National College of Ireland
BSc in Computing
2017/2018

Aaron Beveridge
X14469052
X14469052@student.ncirl.ie

Looking Glass

Technical Report

Contents

Executive Summary

This document outlines the development and testing process for a home security system called Looking Glass. Looking Glass is disguised as a standard home mirror. This is to prevent intruders from noticing the user's security system and possibly disabling it.

Looking Glass is secured by a two-step authentication system which comprises of facial recognition and a gesture pattern unlock. The gesture pattern unlock combines object recognition and the traditional touch-screen pattern unlock. Users have the ability to draw a pattern in the air using their hand which is recorded by Looking Glass. This recording is then compared to a stored pattern.

Looking Glass monitors sensors which can be attached to doors and windows around a user's home. If a sensor is triggered, a notification is displayed on the Looking Glass

1 Introduction

The aim of this document is to provide all the technical details of the Looking Glass security system project. The system provides a user an insight to the status of sensors located around the user's home. All features and functions are described in the document below.

1.1 Background

I have always been fascinated with the possibilities of the RaspberryPi. One project which I particularly admired was the "Magic Mirror", which essentially comprises of a RaspberryPi, an LCD screen and a sheet of two-way glass. When text is displayed on the monitor, it shines through the glass and appears on the reflective side. Typically, magic mirrors display information such as live Twitter feeds, local weather or traffic and Google Calendar reminders.

My inspiration for this project came from a conversation I had with my Uncle. My uncle was telling me about his recent break in. Thieves broke into his home and disabled the alarm box before it could alert authorities. If the thieves were unable to locate the alarm box, they would have been caught. This gave me the idea to combine the "Magic Mirror" concept with a home security system. Only the home owners would know that the mirror contains the security system as it would not display anything unless activated.

Once I complete Looking Glass, I would like to partner with the Gardai or companies who offer home monitoring services such as Eir. Rather than just alerting the user about a sensor being triggered, I would like to send a notification to the Gardai or the home monitoring service. This will give users peace of mind if they are leaving their home vacant for an extended period of time.

1.2 Aims

The aim of Looking Glass is to provide users with a disguised home security system.

Looking Glass will be disguised as an ordinary mirror. Only a registered user will be able to access the system through a two-step authentication process. This two-step process will be made up of a facial recognition and a gesture pattern unlock.

Facial recognition will try to match the users face with a stored image of the any of the registered users.

The gesture pattern unlock is similar in concept to a standard touch-screen pattern unlock. The gesture pattern unlock will recognize a user's hand and track the pattern a user makes with their hand in the air. This pattern will be compared to a stored pattern.

Once unlocked, Looking Glass will give a user an overview of all connected sensors, showing their name, open/closed status and location.

On start-up, Looking Glass will check if any users are registered and if there is a stored pattern. If either of these are not detected, the user will be walked through the set-up process for the missing resource.

If Looking Glass does not detect a user's face for a period of time it will revert to a locked state.

1.3 Technologies

Looking Glass will be controlled by a Raspberry Pi. A Raspberry Pi is a small computer, roughly the size of a credit card. They are typically used for computer science or electronics projects or by enthusiasts who want to learn more about computing.

Both forms of recognition will use OpenCV and Python. OpenCV is a computer vision and machine learning library for Python, Java and C++.

Facial recognition will be achieved by taking a large sample of images of the users face and running them through a method which learns the features of the supplied

images. When a user tries to unlock Looking Glass, their face will be scanned and compared against the registered users.

Gesture recognition will work in a similar fashion to facial recognition. Looking Glass will record the movements made by a user's hand. This recording will be compared to a stored pattern. Looking Glass will unlock if the compared items are in a high similarity range.

The sensors that will be attached to doors or windows will be reed sensors. These sensors contain two parts, a magnet and sensor. The sensor is attached to the frame of the door or window while the magnet is attached to the moving door or window. When the door or window is opened, a circuit is broken. These sensors can be attached directly to the GPIO pins of a Raspberry Pi.

