

# **iDentity-Biometric Face Recognition Time Management System**

---

**Zsolt Toth**

**K00253221**

A Final Year Project submitted in partial fulfilment of the  
requirements of the Technological University of the  
Shannon for the degree of Bachelor of Science (Honours)  
in Software Development.

Supervised By:

Brendan Watson



**TUS**  
Midwest  
Department of  
Information Technology

May, 2023

## Acknowledgements

I want to express my deepest appreciation and gratitude to all those who have supported and guided me throughout the journey of completing my Final Year Project (FYP). This accomplishment would not have been possible without their encouragement, wisdom, and patience.

First and foremost, I would like to extend my sincerest thanks to my FYP supervisor, Mr Brendan Watson, for his unwavering support and guidance throughout this project. His invaluable insights, expertise, and constructive feedback have been instrumental in shaping my work and helping me achieve my goals. I am truly grateful for the time and effort he has invested in mentoring me and the confidence he has shown in my abilities. I would also like to extend my deepest appreciation to my devoted girlfriend, Andreea, for her constant encouragement and support throughout this difficult time. Her forbearance, understanding, and unwavering confidence in me have been fortifying and motivating me. She has been by my side every step, providing emotional support and an ear to listen, for which I am forever indebted.

## Ethical Declaration

I declare that this project and document is wholly my own work except where I have made explicit reference to the work of others. I have read the Department of Information Technology Final Year Project guidelines and relevant institutional regulations, and hereby declare that this document is in line with these requirements.

I have discussed, agreed, and complied with whatever confidentiality or anonymity terms of reference were deemed appropriate by those participating in the research and dealt appropriately with any other ethical matters arising, in line with the TUS Research Ethics Guidelines for Undergraduate and Taught Postgraduate Programmes policy document.

Zsolt Toth

May 1, 2023

## Table of Contents

Acknowledgements .....	ii
Ethical Declaration .....	iii
List of Figures .....	viii
Abstract .....	ix
Chapter 1    Introduction .....	10
1.1    Objective .....	10
1.2    Project Scope .....	11
1.3    Solution Developed .....	11
1.4    Report Structure .....	11
Chapter 2    Research .....	13
2.1    Introduction .....	13
2.2    Why facial recognition is popular .....	13
2.3    Facial recognition in the software industry .....	13
2.3.1    DeepFace handles face recognition on Facebook. ....	13
2.3.2    How does Apple Face ID work, and what is it?.....	14
2.3.3    How does the Google Vision API operate? .....	14
2.4    On-Site Software solutions in the Industry .....	15
2.5    Face Recognition techniques .....	16
2.6    Face detection systems may be categorised into four classes based on their work. 16	
2.7    Facial recognition technology in use .....	18
2.8    Artificial Neural Network .....	19
2.9    Descriptions of well-known neural networks.....	21
2.9.1    Convolutional Neural Network CNN.....	22
2.9.2    Recurrent Neural Network RNN.....	23
Chapter 3    Analysis and Design.....	25
3.1    Application Overview .....	25

3.2	Software Development Process .....	25
3.2.1	Agile .....	25
3.2.2	Behaviour-driven development .....	26
3.2.3	Minimal viable product .....	27
3.3	Data Design .....	28
3.4	Securing application features .....	29
3.5	Use cases .....	31
3.5.1	Upload facial data .....	31
3.5.2	Sign in with facial data .....	31
3.5.3	Sign out with facial data .....	32
3.5.4	Add a new user .....	33
3.5.5	Log in to the user account .....	34
3.5.6	Logout from the user account .....	34
3.5.7	Edit user profile .....	35
3.5.8	Change Password .....	36
3.6	Website Layout .....	37
3.6.1	Locations .....	37
3.6.2	Location details view .....	37
3.6.3	View the location roster's day log .....	38
3.6.4	Location Sign in .....	39
3.6.5	Location Sign off .....	41
3.6.6	Setup user face recognition page .....	42
3.6.7	Edit Profile .....	43
3.6.8	Change Password .....	44
3.7	Tools and Framework Considered .....	44
3.7.1	Jupyter Notebook .....	44
3.7.2	Python .....	46

3.7.3	TensorFlow .....	47
Chapter 4	Implementation .....	49
4.1	Introduction .....	49
4.2	Tools Used.....	49
4.2.1	IDE Support for Programming and Documentation .....	49
4.3	Source Control.....	50
4.4	To install and Run the project .....	50
4.5	Building an OpenCv application to identify user's faces .....	50
4.5.1	How to use Open CV .....	50
4.5.2	How to find a face in an image using OpenCv: .....	50
4.5.3	How to train a facial recognition model using OpenCV:.....	53
4.5.4	How to recognise a face in an image using a trained model in OpenCV..	54
4.5.5	Support for OpenCv processing web inputs.....	55
4.6	Building Identity web project.....	57
4.6.1	Using Django .....	57
4.6.2	The Models .....	58
4.6.3	The Views .....	59
4.6.4	The URLs.....	63
4.6.5	The Admin .....	65
4.6.6	The Forms .....	65
4.6.7	Utilities.....	66
4.6.8	Constants and File Paths .....	67
4.6.9	Face Detection.....	67
4.7	Client Side Functionality .....	67
4.7.1	Use Case #1: User Facial Recognition Setup.....	67
4.7.2	Use Cases #2-3: Location Roaster Sign in/out.....	68
4.7.3	Refactoring Javascript.....	68

4.7.4	Incorporating Javascript into the web app .....	69
4.8	Bulding and deploying the Website .....	69
Chapter 5	Testing and Results .....	70
5.1	Introduction .....	70
5.2	Unit Testing .....	70
5.2.1	What is Unit Testing, and why is important .....	70
5.2.2	Testing Framework in Django .....	70
5.3	Unit Testing Done .....	71
5.4	Test-Driven Development .....	71
5.5	Result Analysis:.....	73
5.6	Discussion of Findings .....	73
Chapter 6	Conclusion: .....	74
6.1	Continued development.....	74
6.2	Machine learning approaches .....	74
6.3	Tracking continuing developments .....	75
6.4	Behaviour-Driven Development .....	76
Chapter 7	Bibliography.....	77

## List of Figures

Figure 1 A multi-layer NN's theoretical structure (GeeksforGeeks, 2021).....	20
Figure 2: Operation of the filter unit used by CNN (Dertat, 2017) .....	23
Figure 3: Recurrent Neural Networks (Olah, 2015).....	24
Figure 4. An unrolled recurrent neural network (Olah, 2015). .....	24
Figure 5: Agile .....	26
Figure 6: Behaviour-driven development Cycle .....	27
Figure 7: MVP .....	27
Figure 8: Gantt Diagram .....	28
Figure 9: Class Diagram.....	29
Figure 10 Sign-in example .....	30
Figure 11 Feature permissions .....	30
Figure 12 Locations list view .....	37
Figure 13 Security notification.....	37
Figure 14 Limerick location view .....	38
Figure 15 Location roster day view .....	38
Figure 16 Location Sign In part 1 .....	39
Figure 17 Location Sign In part 2 .....	40
Figure 18 Location Sign Out part 1. ....	41
Figure 19 Location Sign Out part 2 .....	42
Figure 20 Setting Up Facial recognition. ....	43
Figure 21: Edit User Profile .....	43
Figure 22: change-password.html .....	44
Figure 23: Generator versus discriminator.....	75



## Abstract

Managing security and roster logs in workplaces continues to be a challenging and costly endeavour. The complexities are heightened in environments that require frequent logging of employee attendance at multiple entry and exit points. Consequently, security measures can become lax, especially during peak hours with a high volume of workers. Identity offers an innovative solution by combining a web-based application and a security station equipped with a live camera feed to automate entry and exit of authorized personnel. By leveraging artificial intelligence and facial recognition technology, Identity streamlines security access and roster log management, enhancing efficiency and reducing costs.

## Chapter 1 Introduction

Like other biometrics systems, facial recognition technology analyses and compares distinctive characteristics to identify or authenticate users. Facial recognition software may identify faces in photos, measure their characteristics, and then compare them to templates already stored in a database. Biometric face recognition is used extensively in various contexts, including applications like Facebook or smartphones, workplaces, airports, colleges, and banks. The wide range of potential uses for face-scanning biometric technology demonstrates its versatility (FindBiometrics, 2022).

### 1.1 Objective

This project aims to develop a web-based time management and user identification system that leverages facial recognition technology. The technology employs cameras to take a picture of the user's face and then compare it to a previously registered image in the system to verify the user's identity. The system grants access when the user's identification has been verified and logs the user's entry and exit times. This method may be utilised in various settings, such as businesses, institutions of higher learning, and healthcare facilities, to enhance security and time management.

**This solution's objectives are:**

- To provide a system that uses biometric facial recognition technology to simplify clocks in and out in a workplace.
- A layer of security for the workplace or premises by making it possible to determine whether an unauthorised person is present in a specific location.
- To improve and reduce administrative overload by tracking employees' working hours, i.e., we can see when an employee has finished on a specific day.
- Prevent false claims of hours worked, such as overtime, instead of paying employees for falsely reported hours worked.

### **The academic objectives:**

- To learn about machine learning
- To understand how biometric facial recognition works.
- To fully build and develop a professional web application with an inbuilt facial recognition system.
- And gain experience with academic research

### **1.2 Project Scope**

The project investigates Facial recognition technologies currently employed. A suitable proof-of-concept solution was designed, developed and tested on a small number of users. The delivered software has been limited to the core web application functionality. Controlling automatic door locks with software is outside this project's scope.

### **1.3 Solution Developed**

The developed solution, Identity, comprises a database-driven web application. The web application is written with Python/Django on the server side and some JavaScript. Facial recognition is achieved with the openCV framework. The source code for the web application can be viewed at <https://github.com/zsolt821201/identity-ml-fyp-py>

### **1.4 Report Structure**

This report comprises five chapters: Research, Analysis and Design, Implementation, Testing and Results and Conclusions.

- The research discusses the significance of facial recognition in the software industry, including its popularity and various face recognition techniques and detection systems. It also examines the use of facial recognition technology in the industry and describes artificial neural networks, such as the Convolutional Neural Network and Recurrent Neural Network.
- The analysis and design section overviews the application and software development process, discussing data design and securing application features. It presents use cases and website layouts and describes the tools and frameworks considered for the project.

- The implementation section describes the tools and source control used for the project, provides instructions on how to install and run the project, and details the process of building an OpenCV application to identify users' faces and the Identity web project using Django. It explains client-side functionality and how JavaScript was incorporated into the web app and discusses building and deploying the website
- The testing and results section provides an overview of unit testing and the testing framework used in Django. It presents the unit testing done and the results of test-driven development, analyzing the findings and results of the testing.
- Finally, the conclusion discusses continued development and machine learning approaches, provides insights on tracking continuing developments and Behavior-Driven Development, and critiques the software and its development process.

## Chapter 2 Research

### 2.1 Introduction

This chapter explores how various large software companies have tackled the challenge of facial recognition technology and the underlying mechanisms that power it. The author examines these companies' various techniques and methodologies and explains their solutions. The primary objective of this chapter is to equip the reader with comprehensive knowledge of contemporary facial recognition software and the cutting-edge technologies that drive its operation.

### 2.2 Why facial recognition is popular

In recent years, facial recognition has become more popular because it can be used for many things, like security, marketing, and social media. The technology can identify people and their traits, and thanks to improvements in artificial intelligence and machine learning, it is much more accurate than it used to be. But some concerns have been raised about how facial recognition affects ethics and privacy, especially regarding surveillance and how its algorithms might be biased. The use of facial recognition technology has been discussed and regulated. Some countries and cities have banned or limited the use of this technology. Still, facial recognition is likely an important and widespread technology in many areas, and more research is needed to ensure it is used responsibly and ethically.

### 2.3 Facial recognition in the software industry

#### 2.3.1 DeepFace handles face recognition on Facebook.

A Facebook research team created the deep learning facial recognition algorithm known as DeepFace. In digital photographs, it can identify human faces. The program employs a nine-layer neural network trained using four million photos uploaded on Facebook and has around 120 million connection weights (DeepFace, 2022). With 97% accuracy, DeepFace algorithms can recognise faces nearly as well as a person in an identical situation (DeepFace, 2022). The significant difference between human and machine face recognition is that a person can only maintain a few hundred or thousand faces in their memory. Therefore, they can only recall a limited number of faces. DeepFace, on the other hand, identifies all 1.4 billion users, and this number is continually expanding. The DeepFace System comprises four parts: a 2D alignment module, a 3D alignment module, a frontalization module, and a neural network module. When a picture of a face is sent through them in order, a 4096-dimensional feature vector is made that represents the face.

After that, the feature vector can be used for a wide range of tasks. For instance, a face may be recognised by comparing its feature vector to a list of existing faces and determining which face has the highest resemblance to the other faces' feature vectors (DeepFace, 2022). The fact that the DeepFace facial recognition software learns from each new photo it recognises and adds what it discovers to the database to improve a person's facial profile is one of the program's most advantageous features.

### 2.3.2 How does Apple Face ID work, and what is it?

Before, a simple number password protected Apple products, but starting with the iPhone 5S in 2013, they were getting read off in favour of fingerprint readers in new phones. After having fun with fingerprint readers, Apple decided that if they were going to take unlocking to a new level, they needed to make it safer and faster. In 2017, they released the iPhone X with a unique face recognition feature called Face ID. Face ID creates a comprehensive 3D map of the user's face using a "TrueDepth camera system," comprising an array of sensors, cameras, and a dot projector housed in the notch at the top of the phone display (Pocket-lint, 2018). The technology does a safe authentication check when you peek at your phone, and if it recognises you, you may unlock your phone or quickly and easily authorise a payment (Pocket-lint, 2018). The TrueDepth camera system, neural networks, and

Bionic chips are just a few hardware elements that face ID depends on. Face ID can adapt to changes in a person's appearance, such as the application of cosmetic makeup or the development of facial hair. Suppose there is a more significant change in present appearances, such as shaving one's beard. In that case, Face ID will first verify a user's identification using the user password before updating the user's face data (Pocket-lint, 2018).

### 2.3.3 How does the Google Vision API operate?

One of Google's most intriguing and least-used capabilities is image-based search. The typical method for finding photographs on the Internet is to enter a phrase (keyword) of interest into the Google search bar, then choose the "Images" tab to see the results. In an image-based search, the image itself is the "keyword" or starting point, and the search results are images. Since this is how it works, this function is often called "reverse" search. Google's system for recognising images looks at pictures on the Internet from many angles. Google Vision API can determine a picture based on its shapes, colours,

backgrounds, and locations. It can tell if the image is of a person, an animal, an object, or a product.

With the help of pre-trained models on big-picture datasets, the Google Cloud Vision API employs machine learning to recognise photos. It divides the images into hundreds of categories to identify items, locations, and faces. The findings are then produced with a confidence value (Vision?, What is Google Cloud, n.d.).

## 2.4 On-Site Software solutions in the industry

Of course, in addition to cloud-based methods, on-premises solutions provide facial recognition by default. However, they have much more modest capabilities than their cloud counterparts and are usually not controlled by artificial intelligence. They are mostly not standalone software; they are frameworks implemented in different languages, so their use requires a certain programming skill. However, there are examples of such a framework containing an autonomous package. Among them is commercial software that requires a one-time payment, but open-source solutions that can be used for free are becoming increasingly popular. In this section, the two most common ways will be presented.

### 2.4.1 OpenCV

OpenCV is a BSD-licensed open-source, cross-platform package that is primarily a real transaction for real-time image processing and is today the most popular solution in the field of image analysis, especially since it also has native Nvidia CUDA support. Originally It was written in the C language by Intel engineers. Still, it is also available in C++ and Python, and it has been extended with more language support using shell classes. Different wrap (Goyal, et al., 2017). In the field of face recognition, for detecting landmarks (Facemark API) and is suitable for solving related problems (e.g., finding faces, drawing landmarks), but it also has many other functions ( e.g., use own Face and Fischer face). The solution uses a Haar-Cascade classifier for face recognition, a machine learning-based method that uses cascading layers (Viola & M.Jones, 2003).

### 2.4.2 Dlib

Dlib is a low-level cross-platform open-source software written primarily in C++ but also supported by Python, part of the general artificial intelligence-related libraries that also have a set of tools for face recognition (King, 2009). Among other things, it provides the

ability to search for face highlights and is also suitable for different implementations (such as face clustering). Despite the free Boost software license, many documentation and sample libraries provide step-by-step guidance from translating to performing development tasks. The face recognition module in the package contains a solution that only supports face-to-face recognition based on the so-called HOG (histogram of oriented gradients) method, which uses a distribution of image elements to identify different classes. More recently, methods based on neural networks are also part of the package.

We may conclude from the above example that facial recognition systems have enormous potential commercially and security-wise and how these systems have grown to be incorporated into our daily lives, whether shopping, online banking, or finding someone in the picture. We can use cloud base systems, on-site open-source, or commercial frameworks to develop Face recognition systems.

