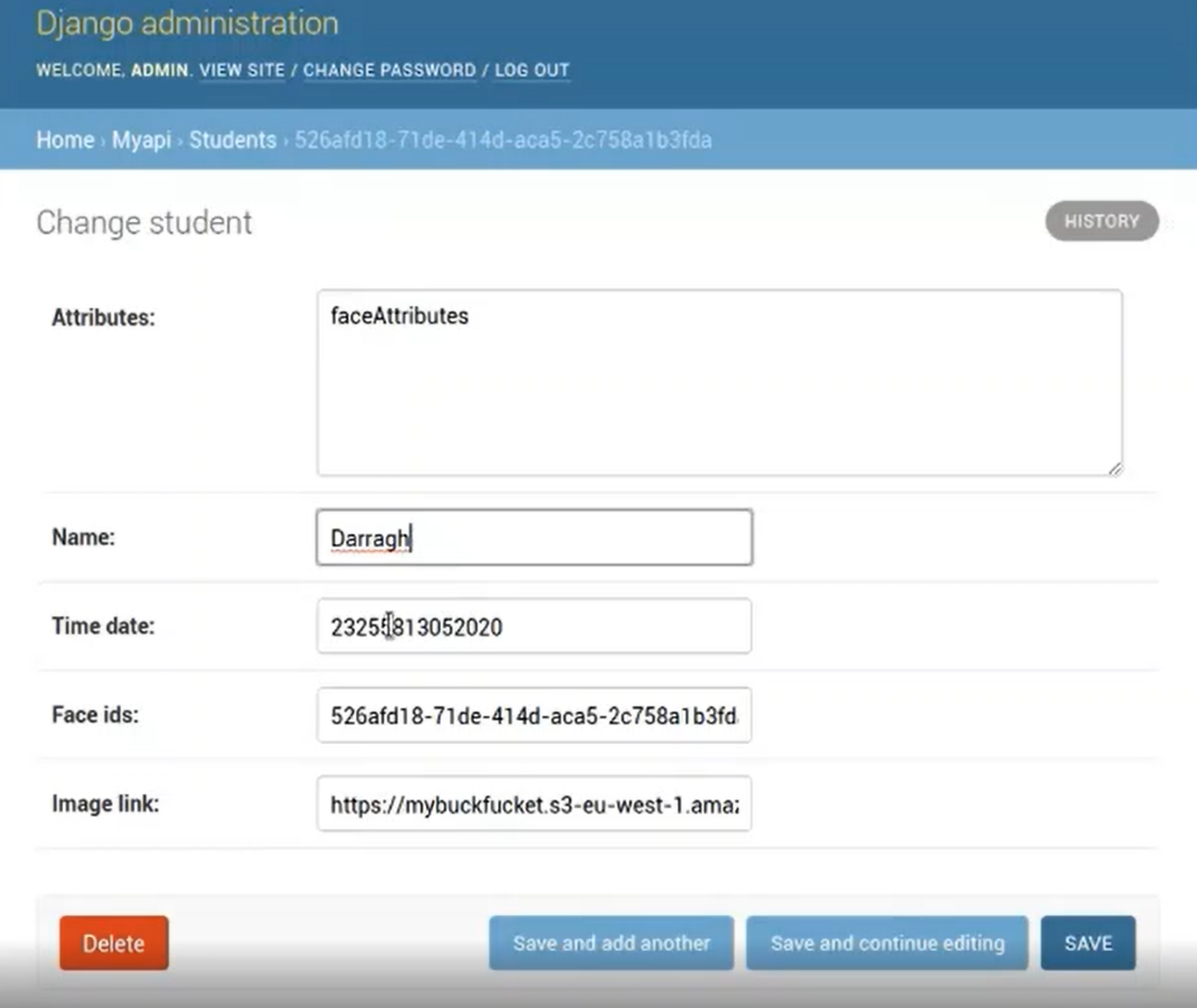


Figure 9-2 JSON saved in MySQL database on Django Server.

After the Face API returns the different attributes and ID of a face, my main python code then posts selected elements of the data to my Django server. The server saves this information in a database. Here I post the attributes of the face, the name, the time it was taken, the ID of the face identified and the link to the image stored online.

I also used to data to fill a table using HTML.