1.4 Structure

The reminder of this report is broken down as follows. Part 2 – System gives a complete breakdown of how the Looking Glass will be developed, tested and evaluated. The sections include:

- **Requirements:** This is further broken down to functional and non-functional requirements of the Looking Glass system as a whole.
- **Design and Architecture:** A look into how Looking Glass is designed and how it functions.
- **Implementation:** This section talks about the different algorithms, classes and methods used within the Looking Glass system.
- **Graphical User Interface (GUI) Layout:** GUI Layout includes mock-ups of the user interface and an explanation of each mock-up.
- **Testing:** In testing, I talk about the different methods of testing which will be implemented.
- **Customer Testing:** This section will show evidence for and results from customer testing.
- **Evaluation:** The evaluation section discusses methods of evaluation and their results.

Part 3 – Conclusions describes the advantages, disadvantages, opportunities and limits of Looking Glass.

Part 4 – Further development or research. Here I will discuss what I could improve upon in the final deliverable of Looking Glass.

2 System

2.1 Requirements

2.1.1 Functional requirements

2.1.1.1 Requirement 1: Gesture Pattern Unlock

2.1.1.1.1 Description & Priority

Object recognition will be the backbone of the gesture pattern unlock system. Looking Glass will be taught to recognize a user's open hand. This requirement is crucial for the development of the two-factor unlock system, therefore it is being ranked as the most important requirement.

2.1.1.1.2 Use Case

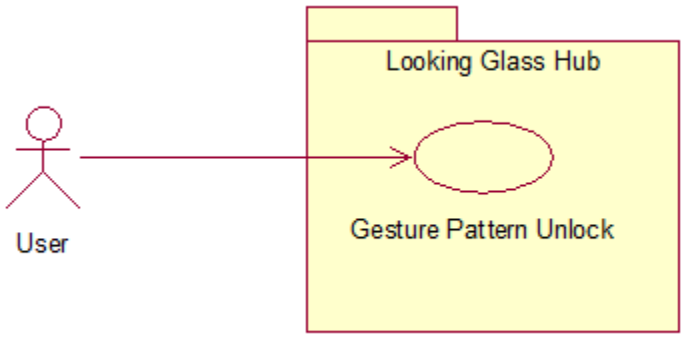
Scope

The scope of this use case is to implement an object recognition software which can detect a user's hand and track its movements. The system should then be able to compare this movement to a stored pattern and verify if both the stored pattern and movement is similar.

Description

This use case describes the gesture pattern unlock mechanism.

Use Case Diagram



Flow Description

Precondition

The system is in the second step of the two-step unlock mechanism.

Activation

This use case starts when a user has passed the facial recognition unlock.

Main flow

1. The system identifies the users hand. (See A1)
2. The user draws their pattern in the air using their hand while the system records the pattern. (See A2)
3. The system compares the user's pattern with a predefined pattern and allows the user to access the Looking Glass main screen (See A3)

Alternate flow

A1: The system cannot identify the users hand.

1. The system will revert to a lock state after 10 seconds if no hand is detected.
2. The use case terminates.

A2: The user does not draw their pattern.

1. The system will revert to a lock state after 10 seconds if no hand is detected.
2. The use case terminates.

A3: The system cannot identify the users hand.

1. The use case terminates if no correct pattern is supplied.

Termination

The system presents the main screen.

Post condition

The system goes into a wait state

2.1.1.2 Requirement 2: Facial Recognition Unlock

2.1.1.2.1 Description & Priority

The facial recognition unlock mechanism will detect a user's face while they stand facing the Looking Glass. If the stored user's facial image matches with the presented face, the system will unlock. As the facial recognition unlock will form part of the two-factor unlock mechanism, it is being ranked high on the list of functional requirements.