## 2.5 Face Recognition techniques

This subject has a broad reach and extensive literature, and an entire series of expert publications deal with its understanding. Therefore, it is not surprising that although employing the same set of tools, the problem hides a variety of specialities: Face recognition is the study of figuring out the formal and shape requirements for recognising a person's face and separating it from the background (Khan, 2018). Face-based identification, a subset of this, employs the former's advances to determine how similar or unlike two faces are and if they can both belong to the same person. Although the separation of these two notions is not strictly scientific in nature, the literature utilises them interchangeably to some degree, and they will be made separate throughout the remainder of the thesis.

## 2.6 Face detection systems

Facial detection systems may be categorised into four classes based on their work.

- Knowledge-based methods
- Feature-invariant method
- Template matching method
- Appearance-based method



### **Knowledge-based methods**

These approaches are based on shared human perceptions of a face. For example, in a photograph of a face, the minor intensities will be around the eyes, while the biggest would be near the nose. They attempt to abstract our understanding of faces into a set of rules. Some basic laws are straightforward to deduce. A face, for instance, typically features two symmetrical eyes, and the region around the eyes is darker than the cheeks. The space between the eyes or the contrast in colour intensity between the lower zone and the eye area is a facial characteristic (Solanki, 2016).

### **Feature-invariant method**

Such techniques involve first identifying the distinguishing features of the face (eyes, mouth, and nose), after which the face is recognised by grouping these features. Characteristics of a face should remain the same regardless of the angle or location, according to feature-invariant techniques. The mouth, nose, eyes, cheekbones, chin, lips, forehead, ears, upper boundaries of the eye sockets, areas around the jawline, the sides of the mouth, and the positioning of the nose and eyes are some of the identifying features of the face that are used in facial identification. The nose's size, the jaw's angle, and the space between the eyes (Solanki, 2016).

### **Template matching method**

These methods are based on looking for a pre-made facial pattern in a photograph. This may be an edged template. The problem with such techniques is that the template we intend to utilise must be extremely representative because different ethnic groups have distinct traits. Furthermore, the pre-processing must be very good, as such systems are sensitive to lighting conditions. Moreover, these algorithms are sluggish and cannot function in real-time since you must experiment with multiple orientations and scalings across the image.

### **Appearance-based methods**

Based on a collection of pre-made teaching samples, a “model” is built, and portions matching this model are sought after in the image. Most of the time, appearance-based

algorithms use statistical analysis and machine learning to figure out what's important in a photo of a face (Solanki, 2016).

- Eigenface based method – PCA Algorithm:

According to (Çarıkçı, 2012), the Eigenfaces method involves recognising a face's typical characteristics and representing those features as a linear combination of "eigenfaces" discovered during the feature extraction process. The main components of the faces in the training set are calculated, and to conduct recognition, the face is projected onto the space formed by the eigenfaces. A comparison is then made using the Euclidean distance between the eigenvectors of the eigenfaces and the eigenface of the image in question. If the distance is narrow enough, the individual is recognised. However, if the distance is too great, the image is considered to belong to an individual for whom the system needs to be trained (Çarıkçı, 2012).

- Distribution-based Methods – LDA Algorithm:

Another method of dimensionality reduction is Fisher's Discriminant Analysis or LDA. Since LDA optimises the between-class scattering matrix measure while minimising the within-class scatter matrix measure, it is an example of a class-specific approach. It is more trustworthy for classification (Solanki, 2016). For both face identification and verification, LDA-based algorithms beat PCA. Fisher's faces are one of the most extensively utilised methods for face recognition. It is dependent on the way of appearance. Fisher invented linear/fisher discriminant analysis for face identification in 1930, which yielded successful results in the face recognition process (Solanki, 2016).

## 2.7 Facial recognition technology in use

Artificial neural networks are one of the most widely utilised machine learning methods. Let's discuss computer vision, language processing, and self-driving cars to illustrate how they can tackle nearly any issue. It is frequently used, for instance, to analyse deep learning, complicated relationships, or interactions with plenty of data. When we hear the words "neural network," neurons in the brain are the first thing that comes to mind. The brain's ability to make decisions comes from these units and the networks they form. One of the most significant differences between a person and a robot or computer program is that a person's brain can recognise, evaluate, and learn from what it sees.

## 2.8 Artificial Neural Network

The neurons inspire the names of artificial neural networks in the brains of more advanced biological creatures (Beresford & Agatonovic-Kustrin, 2000). A biological neuron may receive and send electrical messages to other neurons through its synapse, allowing impulse transmission (Jain, et al., 1996). Neurons are all connected in a much more complex and dense way than phone networks. Each neuron is linked between  $10^3 - 10^4$  with other neurons; the human brain has  $10^{14}$  to  $10^{15}$ . (Jain, et al., 1996). No matter how big they are overall, neurons transfer information in smaller chunks, which is an essential characteristic from the perspective of ANNs. This led scientists to conclude that neurons do not directly convey information but via their interactions (Jain, et al., 1996). The development of ANNs was based on these discoveries. The initial neurons' mathematical models are simple: for a given number of weighted input signals, one is returned if the sum of the inputs exceeds a specified  $u$  value; otherwise, 0 is returned. Jain et al. summarised the action of early neurons as follows: where is a unit jump function in 0,  $x_j$  is the  $j$ th input,  $w_j$  is the synaptic weight associated with it, and  $u$  is the limit to be reached (Jain, et al., 1996).

$$y = 0\{\sum_{j=0}^n w_j x_j - u\}$$

In practice, the threshold value  $u$  is often written as  $w_0$  and is called a constant weight of value  $x_0$  (McCulloch & Pitts, 2014). He also says that the different weights mean different things. From the point of view of neuron activity, a positive weight makes the synapse more active, while a negative weight makes it less active. In today's world, the number of inputs in the original may be very different from the amount in the model (but none of its element numbers can be 0) (Kristof, 2002).

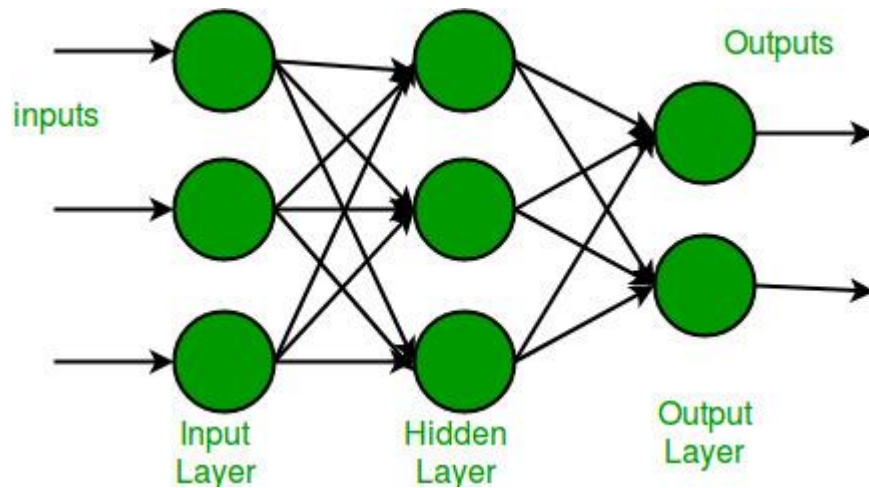


Figure 1 A multi-layer NN's theoretical structure (GeeksforGeeks, 2021).

It has been shown that this model can do general-purpose computation (Jain, et al., 1996)s, but since it is built on several simplifications that do not correspond to how actual neurons work, it has undergone multiple generalisations to reach its present state. Alternative activation functions are used in place of the threshold function. These processes and the threshold function determine how the given neuron will react to the input. One of the most widely utilised functions in use today is the sigmoid activation function (Shanu, n.d.), which has the following formula:

$$y = \frac{1}{1 + e^{-(a-0)b}}$$

Where  $a$  represents the activation and  $b$  determines the curve's shape. One of the essential things about neural networks is that they can learn because they are made up of many layers, and neurons can talk to each other across the layers. Generally speaking, but not always, these networks have an input layer, one output layer, and layers in between. Every neuron in the neural network's input layer, called the "input layer," is connected to the "intermediate layer" so that the data it receives can be sent further. Each neuron in the neural network's input layer is linked to the intermediate layer, allowing it to continue transmitting the data it has received as input (Kristof, 2002). Weighted synapses link the neurons of the input layer to those of the inner layers. The significant features from the perspective of the issue are used to determine how much weight to initially assign to each job, which is pretty task specific. While the neurons in the input layer have fixed weights, the consequences in the intermediate layer are dynamic, demonstrating the potential to learn. The teaching samples that are sent to the network influence these changes. The rules seem to be learned from the supplied representative sample rather than being

explicitly provided by the researchers; in other words, neural network learning is simply the act of dynamically allocating weights to intermediate components (Jain, et al., 1996). In the early stages of learning, neurons provide a random response to the issue, then compare the outcome to the teaching sample and adjust the internal weighting accordingly. Despite its outstanding findings, the neural in the case of networks, the output provided after the process is not guaranteed to be the ideal result. Throughout learning, the network adapts to a particular local minimum value via serial mutations. Most training networks use supervised, unsupervised, or hybrid training methods. The quality of the information given to the network varies depending on which training method is used. In the first approach, the network aims to adjust the weighting to produce outputs near the set of predicted results (the sample with the most essential information) as feasible (Jain, et al., 1996). In a variant known as reinforcement learning, the network doesn't receive the anticipated result but instead works on a value obtained from the discrepancy between the actual and expected outcomes. Instead of producing outputs for various inputs, networks in unsupervised learning aim to reveal the basic patterns present in the training sample and their interactions. In the case of hybrid learning, half of the weights are mapped manually, while the other half is created via directed learning (Jain, et al., 1996). For neural networks to be used in the real world, they need to be well-trained and given data with a similar structure they have never seen before. They can then use the knowledge they gained during training to complete the task.

## 2.9 Descriptions of well-known neural networks

In addition to the many theoretical models, researchers may choose from a wide range of real-world applications. The present explanation focuses on elements that often come to the forefront in face recognition research. Since these application techniques differ, they may also be investigated according to qualities (e.g., topology, layer depth, neuron composition, information propagation direction). The facial recognition scientific literature is hence also referred to as literature.

Two main kinds of configurations often appear in the literature: feedforward and recurrent networks, depending on the direction in which information might travel between the network's neurons. The neuronal architecture of the earlier networks results in a non-circular graph, which causes the outputs to flow from the input directly to the output direction. In contrast, the architecture of the later networks contains neurons with multi-

directional connections, which causes their architecture to remain circular. As stated before, neurons are grouped in layers in both scenarios and produce outputs dependent on the inputs provided by layers that came before them. As they travel down the whole length of the mesh, these outputs are constantly compared to the desired result. The outcome, often referred to as the loss, is the difference between the predicted and actual results. The error rate generated during the performance test travels back to the earlier layers through a process called backpropagation. It helps to dynamically modify the weights representing the importance of the different input neurons (Tang & Fishwick, 1993). Thus, by continually distributing weights in the solution, such nets essentially teach themselves which parts are involved in the problem and in what proportion. Topology, or the connections between the neurons that make up a network, is a characteristic of the network that significantly influences how learning generally proceeds. The most obvious illustration of this is the amount (depth) of the layers that make up the network; deep neural networks are networks with many layers, and their typical learning method is deep learning, although exact numbers are sometimes not explicitly provided (Yi, et al., 2015).

### 2.9.1 Convolutional Neural Network CNN

These networks are distinguished from others because image processing is their major function. They are also capable of comprehending a variety of information types (video, audio, etc.). Our goal is to classify things based on the information in the picture, which is accomplished by transferring the data of an image (at the pixel level) via the network in a typical application.

A convolutional neural network (CNN) scans the details of incoming data rather than interpreting it. Using a layer of 1,000,000 ( $1,000 \times 1,000$ ) neurons for pixel-level interpretation is not the most efficient when 1,000 pixels  $\times$  1,000 pixels image is the input. Instead, the data is sent over the network in detail using a 100x100 pixel filter (Dertat, 2017).

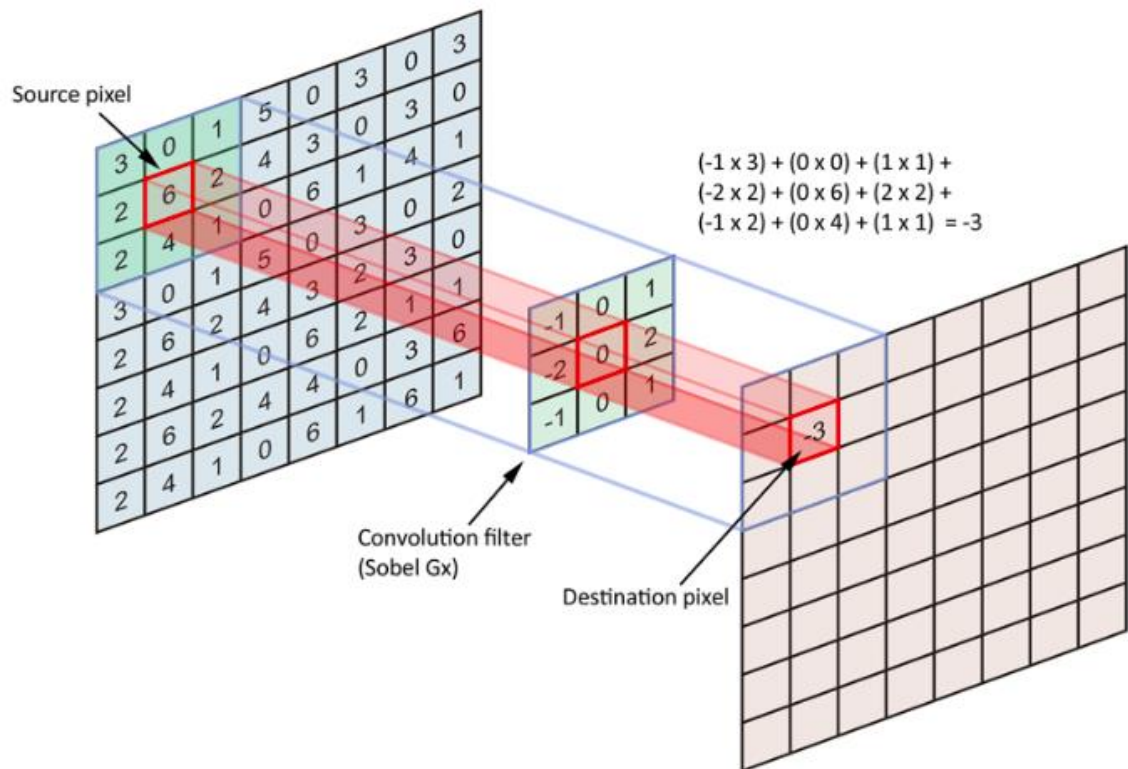


Figure 2: Operation of the filter unit used by CNN (Dertat, 2017)

The two primary components of CNNs are feature discovery and categorisation. During the feature detection process, averaging (convolution), merging, and compression operations are carried out on the image's component units. For instance, if a photo of a dog is submitted into the system, feature recognition will compress the pixels to produce the image's recognisable forms (ears, mouth, legs). The properly trained and parameterised classification can look at all the features based on the ones that have already been found. It can then interpret and classify the whole scene in the image. To train sophisticated neural networks, pre-tagged data sets are required. The most challenging job is to identify system-specific hyperparameters, "structure-defining parameters": construct a neural network, i.e. how many layers are used, how many neurons should be put in the layer, activation function defined in the most layers, size and structure of the filter utilised (Dertat, 2017).

### 2.9.2 Recurrent Neural Network RNN

A feedforward neural network has no memory, so it cannot recall past events. Of course, the exception to this rule is the training process, where neuron weight values change. In a recurrent neural network (RNN), information loops so that neurons can think about how they responded to inputs from before when deciding what to send out.

Let's say that the word "neuron" is fed into a feedforward network, and the system goes through it character by character. When figuring out the output for the 'r' input, characters that have already been used cannot be considered. To determine a complicated object's meaning, you must consider how its parts work together.

The problem is solved via recurrent neural networks. They are recursive networks where data may be stored indefinitely (Olah, 2015).

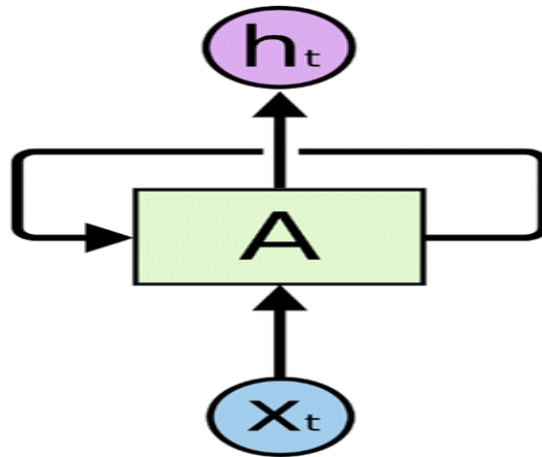


Figure 3: Recurrent Neural Networks (Olah, 2015).

As seen above, a section of the neural network labelled A processes the input  $x_t$  and returns the result  $h_t$ . The network's information may be transmitted from node to node through a loop.