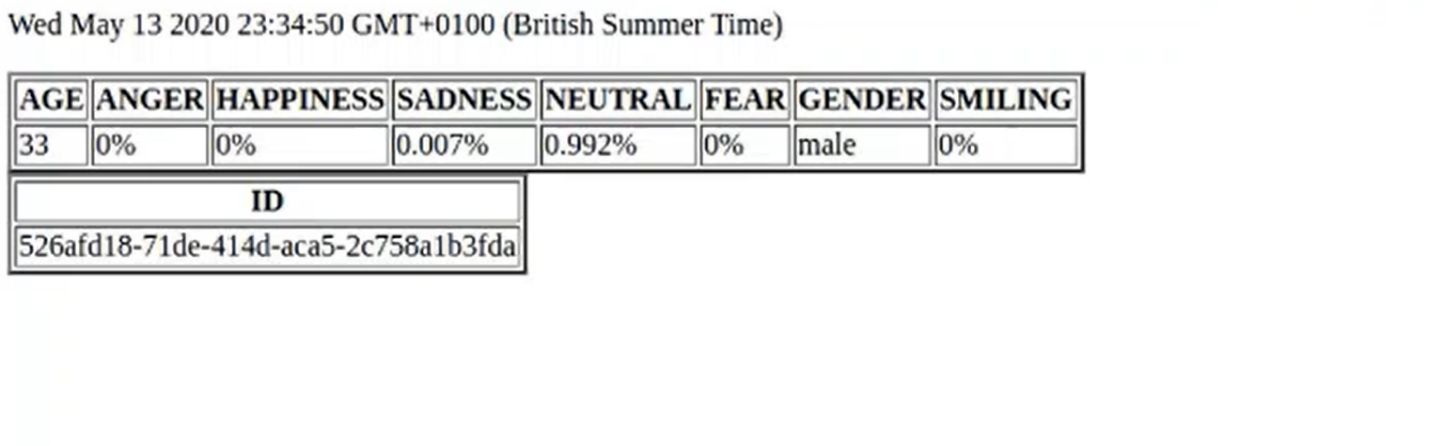


Figure 9-2 JSON data displayed in a table.

Django is a Python-based free and open-source web framework that follows the model-template-view architectural pattern. It handles a lot of the hassle of Web development.

Some alternatives to Django are Flask, web2py and Express.js.

**What is a REST API?**

A RESTful API is an application program interface (API) that uses HTTP requests to GET, PUT, POST and DELETE data. REST (representational state transfer) is an architectural style and approach to communications often used in web services development.

RESTful Web Service does not keep a client’s state on the server. This restriction is called Statelessness.

RESTful APIs use existing HTTP methodologies such as GET, PUT, Post and DELETE. They also use HTTPs status codes.

A RESTful API breaks a transaction into smaller modules. Each module addresses a distinct part of the transaction. This gives developers a lot of flexibility. [10]

# Problem Solving

I faced many problems throughout this project. With some being very frustrating and time consuming. I have learned something from each problem I encountered.

**Problem 1:**

The Raspberry Pi on multiple occasions refused to boot into the Raspbian OS. Sometimes this could be easily solved by clearing the cache and rebooting, but on two of these occasions I was forced to completely reinstall the Raspbian OS because I ran out of memory and the system became corrupted. In order to prevent this from repeatedly happening, memory management was important. My 16GB SD didn’t have much free memory for other files after OpenCV was installed. So I had to remove a lot of the unnecessary bloatware that comes pre-installed on the Raspbian OS, such as ‘wolfram-engine’ and ‘libreoffice’. This freed up space allowing me to continue my project without the risk of corrupting the system for a third time.

**Problem 2**:

When training faces on Microsoft’s Face API, I ran into errors. I trained my first group of faces with no errors. I called this group ‘students’. When trying to add more faces to this group, I was not deleting the previously existing ‘students’ group first, and was returned errors. To solve this issue, I added a method to delete a pre-existing group if it was already trained.

**Problem 3**

I first tried to implement a Nodejs server to store and show my data, as I had a small bit of prior knowledge from other modules in college, but it was unsuccessful. I looked into alternatives to Nodejs and found a great tutorial in the Django documentation. I followed the tutorial and had the Django server working in the same day.