2.1.1.2.2 Use Case

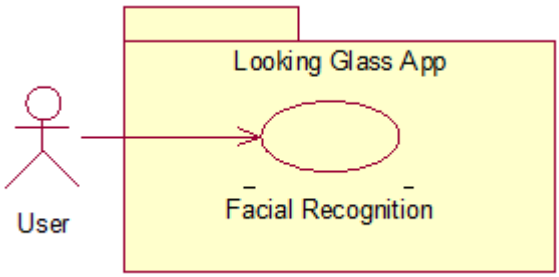
Scope

The scope of this use case is to implement a facial recognition mechanism. This mechanism should recognise a human face and compare it to a stored image of the user's face. If the images match, the system will unlock.

Description

This use case describes the facial recognition unlock mechanism.

Use Case Diagram



Flow Description

Precondition

The system is in a lock state.

Activation

This use case starts when a user stands in front of the Looking Glass.

Main flow

1. The user allows the system to scan their face (See A1)
2. The system confirms the identity of the user based on the scan (See A2)
3. The system progresses to the next stage of authentication.

Alternate flow

A1: The user does not allow the system to scan their face.

1. The system remains in a lock state
2. The use case terminates

A2: The system cannot confirm the user's identity.

1. The system remains in a lock state
2. The use case terminates

Termination

The system presents the next authentication stage.

2.1.1.3 Requirement 3: Two-Factor Unlock

2.1.1.3.1 Description & Priority

The two-factor unlock system will comprise of the facial recognition and gesture pattern unlock mechanisms. The two-factor unlock system will be the main method of keeping the information in Looking Glass secure. Because of this, it is one of the highest priority functional requirements.

2.1.1.3.2 Use Case

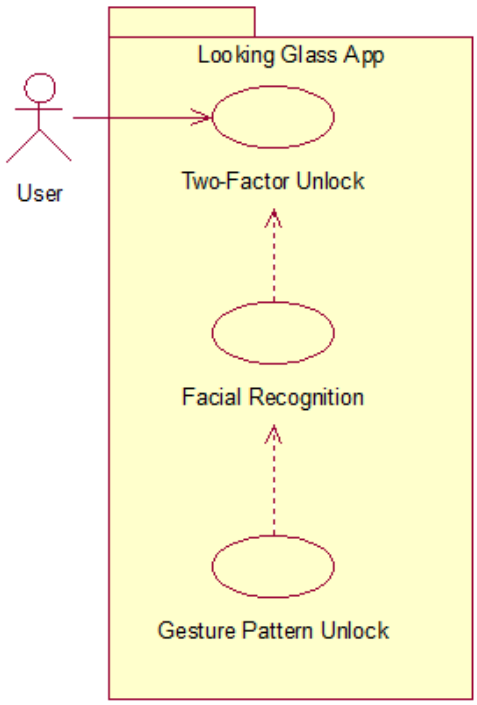
Scope

The scope of this use case is to combine both the gesture pattern unlock and facial recognition unlock mechanisms into one unlock system.

Description

This use case describes the two-factor unlock system.

Use Case Diagram



Flow Description

Precondition

The system is in a lock state.

Activation

This use case starts when a user stands in front of the Looking Glass.

Main flow

1. The user successfully completes the Facial Recognition use case. (See A1)
2. The system progresses to the Gesture Pattern Unlock method.
3. The user successfully completes the Gesture Pattern Unlock use case. (See A2)
4. The system allows the user to access the Looking Glass main screen.

Alternate flow

A1: The user fails the Facial Recognition use case.

1. The system remains in a lock state.
2. The use case terminates.

A2: The user fails the Gesture Pattern Unlock use case.

1. The system reverts to a lock state.
2. The use case terminates.

Termination

The system presents the main screen.

Post condition

The system goes into a wait state.

2.1.1.4 Requirement 4: Self-Lock Mechanism

2.1.1.4.1 Description & Priority

Looking Glass will be unlocked by the two-factor unlock mechanism. The system will lock itself once the it detects that the user has stopped using the system. This lock mechanism is not considered as a high priority requirement.