Recurrent neural networks are imbued with an aura of mysticism thanks to the presence of these loops. In accordance with Olah (2015), it is possible to observe that recurrent neural networks (RNNs) share many similarities with traditional neural networks. RNNs consist of multiple identical networks, with each network transmitting information to the following network. To gain a better understanding of RNNs, it can be useful to imagine what would happen if the loop in the network structure was unfolded (Olah, 2015).

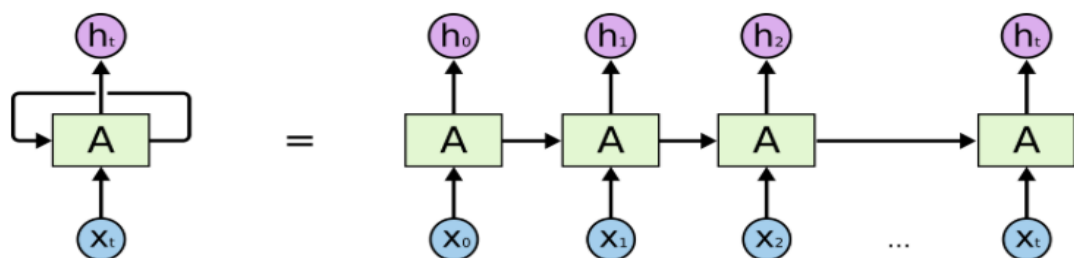


Figure 4. An unrolled recurrent neural network (Olah, 2015).



## Chapter 3 Analysis and Design

### 3.1 Application Overview

The application uses a camera to take a picture of the user's face and then compares it with an image previously registered in the system. This process verifies the user's identity before allowing them access to a specific location, such as a building or a room. Once the user's identification has been verified, the system grants access and logs the user's entry and exit times for that location. This allows for accurate tracking of user movement and maintains security by ensuring that only authorised personnel are granted access to restricted areas.

### 3.2 Software Development Process

Identity was developed using a combination of Agile and Behaviour-driven development methodologies. As a proof-of-concept project for academia, a focus was kept on developing a working Minimal Viable Product (MVP) rather than an unfinished all-features product.

#### 3.2.1 Agile

Agile methodology revolutionizes project management with a focus on an iterative and incremental approach, emphasizing adaptability, collaboration, and continuous improvement. By breaking down projects into smaller goals, Agile empowers developers to respond quickly to challenges and changes, ensuring the final product aligns with stakeholders' needs and expectations. Focusing on regular feedback, cross-functional teamwork, and open communication, Agile can deliver high-quality, innovative solutions that keep pace with the ever-changing requirements of a project.

Key aspects of Agile methodology suited to this project are:

- An iterative and incremental approach to project management
- Emphasis on adaptability and flexibility
- Breaking down projects into smaller, manageable tasks
- Setting short-term goals and adjusting as needed
- Incorporating regular feedback and open communication

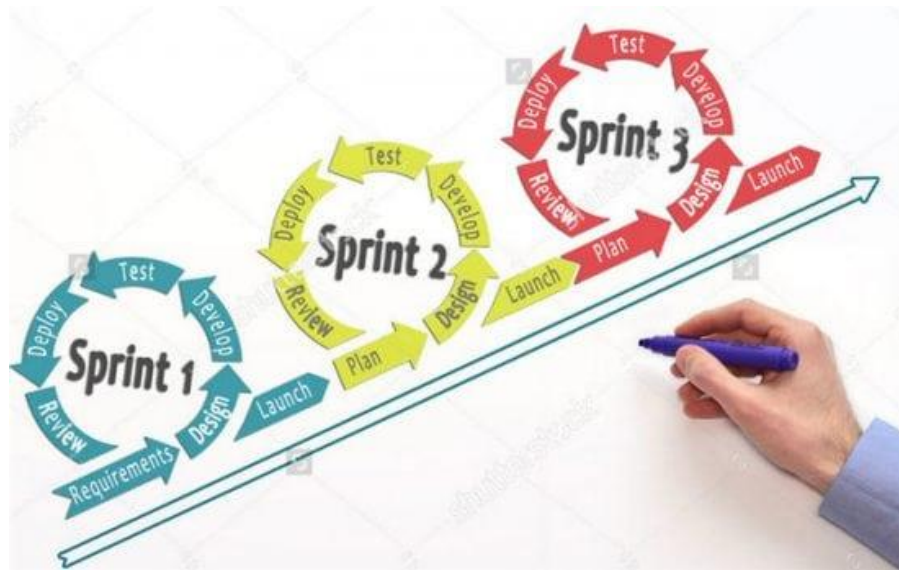


Figure 5: Agile

### 3.2.2 Behaviour-driven development

Behaviour-driven development (BDD) builds upon test-driven development (TDD) to enhance communication and collaboration between technical and non-technical stakeholders. While TDD focuses on writing tests before implementing code, BDD takes it further by emphasizing clear, concise language to define requirements and acceptance criteria, ensuring a shared understanding of software behaviour across the team (Wynne, 2017).

Behaviour-driven development (BDD) is a methodology that bridges the gap between technical and non-technical stakeholders by emphasizing collaboration, communication, and a shared understanding of the software's expected behaviour. BDD encourages using clear, concise, and easily understood language to define requirements and acceptance criteria, facilitating a common understanding among stakeholders. This approach promotes the development of high-quality software that meets user needs while minimizing misunderstandings, misinterpretations, and costly rework.

Key aspects of Behaviour-driven development:

- Collaboration between technical and non-technical stakeholders
- Emphasis on a shared understanding of software behaviour
- A clear, concise, and easy-to-understand language for requirements.
- User story-driven approach to defining features.
- Specification by example to ensure accurate implementation.

- Test-driven development to validate expected behaviour.
- Continuous feedback and adjustment to meet user needs and expectations.



Figure 6: Behaviour-driven development Cycle

### 3.2.3 Minimal viable product

A minimal viable product (MVP) is a simplified working version of a product for release with just enough features to satisfy early adopters and gather valuable user feedback.

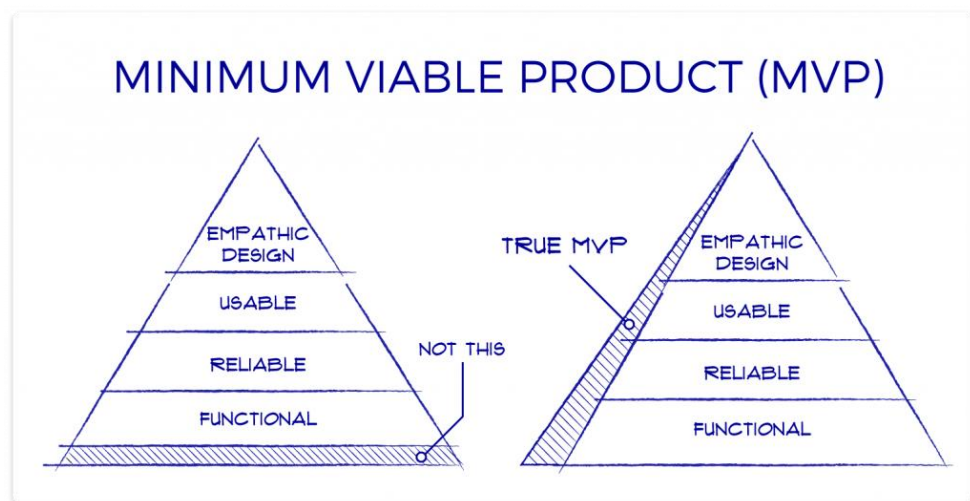


Figure 7: MVP

Gantt diagrams can be useful for keeping track of tasks and focusing on critical tasks for an MVP.

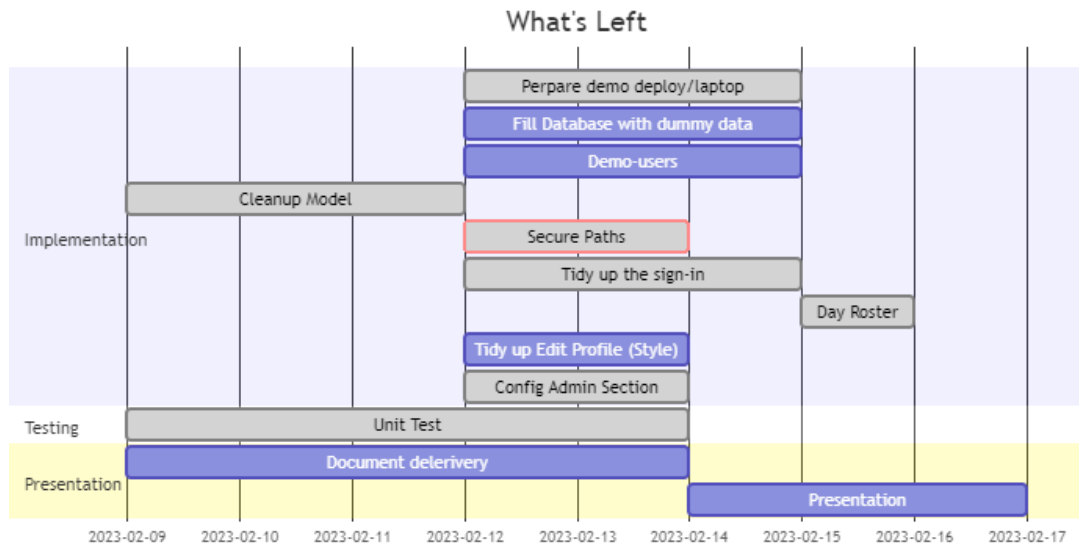


Figure 8: Gantt Diagram

### 3.3 Data Design

The application requires a system for storing important data, which includes information about users, locations, and entry/exit logs, referred to as a “roster”. The application captures facial data to store user data and records users’ permissions to access various locations. Additionally, the application stores information about who is authorised to access a particular location. This helps the application to ensure that only authorised personnel can enter restricted areas, thereby maintaining a secure and safe environment. With this system in place, the application can accurately track users’ movement, maintain security, and ensure that only authorised individuals can access restricted areas.

Admin Group: members of the admin group have /inherit admin privileges. The system must have at least one admin user.

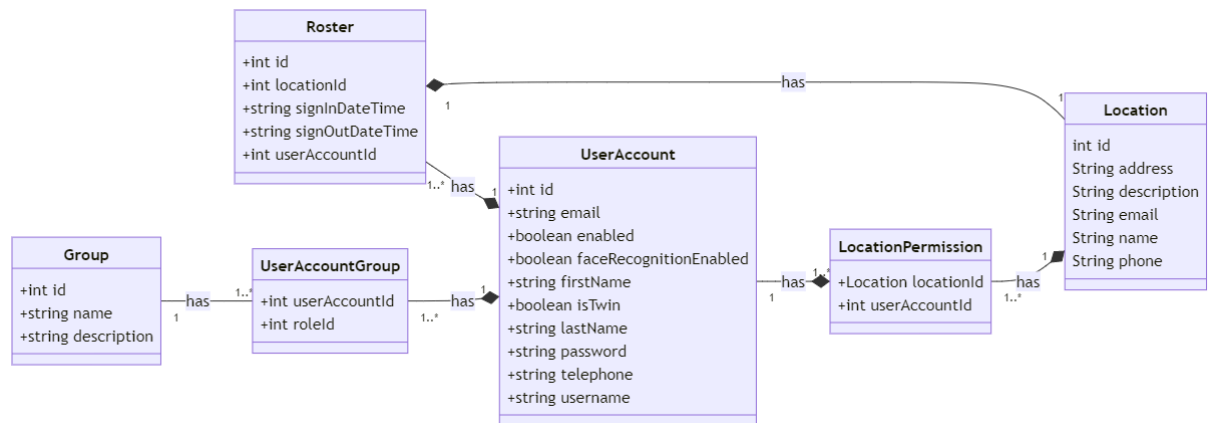


Figure 9: Class Diagram

### 3.4 Securing application features

Only authorised users are granted access to the sign-in and sign-off features to ensure proper security measures. For instance, the organisation's manager can access the machine and activate the sign-in feature for the day. This is usually done in the morning and involves turning on the machine located outside the building's entrance. Once activated, the machine can automatically sign in users with proper authorisation, ensuring that only individuals authorised to access the building are granted entry. Limiting access to these features is important to avoid potential security breaches and maintain control over who has access to the building. This process helps maintain a safe and secure environment for employees and visitors.

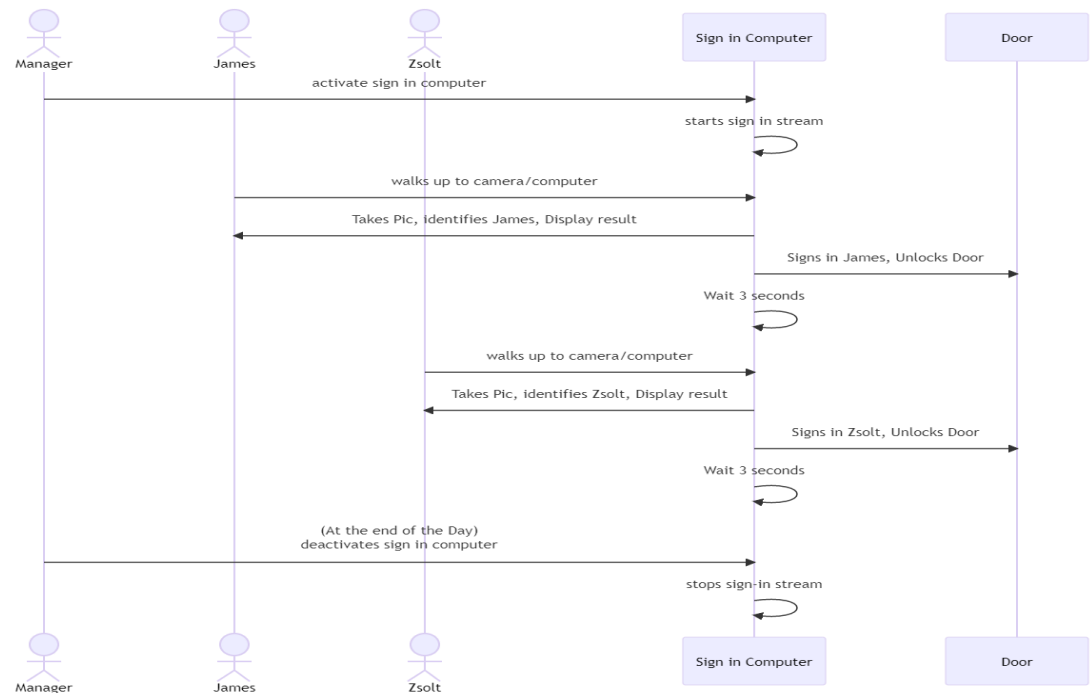


Figure 10 Sign-in example

A location manager or other designated users are granted permission in the application to start sign-in / sign-off features.

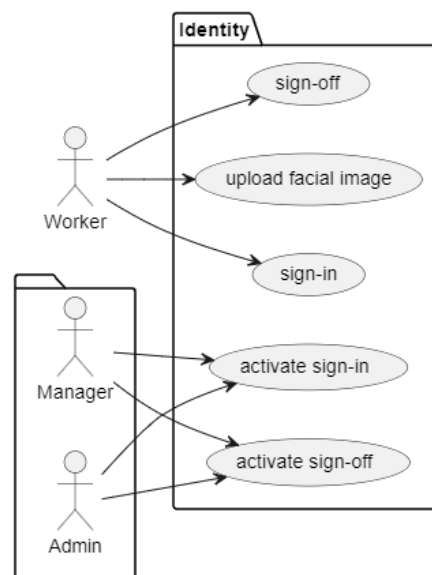


Figure 11 Feature permissions

### 3.5 Use cases.

#### 3.5.1 Upload facial data.

##### **Brief Description:**

This use case describes capturing and storing a user's unique facial features or biometric information, such as the distance between the eyes or the shape of the jawline.

##### **Trigger Event:**

The user requests the system to set up their facial recognition profile.

##### **Main Success Scenario:**

1. The system renders the setup screen and asks the user to look at the camera.
2. The system takes multiple pictures of the user's face.
3. The system processes the pictures and stores the facial recognition profile.
4. The system compiles and stores the facial recognition profile for all users in the trainer.yml file.
5. The system displays a success message.

##### **Alternative Flows:**

1. The application cannot find a face in the video picture stream.
2. If the application doesn't detect a face in 900 still video frames, the user is alerted to an error.

##### **Pre-Conditions:**

1. IsFaceLogin enabled is true for the user.
2. The user is logged in.
3. The user is sitting in front of the camera.

##### **Post-Conditions:**

Facial Data trained and stored.

#### 3.5.2 Sign in with facial data.

##### **Brief Description:**

This use case describes how a user logs in with their facial data.

**Trigger Event:**

The user walks up to sign in computer and looks at the camera.

**Main Success Scenario:**

1. The system renders the login screen and asks the user to look at the camera.
2. The security station client streams pictures of the user's face to the server.
3. The server processes the images, compares them to the facial recognition profile and returns the result to the client station.
4. The security station client displays a success message if the user is recognised.

**Alternative Flows:**

1. The application cannot find a face in the video picture stream.
2. There are multiple faces in the video stream; the user is alerted to an error.
3. If the user has already logged in, the system will alert the user they are already on the roster.