# Conclusion

Overall, I’m happy with the end result of my project. I accomplished the main goals of my project, which was a facial recognition attendance monitor with emotion recognition. Unfortunately, I never got a chance to fully implement the audio transcription into the project, so I am disappointed in that. There are other areas of my project I’d like to improve, such as the GUI. A better GUI would give the project more of a finished look, as it looks very primitive as of now.

Throughout the project I learned about many technologies that I was previously interested in but never took the time to research fully, such as Machine Learning and Facial Recognition. In addition to learning about these technologies, I improved my Python programming skills, A language that I had not much experience in. But a language that is very useful to know. Lastly, I learned more about RESTful APIs, which were always coming up on my work placement the previous year. I never really understood them until I used them in this project.

# References

[1] Raspberrypi.org. 2020. *Camera Module - Raspberry Pi Documentation*. [online] Available at: <https://www.raspberrypi.org/documentation/hardware/camera/>

[2] Us.norton.com. 2020. How Does Facial Recognition Work?. [online] Available at: <https://us.norton.com/internetsecurity-iot-how-facial-recognition-software-works.html>

[3] Electronic Frontier Foundation. 2020. *Face Recognition*. [online] Available at: <https://www.eff.org/pages/face-recognition>

[4] FaceFirst Face Recognition Software. 2020. *21 Amazing Uses For Face Recognition – Facial Recognition Use Cases*. [online] Available at: <https://www.facefirst.com/blog/amazing-uses-for-face-recognition-facial-recognition-use-cases/>

[5] The Conversation. 2020. *Facial Recognition: Ten Reasons You Should Be Worried About The Technology*. [online] Available at: <https://theconversation.com/facial-recognition-ten-reasons-you-should-be-worried-about-the-technology-122137>

[6] Azure.microsoft.com. 2020. *Facial Recognition | Microsoft Azure*. [online] Available at: <https://azure.microsoft.com/en-us/services/cognitive-services/face/#demo>

[7] Docs.microsoft.com. 2020. *Face Detection And Attributes Concepts - Azure Cognitive Services*. [online] Available at: <https://docs.microsoft.com/en-us/azure/cognitive-services/face/concepts/face-detection>

[8] Expert System. 2020. *What Is Machine Learning? A Definition - Expert System*. [online] Available at: <https://expertsystem.com/machine-learning-definition/>

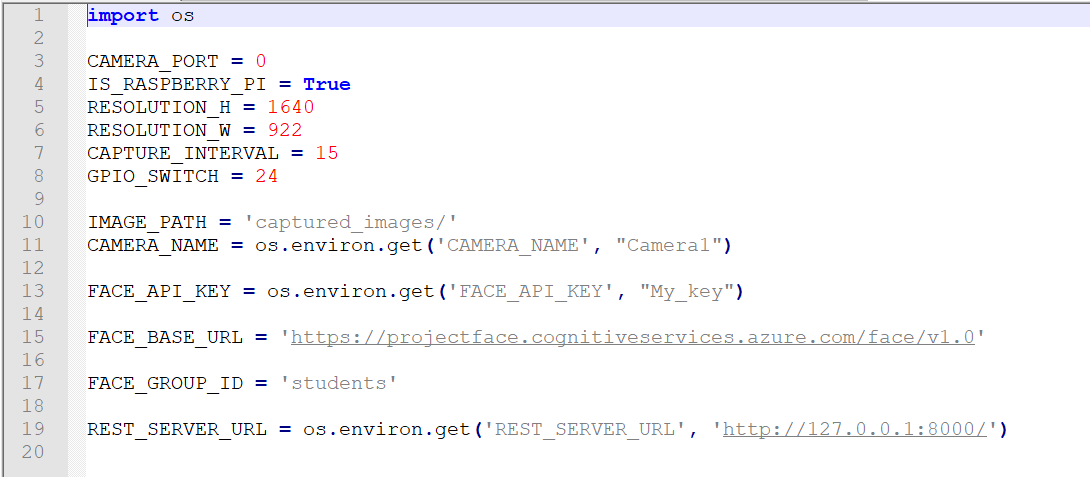
[9] Uk.mathworks.com. 2020. *What Is Machine Learning? | How It Works, Techniques & Applications*. [online] Available at: <https://uk.mathworks.com/discovery/machine-learning.html>

[10] SearchAppArchitecture. 2020. *What Is A Restful API (REST API) And How Does It Work?*. [online] Available at: <https://searchapparchitecture.techtarget.com/definition/RESTful-API>