2.1.1.4.2 Use Case

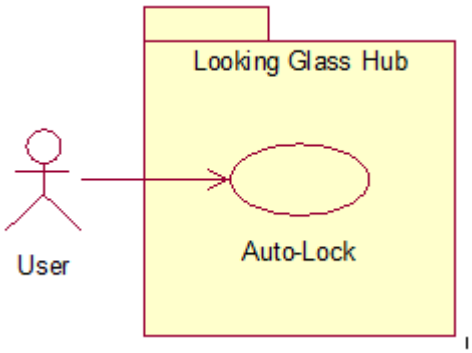
Scope

The scope of this use case is to develop a method which will detect when a user has stopped using the system. The method will then lock the system once this method has been triggered.

Description

This use case describes the self-lock method.

Use Case Diagram



Flow Description

Precondition

The system has been unlocked by the user.

Activation

This use case starts when a user turns away from the Looking Glass.

Main flow

1. The system will continue to detect the user's face.
2. The user will finish using Looking Glass.
3. The system will detect that the user's face is no longer directed at the screen.
4. After a 10 second timer has elapsed, the system will revert to a lock state

Post condition

The system goes into a lock state.

2.1.1.5 Requirement 5: Display Status

2.1.1.5.1 Description & Priority

Once a user has unlocked Looking Glass, it will display the status of all connected sensors. This function is to help a user check if they have forgotten to lock a door or window before they leave their home. As this is not as important as the other requirements, it is ranked as a low priority requirement.

2.1.1.5.2 Use Case

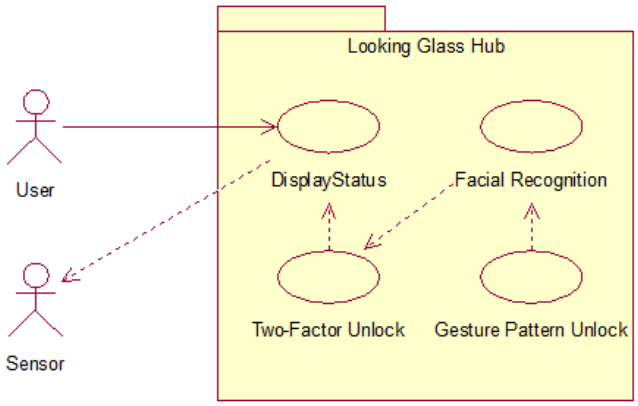
Scope

The scope of this use case is to develop a method to display the status of all sensors connected to Looking Glass.

Description

This use case describes the display status method.

Use Case Diagram



Flow Description

Precondition

At least one sensor has been linked to the system.

Activation

This use case starts when a user has unlocked the system.

Main flow

1. The system pulls information from the sensors.
2. The system displays this information on screen.
3. The system enters a wait state.

Post condition

The system goes into a wait state.

2.1.2 Non-Functional Requirements

2.1.2.1 Performance/Response time requirements

Looking Glass will operate on a Raspberry Pi and have no dependencies on external systems or services. Looking Glass does not need an internet connection. Because of this, factors such as network speeds and connectivity issues do not need to be considered. The sensors will need to have a high accuracy to avoid false signals being sent to the Looking Glass.

2.1.2.2 Availability requirements

As Looking Glass is a home security system, it should be consistently available. Once a user has completed the initial set up process the system should be fully functional and available to the user. Any system downtime should be minimal and based mainly on the user's home power supply. Any other bugs or issues should be patched or removed as soon as possible to ensure full availability to the user.

2.1.2.3 Operating System Requirements

As the application will be developed for a Raspberry Pi, The Raspberry Pi will need to be running the latest version of Raspbian. Looking Glass will use Raspbian Jessie.

2.1.2.4 Recovery requirements

In the event of a system failure, a recovery process should be followed by the user. The user should simply have to restart the Looking Glass to resume its services. This recovery process should take no longer than five minutes

2.1.2.5 Usability requirements

Looking Glass will have a very simple user interface. Looking Glass will only be accessed by passing the two-factor unlock. There will be no other interaction such as touch controls or mouse and keyboard input.