**Pre-Condition:**

1. IsFaceLogin enabled is true for the user.
2. The user must have a facial recognition profile stored in the system.
3. The user must have permission to access that specific location.

**Post-Condition:**

The user is logged in and can access the building.

### 3.5.3 Sign out with facial data.

**Brief Description:**

The use case describes how a person signs out of a building using their face.

**Trigger Event:**

The user walks up to sign out security station and looks at the camera.

**Main Success Scenario:**

1. The system renders the logout screen and asks the user to look at the camera.
2. The client security station streams pictures of the user's face to the server



3. The server processes the images, compares them to the facial recognition profile and returns the result to the client station.
4. The security station client displays a success message if the user is recognised.
5. The security station unlocks the door.

**Alternative Flows:**

1. The programme is unable to locate a face in the video image stream.
2. The user is notified of an error if the video feed contains numerous faces.
3. Users who cannot sign out with their face can manually sign out from the application using their username and password.

**Pre-Conditions:**

1. IsFaceLogin enabled is true for the user.
2. The user must have a facial recognition profile stored in the system.
3. The user must have permission to access that specific location.

**Post-Conditions:**

The user is logged out and sign out time is recorded.

**3.5.4 Add a new user.**

**Brief Description:**

The use case describes how a user is added to the system by an admin.

**Trigger Event:**

The administrator logs in to the admin panel and creates a new user account.

**Main Success Scenario:**

1. The system renders the add user screen and asks the admin to enter the user's first name, last name, email address and telephone number.
2. Admin creates a temporary password for the user.
3. The admin grants the user access to the location they will work in the future.

**Alternative Flows:**

The admin cannot create a user because the user already exists.

**Pre-Conditions:**

The admin must log in to the system.

**Post-Conditions:**

The user is added to the system and can now log in.

### 3.5.5 Log in to the user account.

**Brief Description:**

The process of a user gaining access to a system by providing their unique identification information, which is their username and password.

**Trigger Event:**

The user wants to log in to the system.

**Main Success Scenario:**

1. The system displays the login page.
2. The user asks to enter the username and password.
3. The system displays the user's home page if the username and password are correct.

**Alternative Flows:**

1. The user is not in the system.
2. The username and password are incorrect; the system displays an error message.

**Pre-Conditions:**

The user must have an account created in the system.

**Post-Conditions:**

The user is logged in, and the user's home page gets displayed.

### 3.5.6 Logout from the user account

**Brief Description:**

Ending a user's access to the system and current session. Logging out is an essential security feature that aids in protecting sensitive data and preventing unauthorised access by ensuring the user's session is fully ended.

**Trigger Event:**

The user wants to log out of the system.

**Main Success Scenario:**

1. The user presses the logout button.
2. The system logs out the user.
3. The system displays the login page.

**Alternative Flows:**

None

**Pre-Conditions:**

The user must be logged in to the system.

**Post-Conditions:**

The user is logged out from the system, and the login page gets displayed.

### 3.5.7 Edit user profile.

**Brief Description:**

This feature allows users to change their personal information, such as their name, email address, phone number, and username, as needed. Modifying the profile is important for ensuring a user's information is current and correct.

**Trigger Event:**

The user clicks on the edit profile button.

**Main Success Scenario:**

1. The user selects the Edit Profile option.
2. The user modifies his profile.
3. The user presses the save button.

**Alternative Flows:**

The user wishes to submit the form with all fields blank. The system will show an error message. The profile will remain unchanged.

**Pre-Conditions:**

1. The user must be logged in.
2. The user must have a profile in the system.

**Post-Conditions:**

The user profile is updated.

### 3.5.8 Change Password

**Brief Description:**

Password changes are an important security measure since they safeguard user accounts from illegal access. Passwords can be changed by going to the password change page.

**Trigger Event:**

The user clicks on the Change Password button.

**Main Success Scenario:**

1. The user enters his old password and his new password.
2. The user confirms his new password.
3. The system updates the user password.

**Alternative Flows:**

1. The user enters the wrong old password. The system will display an error message.
2. The user enters a similar new password as the old password. The system will display an error message.
3. The new password is not strong enough. The system will display an error message.
4. The new password is not the same as the re-entered password. The system will display an error message.
5. The new password is less than eight characters. The system will display an error message.

**Pre-Conditions:**

The user must be logged in.

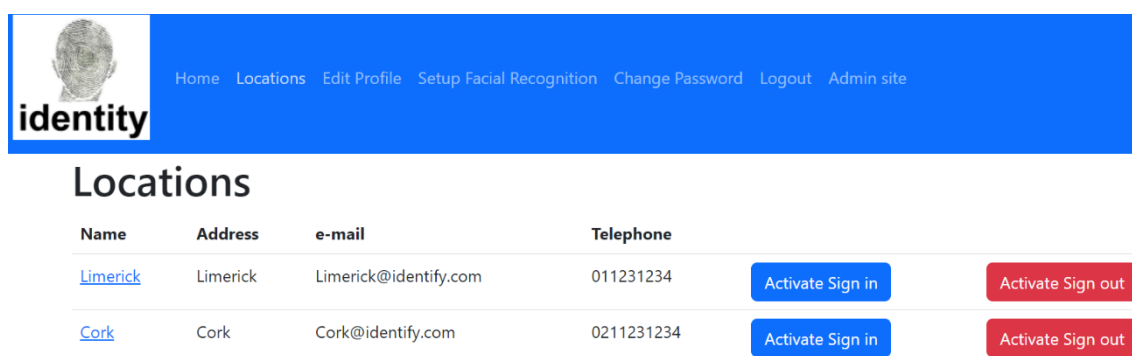
## Post-Conditions:

The user password is updated.

## 3.6 Website Layout

### 3.6.1 Locations

Clicking a location's name will give a details view of the location. Only authorised users, such as administrators or managers, can activate the sign-in /sign-off for a location.



Name	Address	e-mail	Telephone		
<a href="#">Limerick</a>	Limerick	Limerick@identify.com	011231234	Activate Sign in	Activate Sign out
<a href="#">Cork</a>	Cork	Cork@identify.com	0211231234	Activate Sign in	Activate Sign out

Figure 12 Locations list view

An unauthorised user who attempts to activate a location's sign-in/sign-off screen will receive a 403 response.

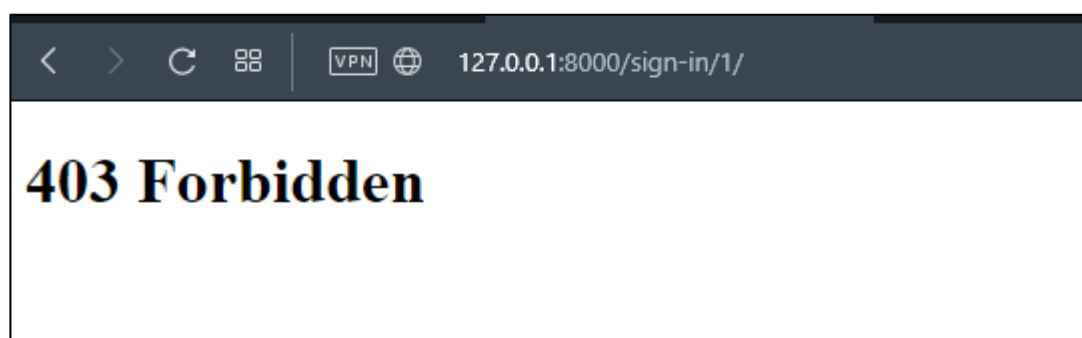



Figure 13 Security notification

### 3.6.2 Location detail's view

The location view displays the:

- Permitted users.
- active sign-in (users signed in with no sign-out yet) users.
- roster logs grouped by day.


[Home](#)
[Locations](#)
[Edit Profile](#)
[Setup Facial Recognition](#)
[Change Password](#)
[Logout](#)
[Admin site](#)

identity

Limerick

Actions

Activate Sign in

Activate Sign out

Permitted Users

Username	First Name	Last Name	Email	
<a href="#">Admin</a>			admin@admin.com	<div>Remove Permission</div>
<a href="#">ZsoltToth</a>	Zsolt	Toth	zsolt@gmail.com	<div>Remove Permission</div>
<a href="#">JoshuaFluke</a>				<div>Remove Permission</div>

Signed in Users

Username	First Name	Last Name	Sign in Date/time	
<a href="#">ZsoltToth</a>	Zsolt	Toth	Feb. 24, 2023, 8:06 p.m.	<div>None</div> <div>Sign out</div>

Day Roster Logs

[Feb. 9, 2023](#)

[Feb. 17, 2023](#)

[Feb. 18, 2023](#)


[Feb. 22, 2023](#)

[Feb. 24, 2023](#)

Figure 14 Limerick location view

### 3.6.3 View the location roster's day log.

The location's roster lists the users' sign-in and sign-out times for the day.


[Home](#)
[Locations](#)
[Edit Profile](#)
[Setup Facial Recognition](#)
[Change Password](#)
[Logout](#)
[Admin site](#)

identity

Limerick

Signed in Users for 2023-02-18

Username	First Name	Last Name	Sign in Date/time	Sign out Date/time
<a href="#">JoshuaFluke</a>			Feb. 18, 2023, 12:18 p.m.	Feb. 18, 2023, 12:18 p.m.
<a href="#">JoshuaFluke</a>			Feb. 18, 2023, 12:28 p.m.	Feb. 18, 2023, 12:58 p.m.
<a href="#">JoshuaFluke</a>			Feb. 18, 2023, 12:58 p.m.	Feb. 18, 2023, 12:59 p.m.
<a href="#">JoshuaFluke</a>			Feb. 18, 2023, 12:59 p.m.	Feb. 18, 2023, 1:40 p.m.

Figure 15 Location roster day view

### 3.6.4 Location Sign in

The location sign-in page takes pictures of the user and streams them to the server.

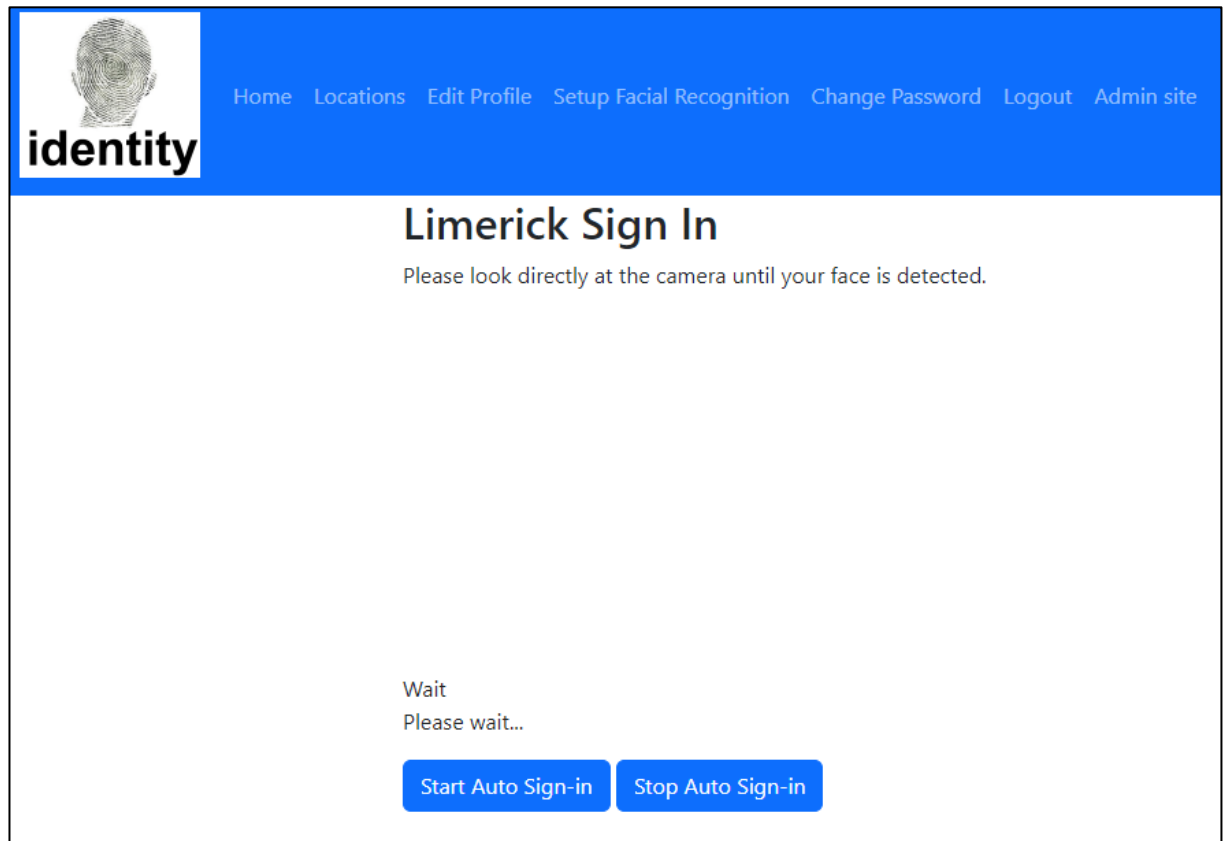


Figure 16 Location Sign In part 1

If the user is recognised, the system displays the user details on the screen and logs the day's timestamp.

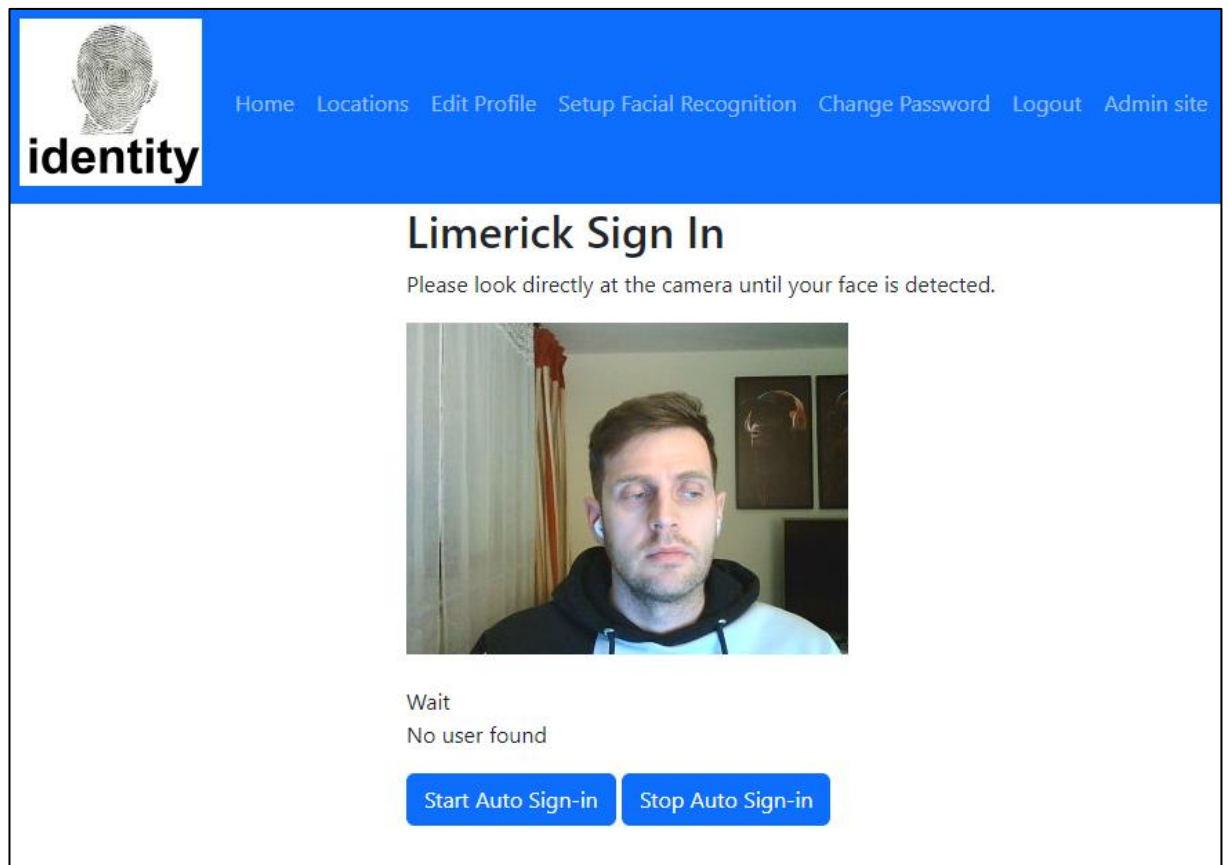


Figure 17 Location Sign In part 2



### 3.6.5 Location Sign off.

The location sign-off page takes pictures of the user and streams them to the server.