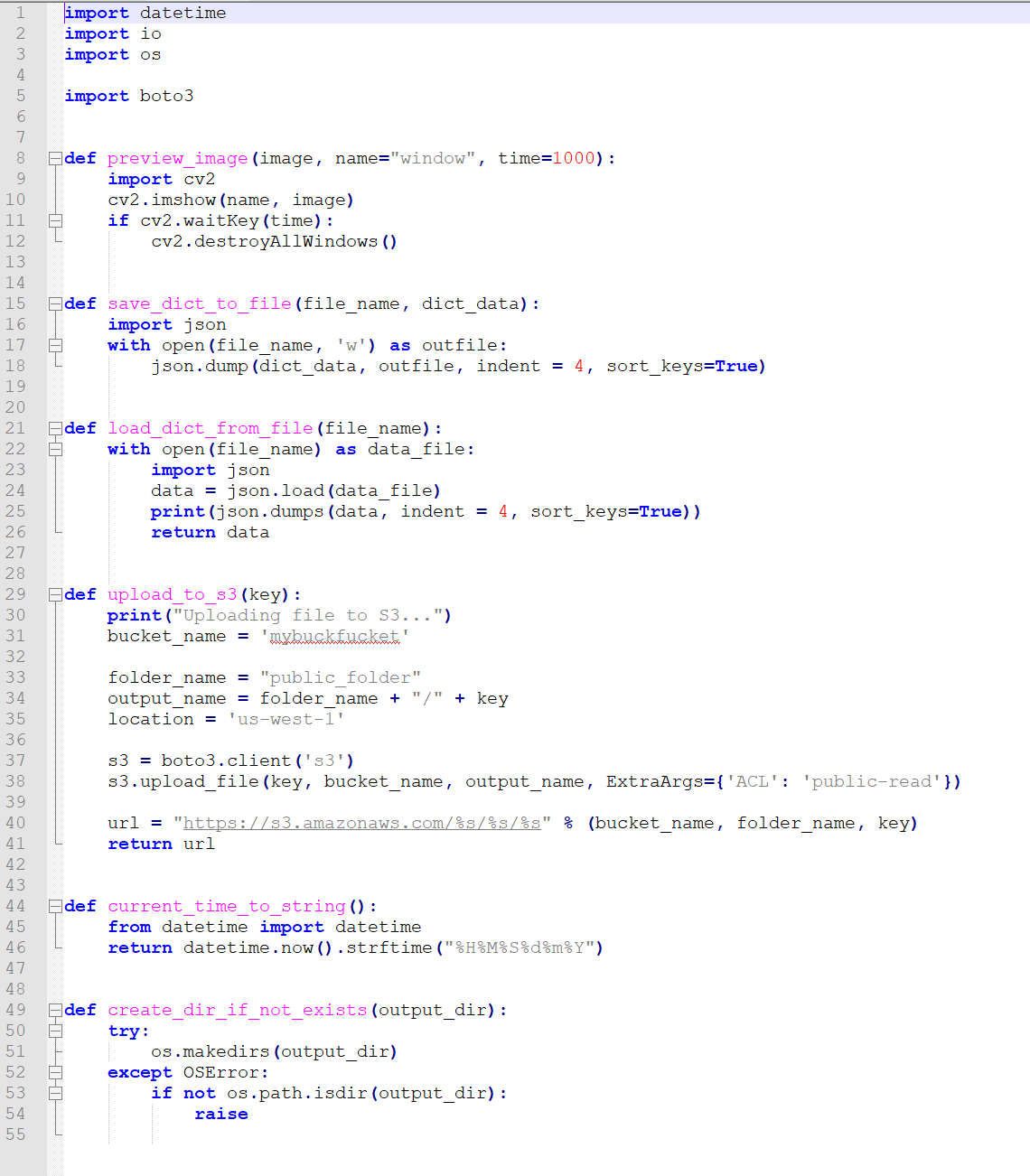
# Code

GITHUB Repo: <https://github.com/g003361711996/Facial-Recognition-Attendance-Monitor>