Once Looking Glass has been unlocked, it will display the status of the connected sensors as long as it detects a face looking at the screen.

So that the user interface will display properly on the reflective side of the mirror, the colour scheme of the user interface will need to comprise of bright colours.

The initial supported language will be UK English.

2.1.2.6 Security requirements

Looking Glass will maintain the following security principles: Confidentiality, Integrity and Authenticity.

Confidentiality: To ensure only authorised user accesses the stored information, the two-factor unlock will be implemented. As an additional security measure, the system will lock if it detects that the user has stopped looking at the screen.

Integrity: To ensure integrity is maintained, data cannot be tampered with by unauthorised users. All information stored on the system will be encrypted.

Authenticity: To make sure only registered users can perform actions on the system, the two-factor login will be implemented. This unlock relies on something a user is (the facial recognition) and something a user knows (the gesture pattern unlock). Only the intended user will be able to pass both security checks as both checks are unique to a user.

2.1.2.7 Reliability requirements

Only minimal down-time will be accepted. Only one factor should cause an outage, a power outage. As this is an external factor, very little to no down-time is to be expected. If an outage does occur, recovery time will be dependent on a user reconnecting to their power source.

2.1.2.8 Maintainability requirements

Looking Glass will need to be maintained.

Updates for the hub will be pushed directly to the hub as to keep it constantly up to date.

All updates for Looking Glass will be developed with the previously mentioned security standards in mind.

2.1.2.9 Compatibility requirements

There will be no need for Looking Glass to communicate with any other external systems. The sensors will be only compatible with doors and windows that open outwards. This is so that the sensors are not accidentally damaged.

2.1.2.10 Environment Requirements

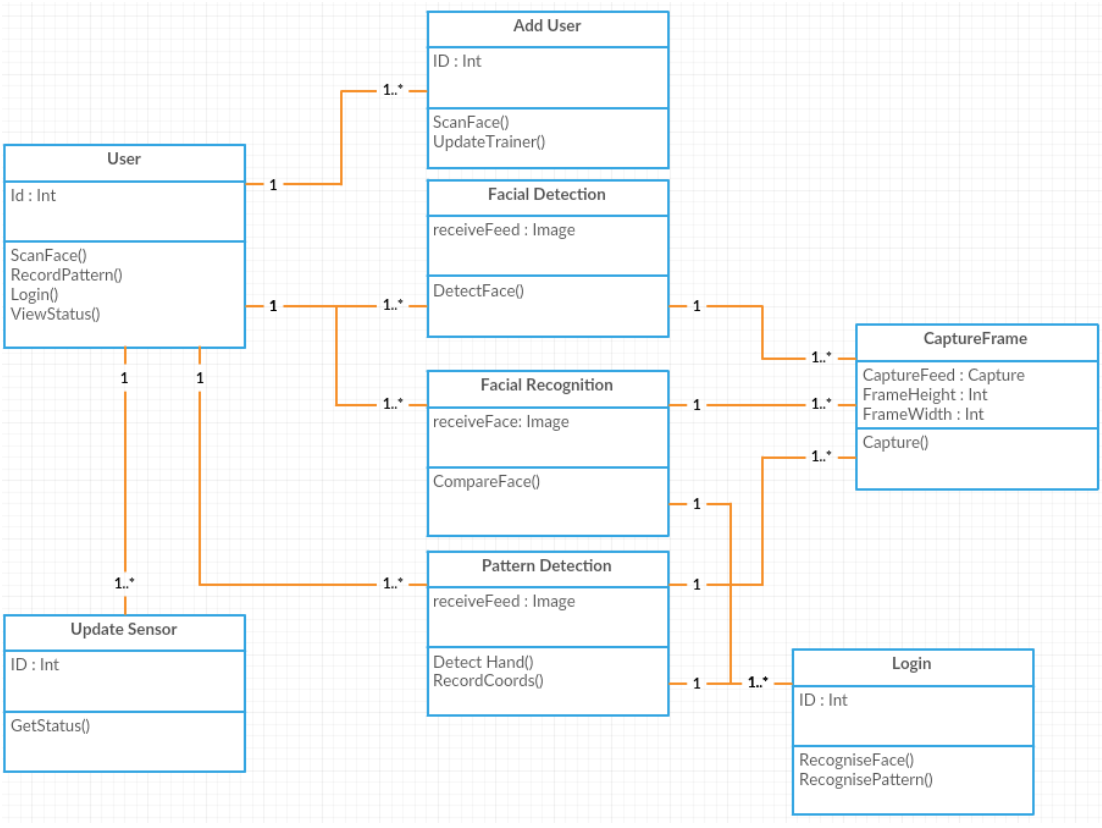
To use Looking Glass optimally, it will need to be in a well-lit room. This is to ensure that the facial and gesture recognition functions work efficiently.