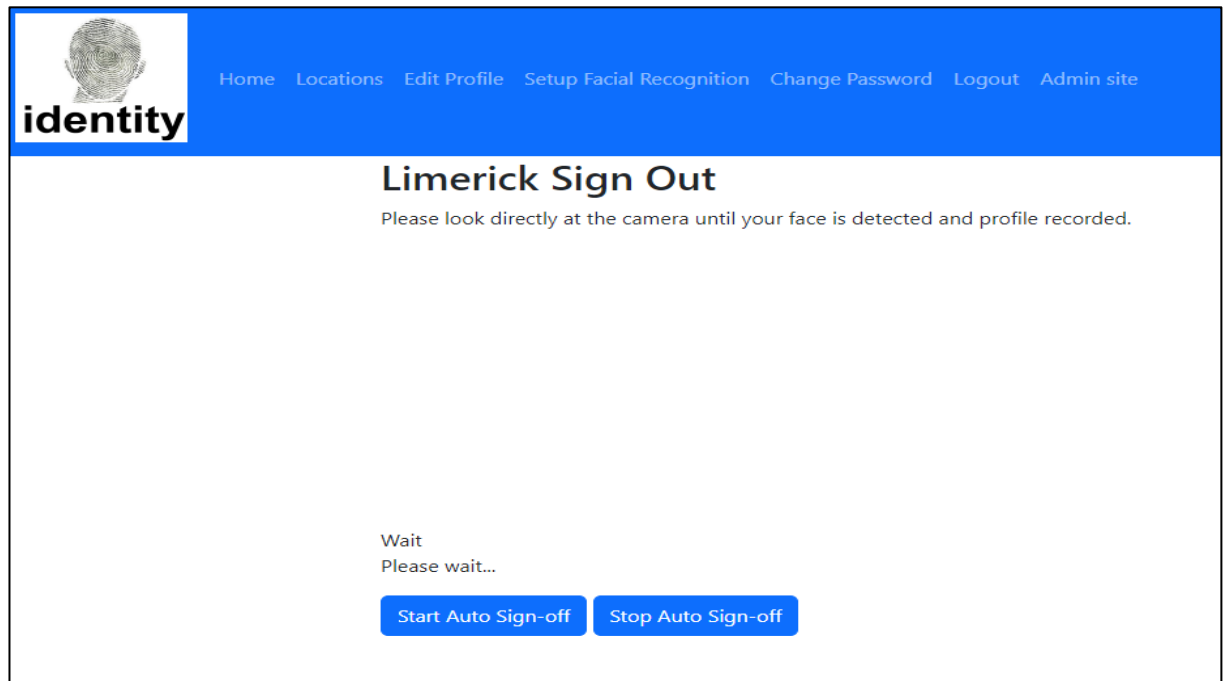


Figure 18 Location Sign Out part 1.

If the user is recognised, the system logs them out from the building and records their finishing time as the current time. This process ensures the user can leave the location and accurately record their work hours.

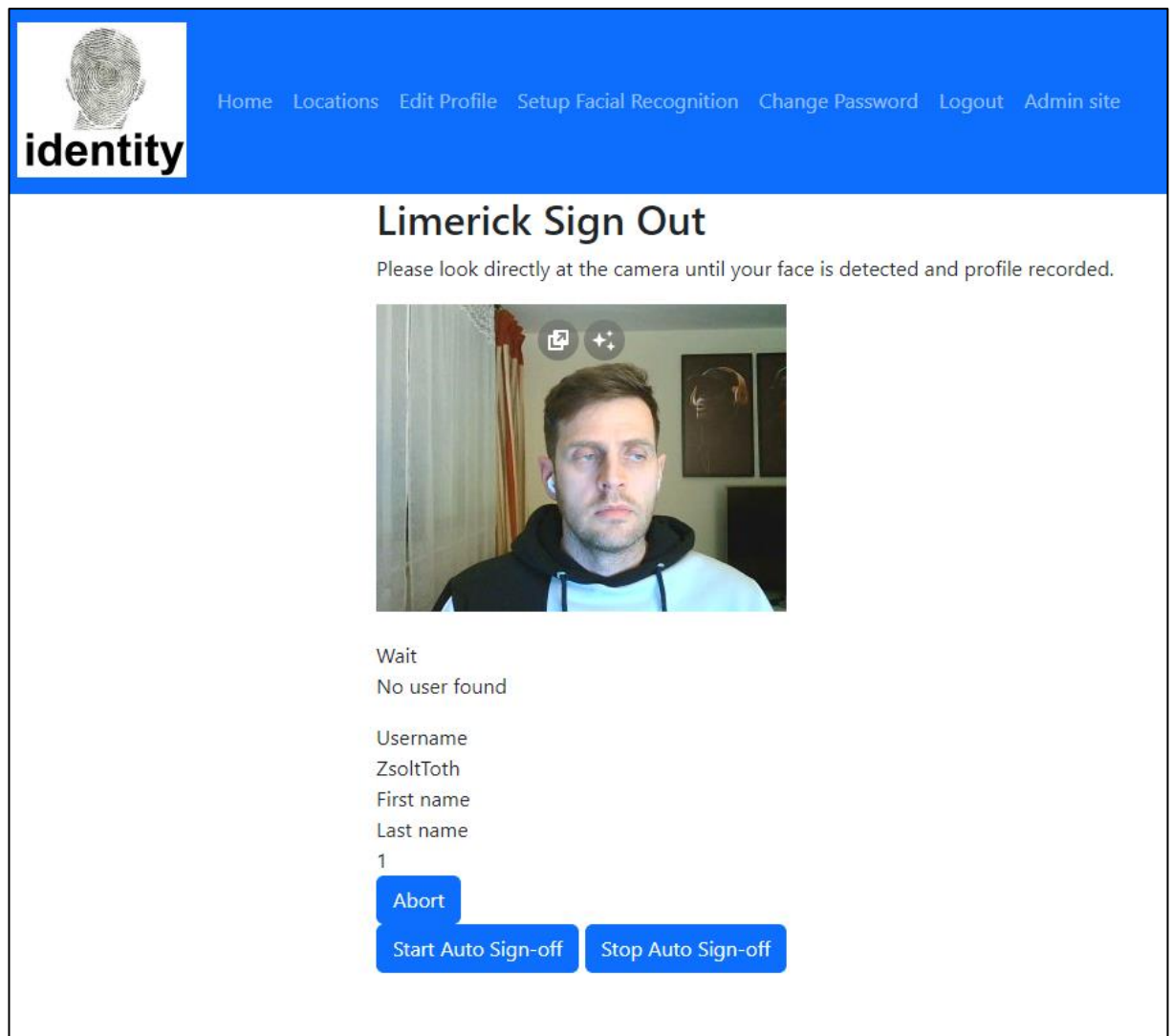


Figure 19 Location Sign Out part 2

### 3.6.6 Setup user face recognition page

The setup facial recognition page captures and uploads images of the user's face to the server. The server then processes and stores the images, trains the facial recognition model, and notifies the user of the setup outcome upon completion.

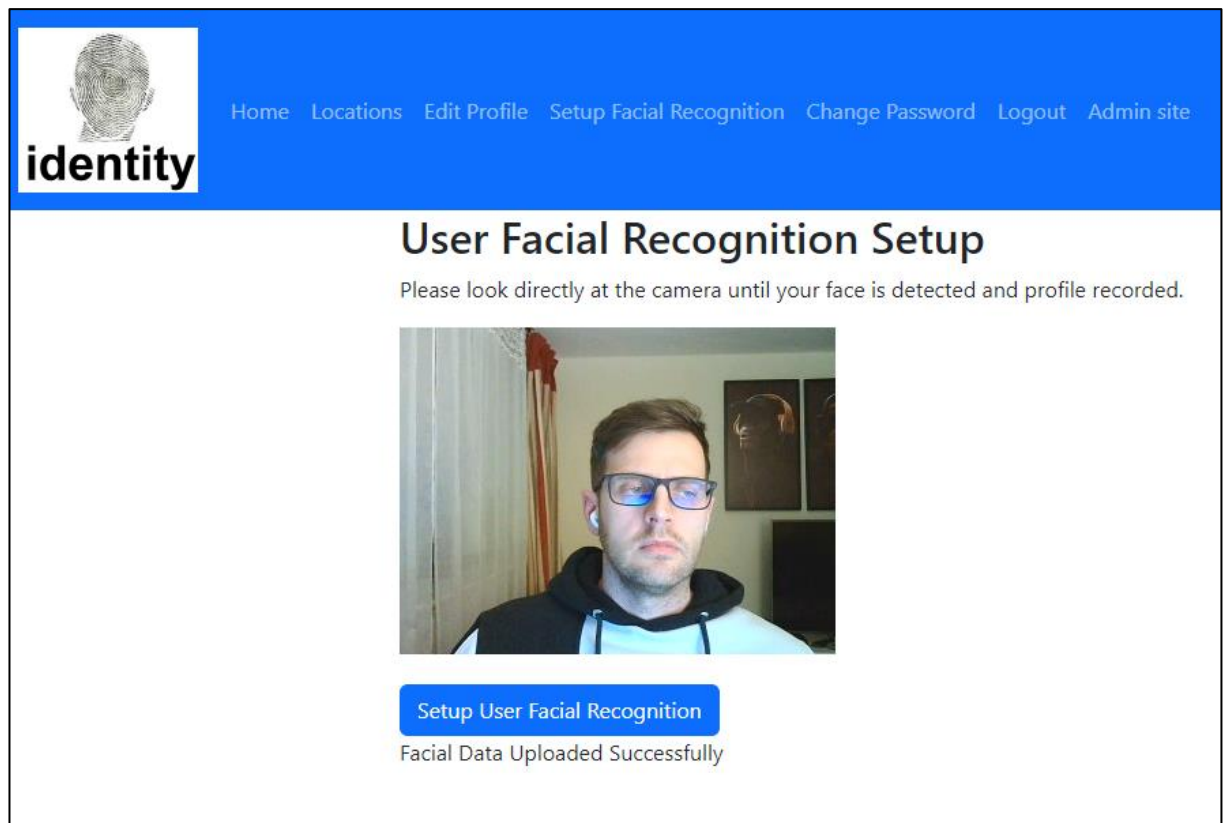


Figure 20 Setting Up Facial recognition.

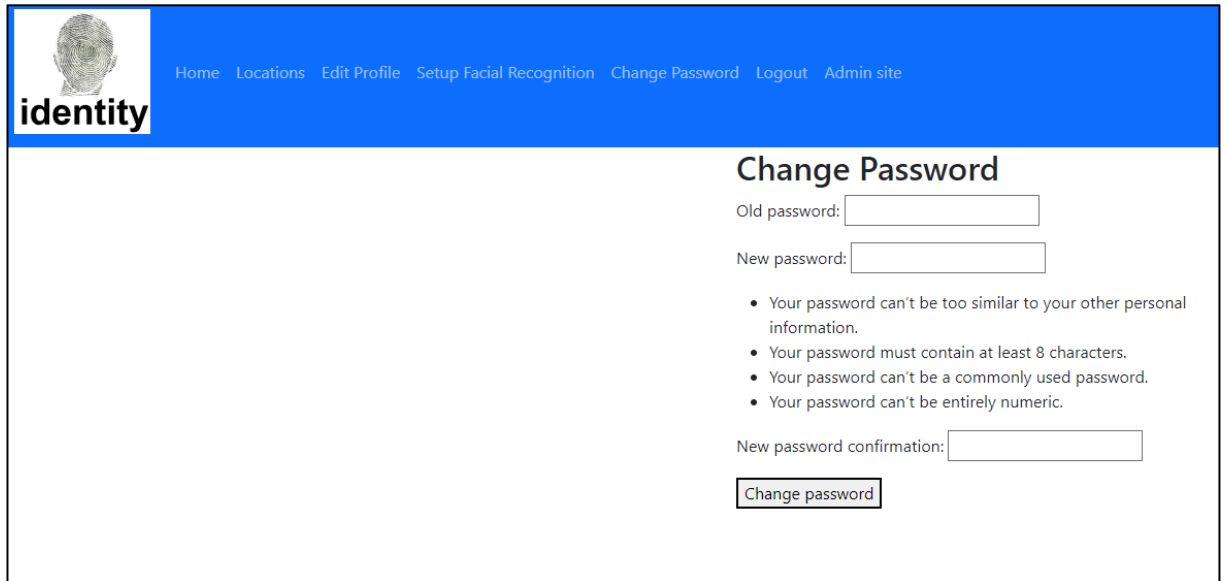
### 3.6.7 Edit Profile

This page allows users to change their personal information, such as their name, email address, phone number, and username, as needed.

Figure 21: Edit User Profile

### 3.6.8 Change Password

A User can change their password here.



identity

Home Locations Edit Profile Setup Facial Recognition Change Password Logout Admin site

## Change Password

Old password:

New password:

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

New password confirmation:

Figure 22: change-password.html

## 3.7 Tools and Framework Considered

### 3.7.1 Jupiter Notebook

Jupyter is an abbreviation for Julia, Python, and R; these are the three programming languages with which Jupyter originated, but it now supports various languages. Jupyter Notebook is an online application that is both open-source and free to use. Jupyter Notebook is made so that programming work can be shown, and others can join easily. With Jupyter Notebook, programmers can combine code, comments, multimedia, and visuals into an interactive document called a “notebook”. Recycled and used again. It lets users write and execute computer code directly in the web browser. It is useful in teaching as we can show examples of how a script or a language works.

#### Advantages of using Jupyter Notebook:

- **Data visualisation.** Most people learn about Jupyter Notebooks for the first time through data visualisation, which is a shared notebook with some datasets shown as graphics. Jupyter Notebook lets users make visualisations, share them, and make real-time changes to the shared code and dataset.
- **Code Share.** Cloud platforms like GitHub and Pastebin allow developers to exchange code but are generally inactive. Jupyter Notebook will enable

developers to preview your code, run it, and examine the results immediately in your browser.

- **Live interaction with code.** The code in Jupyter Notebook is not static; it is real-time, gradually editable, and replay-able, with feedback provided immediately in the browser. Notebooks can have user controls that can be utilised as code input sources.
- **Documenting code samples.** A developer might put some code in a Jupyter Notebook to show how it works step by step with real-time feedback. The code is still fully functional, and the developer can add interaction by explaining, delivering, and talking simultaneously.

A Jupyter Notebook can have many parts, and each one is made up of different blocks.

#### **Components of Jupyter Notebook:**

- **Text and HTML** Anywhere on the page, developers can enter plain text or content written in Markdown syntax to turn it into HTML. The notebook template can have a built-in CSS style or be added to it.
- **Code and output.** Although developers can add support for other languages in the Jupyter environment, such as R or Julia, Jupyter Notebooks' programming is usually written in Python. The code blocks can be run and repeated in whatever order as many times as desired, and the results of the executed code appear immediately after the code blocks.
- **Multimedia** Due to the fact that it is built on web technologies, Jupyter Notebook can display supported forms of multimedia on a web page. They can be pre-programmed by developers with the help of the IPython.display module, or they can be inserted into a notebook as HTML elements.
- **Data** In addition to the `.ipynb`` file that makes up a Jupyter Notebook, data can be given as a separate file or imported programmatically. For instance, code might be inserted into the notebook to download data from a public Internet repository or access a database connection.

With certain limits, Jupyter Notebook can be equally as powerful and helpful.

## The restriction set by Jupyter Notebook

- **Notebooks are not self-contained.** The fact that Jupyter Notebook requires the Jupyter runtime and the libraries the developer wants to use is its biggest drawback. There are a few ways to create independent Jupyter Notebooks, but none of them is supported by the project. It's best to install or give laptops to those with the necessary infrastructure (via Anaconda, for example).
- **The session state is difficult to save.** Using the Jupyter Notebook toolset, you can't save the current state of the code in a Jupyter notebook and then load it again. Every time you load the notebook, the developer must run the code again to get it back to the way it was.
- **There is no interactive debugging or other IDE functionality.** Jupyter Notebook does not provide a full Python programming environment. Many functions that users anticipate from an IDE, such as interactive debugging, code completion, and module management, are missing.

### 3.7.2 Python

#### What is Python?

Python is a powerful, high-level programming language that aims to be easy to read and understand. The term “high-level” refers to how closely related to human languages it is in comparison to other computer languages. Since the Python community is vibrant and active, learning this language will provide developers access to a wealth of useful tools that Python users have created since its inception.

#### Why Python?

In a nutshell, Python's popularity among developers is a result of its widespread use in a variety of industries, including data science, artificial intelligence, desktop and online application development, statistics, mathematics, and scientific research. Developers have access to several publicly accessible libraries, and the number of ready-made user-friendly tools is always growing. Its development community, which is already sizable, is expanding quickly due to the language's ease of learning and suitability for novice programmers. And if someone gets stuck on a work, it's always simple to find information, solutions, and recommendations owing to the enormous Python community.

## What does Python's simplicity mean?

Comparing the approaches that Python and other programming languages (in this case, Java, and C++) take to solve the same straightforward problem is the greatest way to illustrate Python's openness and accessibility. Let's use a classic programming example: get the terminal to say, "Hello World!"

C++	Java	Python
<pre>#include &lt;iostream&gt; int main() {     std::cout&lt;&lt;"Hello,world!\n";     return 0; }</pre>	<pre>class HelloWorldApp {     public static void main(String[] args) {         System.out.println("Hello World!");     } }</pre>	<pre>print("Hello World")</pre>

### 3.7.3 TensorFlow

Since Google released version 1.0 of its open-source AI framework in 2017, it has grown into a platform that defines the industry. With the framework and the detailed documentation that comes with it, it's very easy to put together neural networks with only a few lines of code that can solve a wide range of problems. These networks can run on both CPUs and GPUs (not to mention those specifically designed for solving AI problems). About the hardware for the TPU (Tensor Processing Unit) target (TensorFlow, 2019).