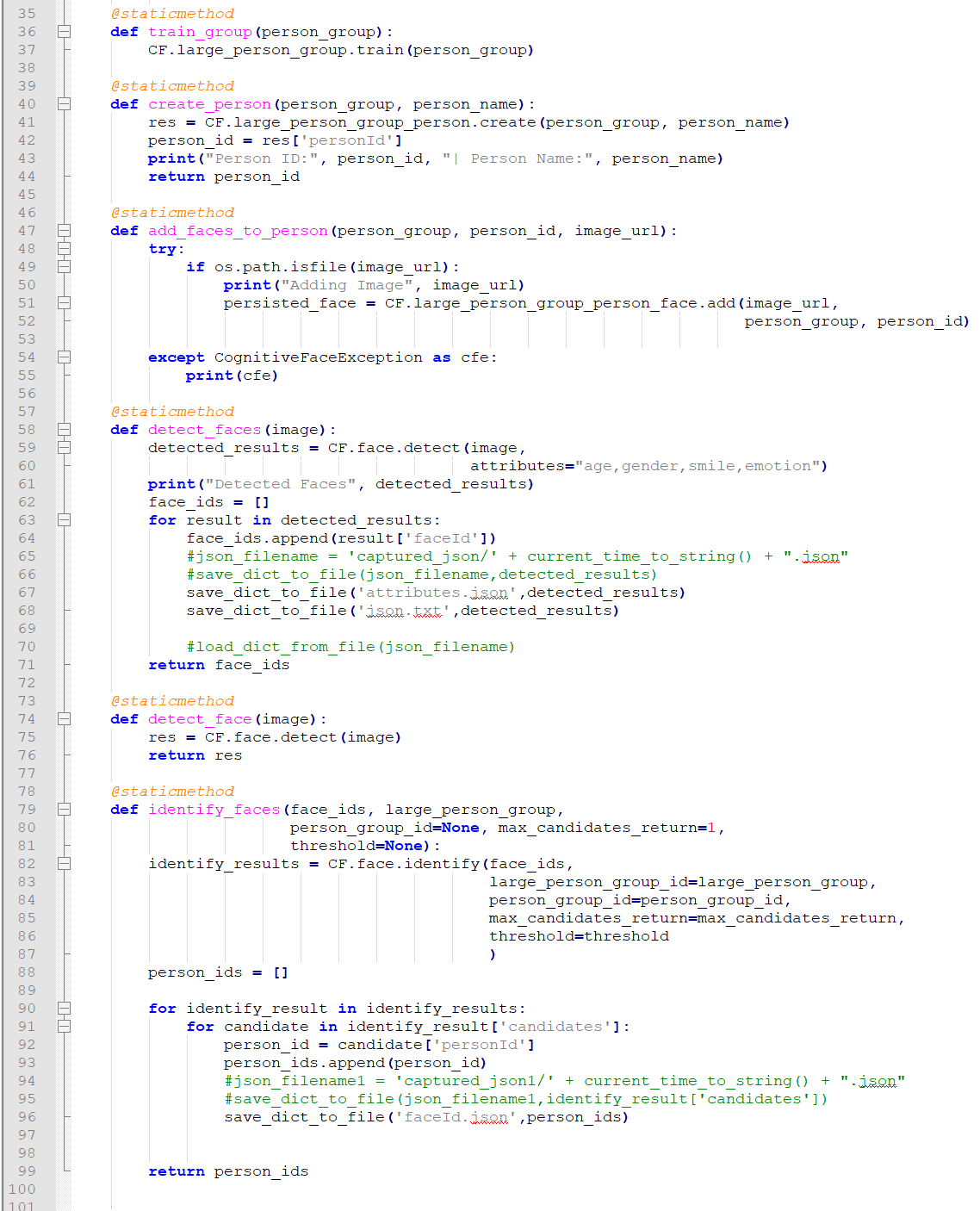
*Constants.py*



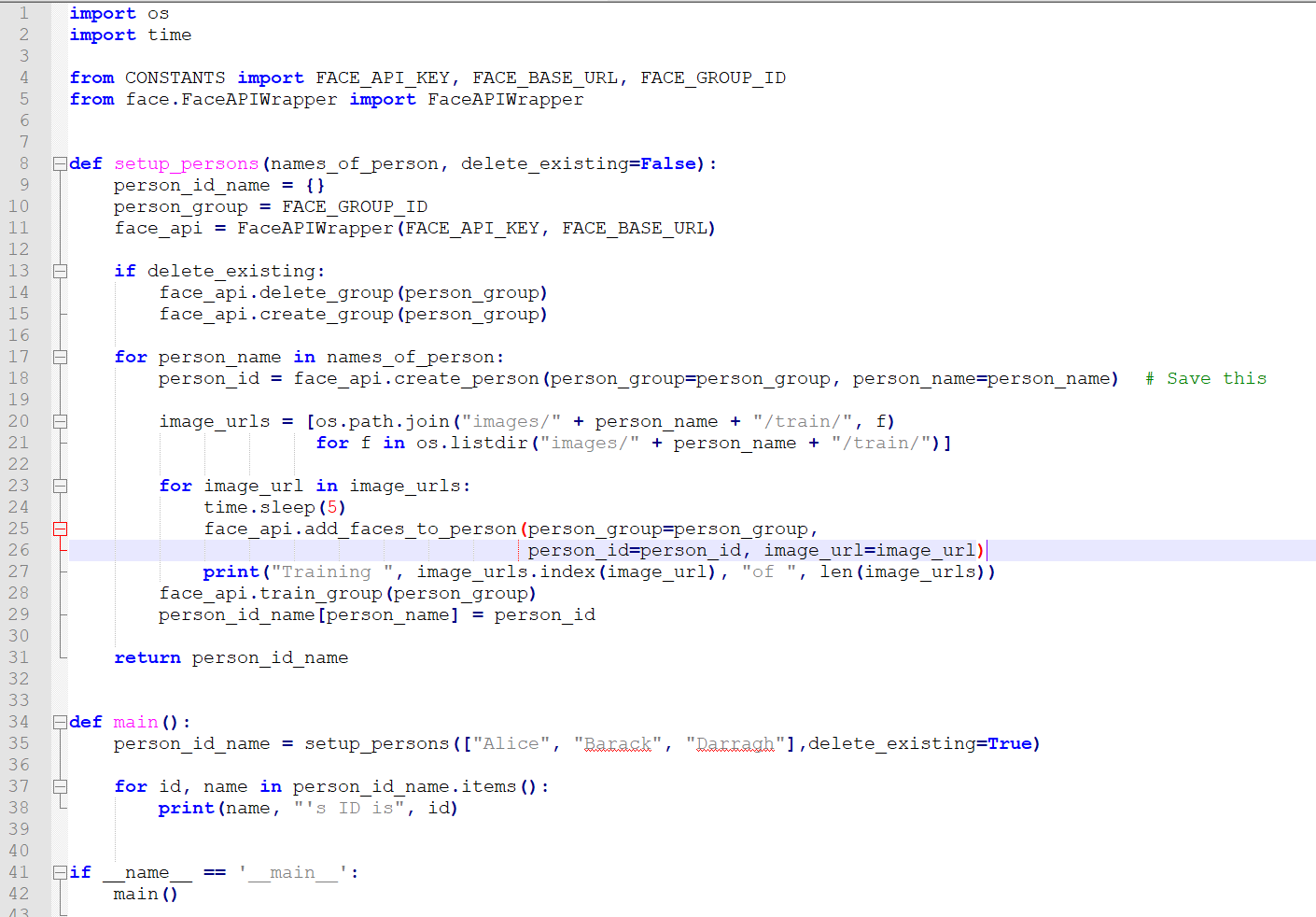
In this file I store all my constants, these can then be referenced in other files.

*Utils.py*

This contains functions that I use in my main file.

*FaceAPIWrapper.py*

This file contains functions that call to Microsoft’s Face API and return the data requested. You can see on line 60 that I’m asking for the Microsoft’s cognitive services to return the attributes, age, gender, smile and emotion.

*train\_setup\_face\_identify.py*

This file trains the faces I want the Face API to recognise. On line 11, we are setting are ‘FACE\_API\_KEY’ and ‘FACE\_API\_ENDPOINT’ to ‘face\_api’ to access MS Face API. We then delete any pre-existing groups. On line 20 we are loading the images in our directory and training our faces. On line 25, we set the ID for the faces to be trained.

*final.py*

This is the main python file. Line 28, OpenCV saves an image. This image is uploaded to an AWS S3 bucket (line 34). This image is also sent to the Face API (line 36). If a face is identified, we print out the ID of the face and the attributes of the face (line 42). After data is returned, we post the results to a Django server running on localhost:8000 (line 49). The code repeats every 15 seconds (line 61).