2.2 Design and Architecture

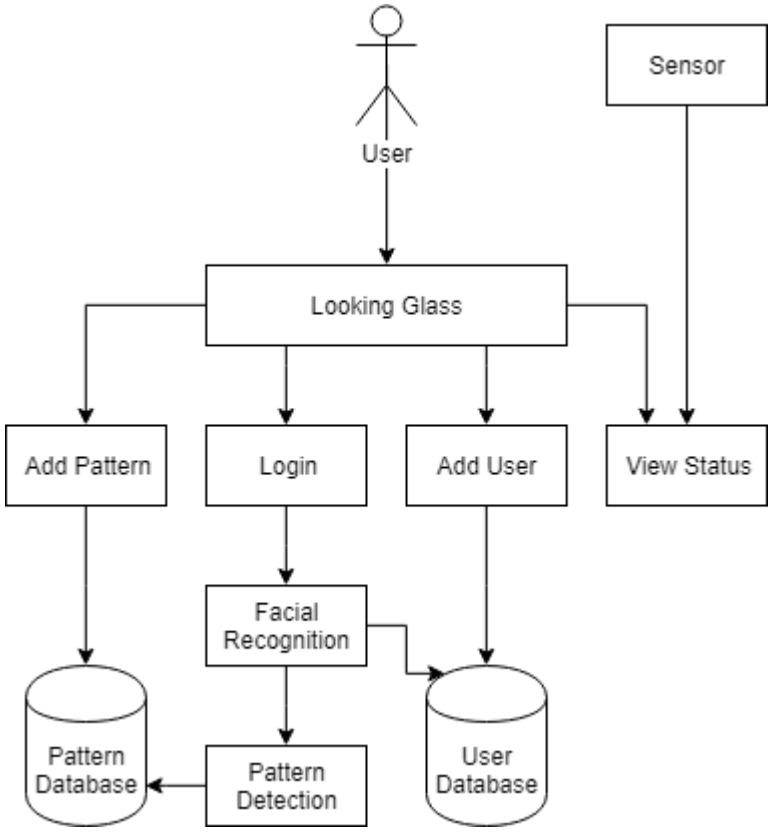
Looking Glass will be written in Python. The recognition functions will rely on OpenCV. Looking Glass will be comprised of three elements, the user, the sensors and the Looking Glass system.

Looking Glass will receive information from the sensors on a near constant basis, while the user will only interact with Looking Glass to unlock it, add a new user or to add a new pattern.

2.2.1 Class Diagram



2.2.2 Use Case Diagram



The above image depicts the use case diagram of Looking Glass.

2.3 Implementation

The purpose of this section is to describe the technologies used in the development of Looking Glass.

2.3.1 Technology Overview

I developed Looking Glass in Python using the Geany text editor. Both the facial recognition and pattern recognition functions have been built upon OpenCV. OpenCV uses Haar Cascades to achieve recognition.

Haar cascade is an object detection algorithm used to detect simple shapes in an image. A cascade is trained by supplying large quantities of positive (images

containing faces or hands in this case) and negative images (images that do not contain either).

I build a responsive user interface using QT Designer for Python.