TensorFlow operation is built on a data flow-based implementation in which the computations and states of the algorithms are represented by data flow graphs, allowing for the parallelisation of diverse processes (TensorFlow, 2019). The technology also can continually modify the internal states, allowing the vast number of parameters that occur in big models to be managed (TensorFlow, 2019). The framework was created in C++ (and CUDA-C), but it is now accessible in several programming languages, including C# and Java; however, it is most used through Python language integration.

They often don't work directly with TensorFlow but rather with Keras, a higher-level software package that was not originally made for TensorFlow but is now part of it. Keras supported lower-level backends and was not originally made for TensorFlow (Team, n.d.). The project's original goal was to create a more user-friendly API interface,

achieved with network configurations that can be set up with just a few lines of code. The API also makes it easy to access important functions, such as the different activation functions used when building a network (e.g., ReLu, Sigmoid, or Softmax42). It might seem obvious that the same backend should run both the neural networks made with TensorFlow and the training, which needs the same processing power as the net size. However, the software package's popularity is demonstrated by the addition of several solutions over the years, such as TensorFlow.js, which targets browsers and other runtime environments that support JavaScript execution, or TensorFlow Lite, which can be used to drive pre-trained networks in a mobile environment (TensorFlow, 2019). They also created APIs for many languages. (An example is Tensorflow Java.) We must modify the model for both strategies, although the first can provide more training.

The author is choosing TensorFlow to build his System because TensorFlow's most well-known Deep Learning framework comes with pre-trained models that can help with image classification. CNN is used to put photos into groups. Most of the time, all it takes to make a model is to put photos into groups to make a similar, positive image. The picture is then taught and retaught using a method called anchoring or Transfer Learning.



## Chapter 4 Implementation

### 4.1 Introduction

The chapter's primary objective is to give the reader a thorough grasp of the project's implementation procedure and the technologies employed. This chapter will cover the hardware and software utilised and the factors in its selection. It describes the project's setup and highlights the components of technical interest to the project.

### 4.2 Tools Used

The software's development process, including the tools, frameworks, and libraries used.

Python 3.9	Programming language
HTML, CSS, JavaScript	Client-side programming languages
Open CV	Image recognition library
Django	Web framework
Pipreqs	Generate python package requirements
Git	Source control management

#### 4.2.1 IDE Support for Programming and Documentation

Anaconda	Manages the python environment and package compatibility
Pip	Python package installer (Sometimes used instead of Anaconda)
Visual Studio Code	The IDE
Mermaid.js	Generating diagrams
Plant UML	Generating UML diagrams

### 4.3 Source Control

Git is the source control technology used, and the source is maintained in a GitHub repository @ <https://github.com/zsolt821201/iDentity-ml-fyp-py>

### 4.4 To install and run the project.

1. Clone the repository.

```
git clone https://github.com/Zsolt821201/iDentity-ml-fyp-py.git
```

2. Install the Python dependencies.

```
pip install -r requirements.txt
```

3. Run the app in VS Code and run the app by pressing the `F5` Key or open a terminal and run the following command:

```
python manage.py runserver
```

4. Open the app in your browser @ <http://localhost:8000>

#### Built-in user accounts are:

Username	Password	Role	Note
Admin	Password	admin	Built-in admin account
ZsoltToth	Password1234!	user	Built-in admin account
JoshuaFluke	Letmein1\$	user	Built-in admin account

### 4.5 Building an OpenCv application to identify user's faces.

#### 4.5.1 How to use Open CV

The OpenCv-python package wraps the OpenCV library in Python. To utilise OpenCv in the application, this package must be installed. c.f. 6.4 Installing and running the project.

In Python files, use the following code to import the package.

```
import cv2
```

#### 4.5.2 How to find a face in an image using OpenCv:

Given a grey-scale image Mat object as input, the function `detect\_user\_face` returns a tuple containing a Boolean flag and a numpy array. The flag is True if a face is detected in the image and False if no face is detected. The numpy array contains the coordinates of the face in the image.

CLASSIFIER\_CONFIGURATION\_FILEPATH is a string constant of the configuration file path. The path is a relative path calculated at runtime.

#### **detect\_user\_face:**

```
1. def detect_user_face(gray_scale_image, min_size=None) -> tuple[bool, ndarray]:
2.     face_detector_classifier =
cv2.CascadeClassifier(CLASSIFIER_CONFIGURATION_FILEPATH)
3.     faces: ndarray = face_detector_classifier.detectMultiScale(
4.         gray_scale_image, scaleFactor=1.3, minNeighbors=5, minSize=min_size)
5.
6.     if len(faces) == 0:
7.         print("Error: No face detected")
8.         return False, None
9.
10.    if len(faces) > 1:
11.        print("Error: More than one face detected")
12.        return False, None
13.
14.
15.    return True, faces[0]
```

On line 2, a CascadeClassifier object is created with the classifier configuration file path and assigned to the variable face\_detector\_classifier.

The detectMultiScale method of the face\_detector\_classifier object detects faces in the image. The detectMultiScale method returns a numpy array of faces. The parameters of detectMultiScales used are:

- The image to be analysed, i.e., **gray\_scale\_image**
- **scalefactor** Some faces in a group photo may be closer to the camera than others. It makes sense that these faces would stand out more than the ones behind them. This factor compensates for that. The intended use of the system is that only one person is looking at the camera at a time; therefore, no scaling factor might not be needed.
- The **minNeighbors** parameter specifies the minimum number of neighbour rectangles (also known as candidate rectangles) that need to be detected around an object for it to be considered a valid detection. It is a filtering mechanism to eliminate false positives and improve detection accuracy.
- **minSize** – Minimum possible object size. Objects smaller than that are ignored. The Identity application uses a default min size of None, i.e., is ignored unless otherwise specified.

Each face is an array of four numbers. The numbers represent a rectangle location of the face in the `gray_scale_image`:

- the first number is the x coordinate of the top left corner of the rectangle.
- the second number is the y coordinate of the top left corner of the rectangle.
- the third number is the width of the rectangle.
- the fourth number is the height of the rectangle.

The `detect_user_face` function checks for the cases of (1) no face detected and (2) more than one face detected. If either of these cases occurs, the function returns `False` and `None`. If a single face is detected, the function returns `True` and the coordinates of the face in the image.

### **`detect_and_save_user_face`**

```
1. def detect_and_save_user_face(user_account_id, image, image_number):
2.     gray_scale_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
3.
4.     face_found, face = detect_user_face(gray_scale_image)
5.     if face_found:
6.         x, y, w, h = face
7.         directory: str = f"{DATABASE_FACE_DIRECTORY}/user-{user_account_id}"
8.         os.makedirs(directory, exist_ok=True)
9.         cv2.imwrite(f"{directory}/{image_number}.jpg",
10.                    gray_scale_image[y:y+h, x:x+w])
11.
12.     return face_found
```

The `detect_and_save_user_face` function accepts as input a user account id, a picture, and an image number. To detect a face in the picture, the function transforms the image to grayscale and executes the detect user face function. OpenCV often uses grayscale images in computer vision tasks due to their reduced complexity, simplification of algorithms, improved robustness, and noise reduction capabilities. Grayscale images have one channel and fewer data to process, allowing faster processing and fewer computational resources. They enable algorithms to focus on essential image aspects without the complexity of colour information, leading to better performance in applications like face recognition. However, in specific applications where colour information is crucial, using colour images or combining grayscale and colour features may yield better results. The choice depends on the problem and the goals of the computer vision application. The function stores a face in the database directory when it is found. The method returns a Boolean flag indicating the detection of a face.

#### 4.5.3 How to train a facial recognition model using OpenCV:

The `face_training` function trains the model using the images in the directory `DATABASE_FACE_DIRECTORY`.

The function saves the trained model to the `DATABASE_FACIAL_TRAINER` file. `recognizer = cv2.face.LBPHFaceRecognizer_create()` creates a face recogniser object. The `recogniser.train` method takes the list of images and the list of user ids as input and trains the model. With the model-trained `recogniser.write` method saves the trained model to the `DATABASE_FACIAL_TRAINER` file.

```
1. def face_training():
2.     faces, face_ids = get_images_and_labels(DATABASE_FACE_DIRECTORY)
3.
4.     recognizer = cv2.face.LBPHFaceRecognizer_create()
5.     recogniser.train(faces, numpy.array(face_ids))
6.
7.     # Ensure the database directory exists before saving the model
8.     os.makedirs(DATABASE_DIRECTORY, exist_ok=True)
9.     # Save the model into trainer/trainer.yml
10.    recogniser.write(DATABASE_FACIAL_TRAINER)
```

The `get_images_and_labels` function takes a directory path as input and returns a tuple of two lists. The first list contains NumPy arrays of images of users' faces, and the second list contains the corresponding user IDs.

The function works by first initialising empty lists for **face\_ids** and **face\_samples**, which will later be populated with user IDs and face images. It then retrieves a list of subdirectories within the specified directory, where each subdirectory represents a user and contains images of their face.

The function then iterates over each subdirectory and retrieves its image files. It extracts the user ID from the subdirectory name using string manipulation techniques and stores it in the **face\_id** variable.

For each image file within a user's subdirectory, the function loads the image using the PIL library, converts it to a NumPy array of type 'uint8', and appends it to the **face\_samples** list. It also appends the corresponding user ID to the **face\_ids** list.

Finally, the function returns a tuple containing the **face\_samples** and **face\_ids** lists.

```
1. def get_images_and_labels(path) -> tuple[list, list]:
2.     face_ids: list[str] = []
3.     face_samples: list[ndarray] = []
4.
5.     user_directories = [os.path.join(path, directory)
6.                          for directory in os.listdir(path)]
7.     for user_directory in user_directories:
8.         user_image_files = [os.path.join(
9.             user_directory, file) for file in os.listdir(user_directory)]
10.        face_id = int(os.path.split(user_directory)[-1].split("-")[1])
11.
12.        for image_path in user_image_files:
13.            face_image_numpy: ndarray = numpy.array(
14.                Image.open(image_path), 'uint8')
15.            face_samples.append(face_image_numpy)
16.            face_ids.append(face_id)
17.
18.    return face_samples, face_ids
```

#### 4.5.4 How to recognise a face in an image using a trained model in OpenCV

To recognise a face in an image, a trained model named **trainer.yml** is required. The **face\_recognition** function takes an image as input and utilises the **face\_image\_recognition** function to recognise a face in the image.

The line **recogniser = cv2.face.LBPHFaceRecognizer\_create()** creates an instance of the Local Binary Patterns Histograms (LBPH) face recogniser in OpenCV. An approach to face identification that describes local texture patterns in photos is the LBPH face recogniser. It is based on the Local Binary Patterns (LBP) operator, which compares the intensity of a centre pixel with its neighbours to encode the local structure in an image. The resulting histograms can be utilised as feature vectors in recognition tasks by combining the LBP codes when developers use the line ``recogniser = cv2.face.LBPHFaceRecognizer_create()`` to make an LBPH face recogniser instance; they can then train it with a set of labelled face images and use it to predict labels (such as person IDs) for new face images it has never seen before.

The **recogniser.read()** function is used to load a face recogniser model from a file that has already been trained. This lets developers save the trained model to disc and reuse it later without retraining the recogniser every time you run your application. The **read()**

function takes a file path as an argument and uses that file to load the model data into the recogniser instance.

```
1. def face_recognition(open_cv_image: ndarray):
2.     recogniser:cv2.face.LBPHFaceRecognizer = cv2.face.LBPHFaceRecognizer_create()
3.     recogniser.read(DATABASE_FACIAL_TRAINER)
4.
5.     user_id, confidence, face = face_image_recognition(recognizer, open_cv_image)
6.
7.     return user_id, confidence
```

The `face_image_recognition` function takes a recogniser object, an image, and a minimum size as input. The function converts the image to a grey-scale image and calls the `[detect_user_face](#how-to-find-a-face-in-an-image)` function to detect a face in the image. If a face is detected, the function calls the `recogniser.predict` method to recognise the face. The `recogniser.predict` method returns a tuple containing the user id and the confidence of the prediction. The `face_image_recognition` function returns a tuple containing the user id, the prediction's confidence, and the face's coordinates in the image.

```
1. def face_image_recognition(recognizer: cv2.face.LBPHFaceRecognizer, image, min_size =
None):
2.     gray_scale_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
3.
4.     is_face_present, face = detect_user_face(gray_scale_image, min_size=min_size)
5.
6.     if(not is_face_present):
7.         return None, 0, None
8.
9.     x, y, width, height = face
10.    user_id, confidence = recognizer.predict(gray_scale_image[y:y+height, x:x+width])
11.
12.    return user_id, confidence, face
```

#### 4.5.5 Support for OpenCv processing web inputs

When images are transferred via HTTP requests, they are typically encoded in Base64.

To manipulate images in Python, it is necessary to convert the image into a numpy array. The `image_to_numpy_array` function accepts an image file path as input and returns a numpy array, utilising the `PIL.Image.open` method to open the image file and the `numpy.array` method to convert the image.

##### 4.5.5.1 File Encoding

To processes Base64 encodings in OpenCV, these images must first be decoded. The `decode_base64` function receives a Base64 encoded string as input and returns a byte array. To do so, the `re.sub` method removes the “data:image/.\*;base64”, prefix from the string, while the `base64.b64decode` method decodes the string.

```

1. import base64
2. import re # Regular expression
3.
4. def decode_base64(image_base64_str):
5.     image_data = re.sub('^data:image/.+;base64,', '', image_base64_str)
6.     return base64.b64decode(image_data)

```

The `stream_image` function creates a `BytesIO` object, taking a Base64 encoded string as input. The function utilises the `decode_base64` function to decode the string and, subsequently, the `BytesIO` method to produce the `BytesIO` object.

```

1. from io import BytesIO
2.
3. def stream_image(image_base64_str: str) -> BytesIO:
4.     return BytesIO(decode_base64(image_base64_str))

```

This `BytesIO` object can be passed to the `numpy.array` and `PIL.Image.open` methods to create a `numpy` array that can be used with the `OpenCV` functions.

```

1. import numpy
2. from PIL import Image
3.
4. def parse_roaster_signing_requests(request) -> tuple[bool, UserAccount]:
5.     request_data = request.POST
6.     image_bytes = stream_image(request_data['image-base64'])
7.     open_cv_image = numpy.array(Image.open(image_bytes))
8.     #...

```



#### 4.5.5.2 Creating Sample users

Sample users were generated from YouTube videos for demonstration purposes. The code below can create a sample user by using a YouTube video.

```
1. def build_sample_user(video_path:str, session_user_account_id:str):
2.
3.     video_capture = cv2.VideoCapture(video_path)
4.     face_detector_classifier = cv2.CascadeClassifier(CLASSIFIER_CONFIGURATION)
5.
6.     image_number: int = 0
7.     FACE_SAMPLE_COUNT=30
8.     ESCAPE_KEY: int = 27
9.     while(image_number < FACE_SAMPLE_COUNT):
10.         is_video_capture_open, open_cv_image = video_capture.read()
11.
12.         if not is_video_capture_open:
13.             print("Error: Camera is not opened")
14.             break
15.
16.         cv2.imshow('image', open_cv_image)
17.
18.         gray_scale_image = cv2.cvtColor(open_cv_image, cv2.COLOR_BGR2GRAY)
19.
20.         faces: ndarray = face_detector_classifier.detectMultiScale(
21.             gray_scale_image, 1.3, 5)
22.
23.         face_found = detect_and_save_user_face(
24.             session_user_account_id, open_cv_image, image_number)
25.         if face_found:
26.             image_number += 1
27.
28.
29.
30.         for (x, y, w, h) in faces:
31.             cv2.rectangle(open_cv_image, (x, y), (x+w, y+h), (255, 0, 0), 2)
32.             cv2.imshow('image', open_cv_image)
33.
34.             if cv2.waitKey(1) == ESCAPE_KEY:
35.                 break
36.         # Do a bit of cleanup
37.         video_capture.release()
38.         cv2.destroyAllWindows()
```

### 4.6 Building Identity web project

Django was chosen as the framework to build the website with.

#### 4.6.1 Using Django

The following is required to use Django:

1. Python
2. Django package

#### 4.6.1.1 Create a new Django project.

Once Django is installed, you can create a new project by running the following command:

```
django-admin startproject identity_website
```

Replace `project\_name` with the name of your project. This command will create a new directory with the same name as your project, containing the basic files and folders needed to start a new Django project.

Create a new Django app: A Django project comprises one or more apps. An app is a module that contains models, views, templates, and other code that serves a specific purpose.

You can create a new app by running the following command:

```
python manage.py startapp identity
```

Replace `app\_name` with the name of your app. This command will create a new directory with the same name as your app, containing the basic files and folders needed to start a new Django app.

#### 4.6.2 The Models

Django will implement the database using the definitions in the `models.py` file. The developer never has to write SQL code. The `models.py` file defines the data structures for your application. Each class in `models.py` represents a table in the database. Each attribute of the class represents a column in the table. The following code defines a Location class representing a database table.

E.g.

```
1. from django.db import models
2.
3.
4. class Location(models.Model):
5.     id = models.AutoField(primary_key=True)
6.     address = models.CharField(max_length=200)
7.     description = models.CharField(max_length=200)
8.     email = models.EmailField(max_length=200, unique=True)
9.     name = models.CharField(max_length=200, unique=True)
10.    telephone = models.CharField(max_length=20)
11.
12.    def __str__(self):
13.        return self.name
```

creates a Table in the database

```
1. CREATE TABLE "identity_location"
2. ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,
3.  "address" varchar(200) NOT NULL,
4.  "description" varchar(200) NOT NULL, "email" varchar(200) NOT NULL UNIQUE, "name"
   varchar(200) NOT NULL UNIQUE, "telephone" varchar(20) NOT NULL)
```

#### 4.6.2.1 Migrations

To create the database, perform these migrations. Note 0001 increments with each migration. A second migration would be 0002, and so on.

```
1. python manage.py makemigrations identity
2. python manage.py sqlmigrate identity 0001
3. python manage.py migrate
```

#### 4.6.2.2 Overriding the default user Model

In Django, user accounts are represented with a built-in User model. The Identity App requires custom fields for its User model not present in `django.contrib.auth.User`.

To override the default User model:

1. Create a new model, in this example, named `UserAccount`, that inherits from `AbstractBaseUser` classes. This new model will be used instead of `django.contrib.auth.User`.
2. Add any additional fields you require to the new model `UserAccount`.
3. When referencing the `UserAccount` Model, use `settings.AUTH_USER_MODEL` rather than `UserAccount`

```
1. class Roster(models.Model):
2.     id = models.AutoField(primary_key=True)
3.     #...#
4.     user_account = models.ForeignKey(
5.         settings.AUTH_USER_MODEL,
6.         on_delete=models.CASCADE,
7.     )
```

4. Then set the `AUTH_USER_MODEL` setting in the Website App `settings.py` file to point to your new User model `UserAccount` with its namespace.

```
AUTH_USER_MODEL = 'identity.UserAccount'
```

#### 4.6.3 The Views

In Django, the `views.py` file defines the actions that will take place when a user accesses a specific URL. Essentially, each function within this file acts as a controller in a standard MVC architecture, facilitating the communication between the model and views of the

application. Views can use templates to generate the HTML presented to the user and alternatively return data in JSON format that a JavaScript application can utilise.

#### 4.6.3.1 *The Templates*

Templates is a project folder that contains HTML files. The `views.py` file and the `templates` folder must be in the same directory.

The following code, for example, provides a function named `index` that returns a rendered template called `website/index.html` relative to the `templates` folder.

```
def index(request):  
    return render(request, 'website/index.html')
```

#### 4.6.3.2 *The Static Files*

The application does not produce static files like images, CSS, and JavaScript files. These files are typically stored within a folder named "static" in the project directory, which should be in the same directory as the `views.py` file to refer to a static file within a template; the "static" tag can be utilised.

```
{% load static %}  
  
<!--...-->  
  
  
  
<!--...-->
```

#### 4.6.3.3 *Passing Data to Templates(The Context)*

The context is a dictionary containing the information supplied to the template. As the third parameter, the context is supplied to the `render` method.

The identified parameters in the context are then available as variables in the template. The following code, for example, establishes a method named `setup facial recognition`, which returns a rendered template from `user-accounts/setup-facial-recognition.html` in the `templates` folder.

#### **views.py**

```
1. def setup_facial_recognition(request):  
2.     user_account = get_object_or_404(UserAccount, pk=request.user.id)  
3.     context = { 'user_account_id': user_account.id }  
4.     return render(request, 'user-accounts/setup-facial-recognition.html', context)
```

## user-accounts/setup-facial-recognition.html

```
1. {% extends "master.html" %}
2. {% load static %}
3. {% block content %}
4.     <div class="column">
5.         <h2>User Facial Recognition Setup</h2>
6.         <p>Please look directly at the camera until your face is detected and profile
recorded.</p>
7.         <video id="videoInput" width="320" height="240">
8.             </video>
9.         <p id="errorMessage" class="error"></p>
10.        <input type="hidden"
11.            id="user-account-id"
12.            name="user-account-id"
13.            value="{{ user_account_id }}" />
14.        <button id="startButton"
15.            class="btn btn-primary"
16.            onClick="setupUserFacialRecognition()">
17.            Setup User Facial Recognition
18.        </button>
19.        <p id="statusMessage"></p>
20.    </div>
21. {% endblock content %}
22. {% block scripts %}
23.     <script src="{% static 'js/facial-login.js' %}" %}</script>
24.     <script type="text/javascript">startCamera(VideoResolutionFormatNames.QVGA,
'videoInput');</script>
25. {% endblock scripts %}
```

### 4.6.3.4 Extending templates

A base template is one from which other templates are extended. Typically, they include named blocks that other templates can overwrite with material relevant to a particular view, along with the HTML shared by all pages in the application. By doing this, one may avoid using the same HTML twice in different templates. A base template, for instance, may have the HTML for the header, footer, and navigation bar. After that, additional templates can build on the basic template by adding new HTML to the designated blocks.

For instance, the following code blocks build a basic template with a title block named master.html, and login.html extends the base template and replaces the title block.

## master.html

```
1. <html lang="en">
2.     <head>
3.         <title>Identity
4.             {% block title %}
5.             {% endblock title %}
6.         </title>
7.     </head>
8.     <body></body>
9. </html>
```

## login.html

```
1. {% extends 'identity/master.html' %}
2. {% block title %}Login{% endblock title %}
```

Only the text inside the block tags is required to construct a view. The base template is used to create the remaining HTML.

### 4.6.3.5 Handling 404 Errors

To improve the code, the function `get_object_or_404` can generate a 404-error message if the object cannot be located. For instance, consider the function below called "user\_account", which returns a rendered template from the "user-accounts/user-account.html" file, located relative to the "templates" folder. If the user account cannot be located, a 404-error message will be returned.

```
1. def location_details(request, location_id):
2.     location = get_object_or_404(Location, pk=location_id)
```

### 4.6.3.6 Securing the Views

Decorators aid with the security of an app. The `@login_required` decorator ensures a user is logged in before using a method. The `@permission_required` decorator checks that a user has certain permission before calling a method. The `@csrf_exempt` decorator enables the call of a function from a POST request.

The `@login_required` decorator will send the user to the login page if they are not signed in. If a user does not have the necessary permissions, the `@permission_required` decorator will produce a 403 error and redirect the user properly.

```
1. from django.contrib.auth.decorators import login_required, permission_required
2.
3. @login_required
4. @permission_required('identity.activate_sign_in', raise_exception=True)
5. def sign_in(request, location_id):
6.     location = get_object_or_404(Location, pk=location_id)
7.     return render(request, 'user-accounts/sign-in-new.html', {'location': location})
```

### 4.6.3.7 Securing blocks of the Templates

To safeguard certain sections of templates, the `if` tag can be utilised. Specifically, the `if` tag, in combination with the `user.is_authenticated` function can be used to verify if a user is authenticated. In the following example, the `if` tag is implemented to exhibit the

appropriate navigation bar links when a user is authenticated. However, the login link will be displayed instead if a user is not authenticated.

```
1. from django.contrib.auth.decorators import login_required, permission_required
2.
3. @login_required
4. @permission_required('identity.activate_sign_in', raise_exception=True)
5. def sign_in(request, location_id):
6.     location = get_object_or_404(Location, pk=location_id)
7.     return render(request, 'user-accounts/sign-in-new.html', {'location': location})
```

To determine if a user has a specific permission, the if tag in combination with the perms.identity.activate\_sign\_in function can be utilised. In the example provided below, the if tag is implemented to exhibit the "Activate Sign In" button only if the user has the permission "identity.activate\_sign\_in".

```
1. <td><p id="my-desc">Actions</p></td>
2. <td>
3.     {% if perms.identity.activate_sign_in %}
4.     <a class="btn btn-primary" href="{% url 'sign_in' location.id %}">Activate Sign n</a>
5.     {% endif %}
6. </td>
```

#### 4.6.4 The URLs

In the Identity project, the urls.py file can be found within the Identity folder and establishes a link between URLs and their corresponding view functions as defined in views.py. This is accomplished through the path function, which maps URL patterns to their associated views. The name parameter can also be specified in the path function to provide a reference to the URL pattern within templates, enabling developers to create dynamic links between pages.

```
1. from django.urls import path
2. from . import views
3. from .views import UserEditView, PasswordsChangeView
4.
5. urlpatterns = [
6.     # ...
7.     path('', views.index, name='index'),
8.     path('edit-user-profile/', UserEditView.as_view(), name="edit-user-profile"),
9.     path('locations/', views.locations, name='locations'),
10.    path('locations/<int:location_id>/', views.location_details, name='details'),
11.    # ...
```

```
1. path('locations/', views.locations, name='locations')
```

The URL is then accessed as follows: <http://localhost:8000/locations>

URL parameters are defined using angle brackets, e.g., <int:location\_id>. The int parameter defines the type of the parameter. The location\_id parameter is the name of the

parameter. The parameter is then available in the function, e.g., `def location_details(request, location_id)`. The URL is then defined as follows:

```
1. path('locations/<int:location_id>/', views.location_details, name='details')
```

The URL is then accessed as follows: <http://localhost:8000/locations/1/>

Generic views are used to map the URL to the function by using the `as_view` function, e.g., `path('edit-user-profile/', UserEditView.as_view(), name="edit-user-profile")`.

The URL is then accessed as follows: <http://localhost:8000/edit-user-profile>

#### 4.6.4.1 Referencing the URLs

The `url` function can be utilised to reference a URL within a Django template. This function generates a URL based on the name of the URL pattern defined in the `urls.py` file and any associated parameters.

In the example provided, the `url` function is used within a template called `master.html` to generate a link to the "locations" URL pattern. The `url` function takes a single argument, which is the name assigned to the URL pattern in the `urls.py` file (in this case, "locations"). When the template is rendered, this `url` function will generate a URL corresponding to the "locations" URL pattern, which will be used as the `href` attribute for the "Locations" link.

```
1. <a class="nav-link" href="{% url 'locations' %}">Locations</a>
```

#### 4.6.4.2 Adding the project URLs to the website

To incorporate the URLs from the Identity app into the main website app, the code `path('', include('identity.urls'))` can be added to the website's `urls.py` file. This code specifies that any URLs defined within the identity app should be included within the website app's URLs at the root level.

In the provided code snippet, the `include` function incorporates the URLs defined within the `'identity.urls'` file. The empty string `"` in the first argument of the `path` function specifies that these URLs should be added to the root level of the website. The admin



URL is also included in the url patterns list, allowing developers to access the Django admin panel.

```
1. from django.contrib import admin
2. from django.urls import path, include
3.
4. urlpatterns = [
5.     path('admin/', admin.site.urls),
6.     path('', include('identity.urls')),
7. ]
```

#### 4.6.5 The admin

The default Django application, the admin site, facilitates a user-friendly interface for managing site content. It can be accessed via **/admin** and enables the management of database tables and user accounts.

admin.py

Custom forms can be utilised and created in the admin.py file to enhance the admin site with extra fields.

#### 4.6.6 The Forms

Django forms, which differ from HTML forms, are employed to verify the accuracy of user-entered data. These forms are defined in the forms.py file. Additionally, Crispy Forms is a third-party Django package that streamlines form creation with less code, allowing for the creation of visually appealing forms. This package offers a convenient method for arranging HTML forms through customisable template packs, which can be easily adapted to meet project specifications.

The package django-crispy-forms is required to use the crispy forms in Django. The package can be installed using the following command:

```
1. pip install django-crispy-forms
```

In the settings.py file, the crispy forms are added to the installed apps.

```
1. INSTALLED_APPS = [
2.     # ...
3.     'crispy_forms',
4. ]
```

In the settings.py file, the crispy forms template pack is added to the template context processors.

```
1. TEMPLATES = [  
2.     {  
3.         'BACKEND': 'django.template.backends.django.DjangoTemplates',  
4.         'DIRS': [],  
5.         'APP_DIRS': True,  
6.         'OPTIONS': {  
7.             'context_processors': [  
8.                 # ...  
9.                 'django.template.context_processors.request',  
10.            ],  
11.        },  
12.    ],  
13.]
```

Example use of the crispy forms in the template [edit-user-profile.html](#)

```
1. {% extends "master.html" %}  
2. {% load static %}  
3. {% block title %}  
4.     Edit User Profile  
5. {% endblock title %}  
6. {% block content %}  
7.     <form method="post">  
8.         <div class="container" style="width: 30rem;">  
9.             <h1>Edit User Profile</h1>  
10.            <div class="form-group">  
11.                {% csrf_token %}  
12.                {{ form.as_p }}  
13.                <br/>  
14.                <button class="btn btn-primary" type="submit">Update Profile</button>  
15.            </div>  
16.        </div>  
17.    </form>  
18. {% endblock content %}
```

The tag {{ form.as\_p }} is used to render the crispy form.

#### 4.6.7 Utilities

The application code for image processing was placed in `utilities.py` This strategy offers many advantages. Secondly, isolating the image processing logic from the core application code enables better code structure. Second, it allows the reuse of the code across other application components. Thirdly, testing is simpler because the image processing functionality can be tested separately from the rest of the programme.

Haarcascades are pre-trained classifiers that are used to detect objects in images. The identity project used haarcascade\_frontalface\_default.xml to detect faces in images.

#### 4.6.8 Constants and File Paths

Constants for FilePaths and application Settings are defined at the top of the utilities.py file.

```
1. PARENT_DIRECTORY: Path = Path(__file__).resolve().parent.parent
2. CLASSIFIER_CONFIGURATION: str = str(
3.     PARENT_DIRECTORY / 'haarcascade_frontalface_default.xml')
4. DATABASE_DIRECTORY: str = str(PARENT_DIRECTORY / 'database/')
5. DATABASE_FACE_DIRECTORY: str = str(
6.     PARENT_DIRECTORY / 'database/identity_face_dataset')
7. DATABASE_LOG_DIRECTORY: str = str(
8.     PARENT_DIRECTORY / 'database/log')
9. DATABASE_FACIAL_TRAINER: str = str(PARENT_DIRECTORY / 'database/trainer.yml')
10. FACE_CONFIDENCE_LEVEL: float = 80.0
```

FACE\_CONFIDENCE\_LEVEL is the maximum confidence level acceptable that a face can be considered a match. The lower the value, the more confident the application is that the face is identified correctly. File paths are defined relative to the utilities.py file for the directories used by the application, such as the database directory, the database/identity\_face\_dataset directory, and the database/trainer.yml file.

#### 4.6.9 Face Detection

The OpenCV library detects faces in images and trains the application to recognise them. At the same time, pre-trained classifiers known as Haarcascades are employed to identify objects in images.

Haarcascade\_frontalface\_default.xml, which is available on [github.com/kipr/opencv](https://github.com/kipr/opencv), was utilised by the identity project to detect faces in images.

### 4.7 Client-Side Functionality

JavaScript is employed to transmit a video stream from the client machine to the server. The server is configured to monitor three different use cases, which include:

- Setting up user facial recognition (Use case #1)
- Signing in at a location roaster (Use case #2)
- Signing out at a location roaster (Use case #3)

#### 4.7.1 Use Case #1: User Facial Recognition Setup

The initial code development focused on facilitating user setup of facial recognition capabilities. To accomplish this, a collection of procedural functions were implemented. However, several challenges arose with using procedural functions during the

developmental and testing phases. Specifically, these functions proved challenging to test, maintain, and repurpose for other use cases.

#### 4.7.2 Use Cases #2-3: Location Roaster Sign in/out.

Use cases 2 and 3 exhibit similarities in their operations, with both employing the transmission of a video stream, location ID, and user ID to the server. The sole distinction between the two cases is the action executed by the server. Use case 2 pertains to user sign-in, while use case 3 pertains to user sign-out. Within the client-side JavaScript code, the action of sign-in or sign-out is established based on the URL of the request.

#### 4.7.3 Refactoring JavaScript

The client-side JavaScript code was refactored to address issues with the procedural functions used in the use case 1, incorporating classes designed to be reusable for use cases 2 and 3 while independent of the associated HTML page elements. This was accomplished through dependency injection, which permits the passing of HTML page elements to the class as parameters during construction.

```
1. const webPageControls = new WebPageControls();  
2. const manager = new Manager(webPageControls, UrlPaths.PERFORM_SIGN_OUT_URL);
```

The `WebPageControls` class is responsible for the HTML page elements and is implemented to abstract such elements from the remainder of the code. This permits the code to be reused across various HTML pages while facilitating enhanced maintainability by providing a centralised access point to the HTML page elements.

The `Manager` is a class used to manage the video stream. The class abstracts the video stream from the rest of the code. This allows the code to be reused for different use cases. The class also provides a single point of access to the video stream. This allows the code to be more maintainable. The `Manager` is 'dependent' on the `WebPageControls` class. The `WebPageControls` class is passed to the `Manager` class as a parameter 'injected' during construction. Hence the term dependency injection.

The second parameter to the `Manager` class is the URL to which the video stream is sent. This allows the `Manager` class to be reused for use cases 2 and 3 by passing in the appropriate URL.

#### 4.7.4 Incorporating JavaScript into the web app

The JavaScript resources are saved in the static folder of the web app and accessed via Django static functions. c.f. The Static Files

### 4.8 Building and deploying the Website

The web app, situated within the website folder `identity\_website`, is utilised to showcase content on the website.

#### Linking the App to the Website

To link the app to the website, add the app to the `INSTALLED_APPS` list in the `settings.py` file.

```
1. INSTALLED_APPS = [  
2.     'identity.apps.IdentityConfig',  
3. ]
```

#### Configuring Login Paths

Add the following code in the settings.py to set the login and redirect URLs.

```
1. LOGIN_URL = '/login/'  
2. LOGIN_REDIRECT_URL = '/login/'
```

## Chapter 5 Testing and Results

### 5.1 Introduction

Unit testing was utilised to isolate and test individual code components for this application during the testing process. This approach allowed for the verification of the application's functionality in isolation, making it easier to identify and address any issues that arose.

The OpenCV face recognition algorithm was extensively tested throughout the testing phase to ensure it correctly identified registered users and rejected unauthorised access attempts. The time logging feature was also thoroughly tested to record user entry and exit times accurately.

### 5.2 Unit Testing

#### 5.2.1 What is Unit Testing, and why is important?

Unit testing is a method used in software development to test separate parts or units of an application to make sure they work as expected. Each unit is tested independently, and any dependencies are simulated to ensure that the test only looks at the unit in question. Unit testing is important because it lets developers find bugs early in the development process before they get harder to fix and cost more money. By testing units separately, developers can easily and quickly find problems that might be hard to find with other types of testing. Unit testing also ensures that changes to the codebase don't introduce new bugs or break features already there. Developers can ensure that changes are tested well before being added to the codebase's main branch by using unit tests as part of a continuous integration process. Unit testing also makes it easier to write clearer and easier-to-understand code. By breaking the application into smaller, easier-to-handle pieces, developers can write code that is easier to read, understand, and keep up to date. In conclusion, unit testing is an important part of software development that makes software applications better and more reliable. It can help find bugs early, stop regression, and encourage better coding practices, making software systems more stable and reliable.

#### 5.2.2 Testing Framework in Django

The author utilised the testing framework built into Django for the project, which offers comprehensive support for unit testing. Although the framework also includes functionality for integration and functional testing, only unit testing was employed in this case.

To run the test, we must run the following in the terminal “python\manage.py test”.

### 5.3 Unit Testing Done

Unit testing was used to test the functions of face recognition.

- get\_images\_and\_labels
- detect\_user\_face
- face\_recognition

The unit test “test\_detect\_user\_face” takes an image known to have one face in the image. Lines 7 and 8 assert that the function detect\_user\_face works correctly if is\_face\_present is true, and face is not none.

```
1. def test_detect_user_face(self):
2.     image_path = "identity/tests/secret/image-with-one-face.jpg"
3.     open_cv_image = numpy.array(Image.open(image_path))
4.     gray_scale_image = cv2.cvtColor(open_cv_image, cv2.COLOR_BGR2GRAY)
5.
6.     is_face_present, face = detect_user_face(gray_scale_image)
7.     self.assertTrue(is_face_present)
8.     self.assertIsNotNone(face)
```

### 5.4 Test-Driven Development

TDD was implemented using selenium with Cucumber (Gherkin).

First, a scenario is written using Cucumber. This scenario checks if the locations are loaded correctly, verifying the emails in the HTML table.

```
1. Feature: Sign in activation feature
3. @locations
4. Scenario: 2 default locations exist
5.     Given application is opened
6.     When presented in Locations site
7.     Then locations with email Limerick@identify.com, Cork@identify.com presented
```

behave\_test\features\feature\_files\activate\_sign\_in\activate\_sign\_in.feature

The following Python code files support the above scenario. The code parse the HTML of the locations page so that the scenario can check the page contains the following emails: Limerick@identify.com and Cork@identify in the 3<sup>rd</sup> column of the table.

```

1. label = {
2.     'location_label': '#my-desc',
3.     'sign_in_label': '.container h2'
4. }
5. header = {
6.     'header_links': '.navbar-nav .nav-item .nav-link'
7. }
8. table = {
9.     "content": '.table tbody tr'
10. }
11.
12. def table_rows(position):
13.     return f'.table tbody tr:nth-of-type({position})'
14.
15. def row_positions(row, position):
16.     return f'.table tbody tr:nth-of-type({row}) td:nth-of-type({position})'

```

behave\_test\pom\location\location\_test.py

```

1. from behave_test.pom.Base import Base
2. from behave_test.pom.location.location_test import label, header, table, \
3.     row_positions
4.
5.
6. class Location(Base):
7.     def __init__(self, driver):
8.         super().__init__(driver)
9.
10.    def open_location(self, url):
11.        self.driver.get(url)
12.
13.    def verify_redirection(self):
14.        self.visibility_of_element_presented(label['location_label'])
15.        return self.return_element_value(label['location_label'])
16.
17.    #...
18.
19.    def search_specific_location(self, location):
20.        rows = self.return_locations(table['content'])
21.        for i in range(len(rows)):
22.            source_value = row_positions(i + 1, 3)
23.            if self.return_element_value(source_value) == location:
24.                return True
25.        return False
26.
27.    #...

```

behave\_test\pom\location\Location.py



The location scenario test is then called by general\_steps.py

```
1. from behave import *
2.
3. from behave_test.pom.location.Location import Location
4.
5. use_step_matcher("re")
6.
7.
8. @step("application is opened")
9. def step_impl(context):
10.     context.location = Location(context.driver)
11.     context.location.open_location(f'{context.url}locations')
```

behave\_test\features\steps\general\_steps.py

## 5.5 Result Analysis:

From unit-testing the function “face\_recognition”, the FACE\_CONFIDENCE\_LEVEL was reduced to 85.0 from 95.0 because the recogniser rejected images the human tester deemed acceptable.

## 5.6 Discussion of Findings

This project’s purpose was to develop and learn new skills. A lot of development problems resulted from a lack of knowledge. These problems are overcome by acquiring new skills involving trial and error. Some development time and bugs would have been reduced if the Use Cases in design had been more detailed. e.g., The initial version of the Sign in with facial use-case did not document stopping a sign-on with an incorrect identification. An abort option with a countdown to sign-on and a user identification display had to be added to the app.

## Chapter 6 Conclusion:

It is the opinion of this author the continued development of 'Identity' as a proof of concept holds significant promise for meeting the needs of different businesses with varying levels of fault tolerance in different domains.

### 6.1 Continued development

Before bringing a product like 'Identity' to market, it's important to perform data analysis to ensure its viability. However, as a proof of concept, 'Identity' could be scaled to meet the needs of different businesses with varying levels of fault tolerance in different domains. This flexibility makes 'Identity' a promising solution for businesses looking to enhance their security and authentication processes.

Security Tiers:

1. Low consequence (low): Supermarkets have low consequences for misidentifying a person and granting access to an authorised user.
2. Accidental intrusions (high): The consequences of misidentifying a person and granting access to an authorised user are potentially lethal for hospitals. An unauthorised person wandering around the hospital cannot be permitted by 'Identity' entry to a live sterile ER operating room.
3. Malicious attacks (high): For prisons, the consequences of misidentifying a person and granting access to an authorised user are always assumed to be lethal. Every authentication request is assumed to be potentially malicious (e.g., A prisoner holding up a photo of a guard in front of the 'Identity' station's camera), requiring a higher (confidence value) burden of proof to confirm user access. A prisoner cannot be permitted by 'Identity' entry or unlogged access to any location in prison.

Resources and time will be required for big data analysis and confidence testing to determine the optimal confidence value to use 'Identity' for high-security tier environments (Hospitals, Prisons).

### 6.2 Machine learning approaches

Regarding machine learning approaches, unsupervised learning is a cost-effective, time-efficient option and well-suited for low-tier domains where the consequences of misidentification are relatively low.

On the other hand, supervised learning is a more resource-intensive option better suited for high-security tier domains like those that deal with accidental intrusions or malicious attacks, where accuracy and precision are paramount.

### 6.3 Tracking continuing developments

Liveness detection is a security feature that aims to ensure that a biometric system such as ‘identity’ interacts with a live, real person rather than a static image, video, or mask. It is intended to stop spoofing assaults, in which a perpetrator tries to trick a facial recognition system by utilising a picture, video, or 3D model of the person's face (Saptarshi Chakraborty, 2014).

The war against ‘impersonation and fraud’ is never-ending. A promising development in combating fraud is Generative Adversarial Networks (GANs). GANs are a class of deep learning models that consist of two neural networks, a generator, and a discriminator, which compete against each other in a process known as adversarial training. In the context of biometric security, GANs can enhance biometric systems' performance, robustness, and security by generating synthetic biometric data or improving the detection of spoofing attempts.

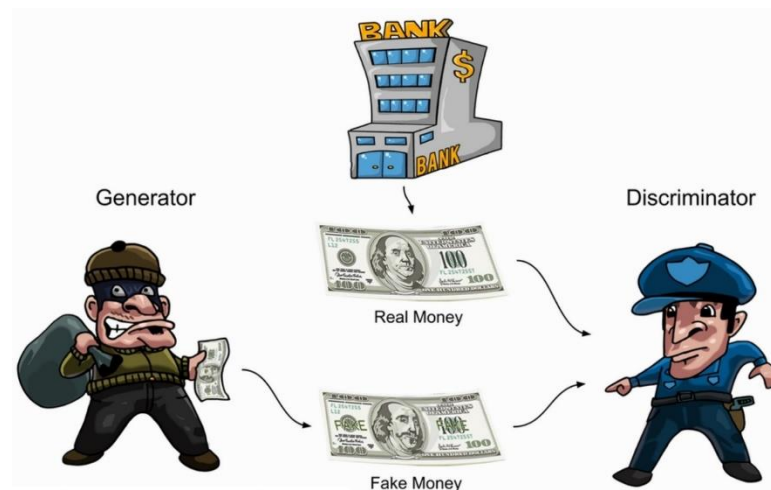


Figure 23: Generator versus discriminator

The science of image recognition will never stop. New and different techniques and challenges will give rise to new solutions. Continuous literature reviews and new solutions will always be required.

## 6.4 Behaviour-Driven Development

At a late stage of the project development, BDD and TDD were introduced and found to be beneficial. For further development of the project, BDD and TDD should be used from the outset to minimize errors, focus on goals, and save development time.

## Chapter 7 Bibliography

Anon., n.d. *Review of Face Recognition Techniques - Pennsylvania State University*.

[Online]

Available at:

<https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.742.1483&rep=rep1&type=pdf>

[Accessed 12 10 2022].

Anon., n.d. *What is Google Cloud Vision?*. [Online]

Available at: [www.resourcespace.com](http://www.resourcespace.com)

[Accessed 25 10 2022].

Beresford, R. & Agatonovic-Kustrin, S., 2000. Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. *Journal of Pharmaceutical and Biomedical Analysis*, 22(5), pp. 717-727.

Çarıkcı, M., 2012. A Face Recognition System Based on Eigenfaces Method. p. 6.

DeepFace, 2022. *Wikipedia*. [Online]

Available at:

<https://en.wikipedia.org/wiki/DeepFace#:~:text=DeepFace%20is%20a%20deep%20learning>

[Accessed 23 10 2022].

Dertat, A., 2017. *towardsdatascience*. [Online]

Available at: <https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2>

[Accessed 4 November 2022].

FindBiometrics, 2022. *Biometric Facial Recognition - FindBiometrics*. [Online]

Available at: <https://findbiometrics.com/solutions/facial-recognition/>

[Accessed 30 September 2022].

GeeksforGeeks, 2021. *GeeksforGeeks Multi-Layer Perceptron Learning in Tensorflow*.

[Online]

Available at: <https://www.geeksforgeeks.org/multi-layer-perceptron-learning-in-tensorflow/>

[Accessed 1 11 2022].

- Goyal, K., Agarwal, K. & Kumar, R., 2017. *ieeexplore*. [Online]  
Available at: <https://ieeexplore.ieee.org/document/8203730/authors#authors>  
[Accessed 6 November 2022].
- Jain, A. K., Mao, J. & Mohiuddin, K., 1996. Artificial neural networks: a tutorial. *IEEE Computer*, , 29(3), pp. 31-44.
- Karamizadeh, S. et al., 2013. *Journal of Signal and Information Processing*. [Online]  
Available at: [https://file.scirp.org/pdf/JSIP\\_2013101711003963.pdf](https://file.scirp.org/pdf/JSIP_2013101711003963.pdf)  
[Accessed 31 10 2022].
- Khan, F., 2018. Facial Expression Recognition using Facial Landmark Detection and Feature Extraction via Neural Networks. *ArXiv*, Volume abs/1812.04510, p. 7.
- King, D. E., 2009. *jmlr*. [Online]  
Available at: <https://www.jmlr.org/papers/volume10/king09a/king09a.pdf>  
[Accessed 6 November 2022].
- Kristof, T., 2002. *researchgate*. [Online]  
Available at:  
[https://www.researchgate.net/publication/283463205\\_A\\_mesterseges\\_neuralis\\_halok\\_a\\_jovokutatas\\_szolgalataban\\_Artificial\\_neural\\_networks\\_in\\_Futures\\_Studies](https://www.researchgate.net/publication/283463205_A_mesterseges_neuralis_halok_a_jovokutatas_szolgalataban_Artificial_neural_networks_in_Futures_Studies)  
[Accessed 1 11 2022].
- McCulloch, W. S. & Pitts, W., 2014. *Cambridge Core*. [Online]  
Available at: <https://www.cambridge.org/core/journals/journal-of-symbolic-logic/article/warren-s-mcculloch-and-walter-pitts-a-logical-calculus-of-the-ideas-immanent-in-nervous-activity-bulletin-of-mathematical-biophysics-vol-5-1943-pp-115133/7DFDC43EC1E5BD05E9DA85E1C41>  
[Accessed 1 11 2022].
- Olah, C., 2015. *GitHub Blog*. [Online]  
Available at: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>  
[Accessed 4 November 2022].
- Pocket-lint, 2018. *Pocket-lint*. [Online]  
Available at: <https://www.pocket-lint.com/phones/news/apple/142207-what-is-apple-face-id-and-how-does-it-work>  
[Accessed 23 10 2022].

Saptarshi Chakraborty, D. D., 2014. An Overview of Face Liveness Detection. *International Journal on Information Theory*, 14 04, 3(2), pp. 11-25.

Shanu, S., n.d. *InsideAIML*. [Online]

Available at: <https://www.insideaiml.com/blog/Activation-Functions-In-Neural-Network-1089>

[Accessed 1 11 2022].

Solanki, K., 2016. Review of Face Recognition Techniques. *Review of Face Recognition Techniques*, p. 5.

Taigman, Y., Yang, M., Ranzato, M. & Wolf, L., 2014. *DeepFace: Closing the Gap to Human-Level Performance in Face Verification*. [Online]

Available at: [http://cs.toronto.edu/~ranzato/publications/taigman\\_cvpr14.pdf](http://cs.toronto.edu/~ranzato/publications/taigman_cvpr14.pdf)

[Accessed 23 10 2022].

Tang, Z. & Fishwick, P. A., 1993. *Researchgate*. [Online]

Available at:

[https://www.researchgate.net/publication/220668844\\_Feedforward\\_Neural\\_Nets\\_as\\_Models\\_for\\_Time\\_Series\\_Forecasting](https://www.researchgate.net/publication/220668844_Feedforward_Neural_Nets_as_Models_for_Time_Series_Forecasting)

[Accessed 3 November 2022].

Team, K., n.d. *Keras documentation: Backend utilities*. [Online]

Available at: [https://keras.io/api/utils/backend\\_utils/](https://keras.io/api/utils/backend_utils/)

[Accessed 01 12 2022].

TensorFlow, 2019. *TensorFlow*. [Online]

Available at: <https://www.tensorflow.org>

[Accessed 01 12 2022].

Viola, P. & M.Jones, 2003. *ieeexplore*. [Online]

Available at: <https://ieeexplore.ieee.org/document/990517>

[Accessed 6 November 2022].

Vision?, What is Google Cloud, n.d. *www.resourcespace.com*. [Online]

Available at: <https://www.resourcespace.com/blog/what-is-google-vision>

[Accessed 25 10 2022].

Wynne, M. H. A. & T. S., 2017. *The Cucumber Book: Behaviour-Driven Development for Testers and Developers*. s.l.:Pragmatic Bookshelf.

Yi, S., Ding, L., Xiaogang , W. & Xiaoou , T., 2015. *researchgate*. [Online]

Available at:

[https://www.researchgate.net/publication/271855676\\_DeepID3\\_Face\\_Recognition\\_wit](https://www.researchgate.net/publication/271855676_DeepID3_Face_Recognition_with_Very_Deep_Neural_Networks)  
[h\\_Very\\_Deep\\_Neural\\_Networks](https://www.researchgate.net/publication/271855676_DeepID3_Face_Recognition_wit)

[Accessed 03 November 2022].