2.3.2 Security

Security is the most important feature of Looking Glass and was implemented wherever possible. The main security features are the facial recognition and pattern recognition functions contained within the two-factor login system as well as the auto lock function.

2.3.2.1 Facial Recognition

Facial recognition is achieved using Haar Cascade files and OpenCV. OpenCV captures each frame of the attached camera and converts them to greyscale to improve accuracy. These grey frames are passed though a recognizer which is trained to recognize the faces of registered users.

For every face detected in the frame, a blue square is drawn around it. This is for registered and non-registered users alike.

If Looking Glass recognises a registered users face, their user number and confidence percentage is printed above their blue square.

Looking Glass will display the login screen required for Pattern recognition once a registered user has been found.

```
###Captures frame from video feed
ret, frame = capture.read()
#converts frame to grayscale as cv2 uses grayscale for most opperations
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(gray, scaleFactor=1.5, minNeighbors=5)
for (x, y, w, h) in faces:
    #region on interest
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = frame[y:y+h, x:x+w]

    ####facial recognition
    id_, conf = recognizer.predict(roi_gray)
    if conf>=45 and conf <=85:
        print(id_)
        print(labels[id_]+" "+repr(conf)+"%")
        font = cv2.FONT_HERSHEY_SIMPLEX
        name = labels[id_]
        color = (255, 255, 255)
        stroke = 2
        percent = int(round(conf))
        cv2.putText(frame, trueName + " " + repr(percent) + "%", (x,y), font, 1, color, stroke, cv2.LINE_AA)
        Login.show()
        MainWindow.hide()
```

2.3.2.2 Pattern Recognition

Pattern recognition works in a similar way to facial recognition in that frames are converted to grayscale and passed to a recognizer. When a hand is recognised the X and Y coordinates of the hand are written to a csv file.

```
for (x,y,w,h) in hands:

    #draws green rectangle 2px thick around all positive matches
    cv2.rectangle(img, (x,y), (x+w, y+h), (0,255,0), 2)
    #creates csv writer object
    thewriter = csv.writer(log)
    #writes x,y values to new row in log.csv
    thewriter.writerow([x,y])
```

Once a timer has elapsed, the contents of this csv file are compared to a separate csv file which contains the pre-recorded pattern. The files are compared on the grounds of length and values.

```
with open("gesture.csv","r") as master:
    passwordreader = csv.reader(master,delimiter = ",")
    passworddata = list(passwordreader)
    password_row_count = len(passworddata)

with open("log.csv","r") as newInput:
    reader = csv.reader(newInput,delimiter = ",")
    data = list(reader)
    row_count = len(data)

row_count_Float = float(row_count)
password_row_count_Float = float(password_row_count)
percentage = float(row_count_Float/password_row_count_Float)*100
print "Saved pattern rows: "+repr(password_row_count)
print "Inputted rows: "+repr(row_count)
print "Similar length percentage: "+repr(percentage)

if percentage<40:
    print "Percentage is too low"

##### checks contents #####

with open("gesture.csv","r") as gesture:
    gesture_indices = dict((r[1], i) for i, r in enumerate(csv.reader(gesture)))

positiveCount = 0
negativeCount = 0
with open("log.csv","r") as UserInput:
    with open("results.csv", "wb") as results:
        reader = csv.reader(UserInput)
        writer = csv.writer(results)

        writer.writerow(next(reader, []) + ["RESULTS"])

        for row in reader:
            index = gesture_indices.get(row[1])
            if index is not None:
                message = "FOUND in saved pattern (row {})".format(index)
                positiveCount = positiveCount+1
            else:
                message = "NOT FOUND in saved pattern"
                negativeCount = negativeCount+1
            writer.writerow(row + [message])
```

If the user has entered a sufficiently similar pattern, Looking Glass will display the main status screen and will end the pattern recognition function.

```
with open("results.csv","r") as results:
    readerresults = csv.reader(results,delimiter = ",")
    dataresults = list(readerresults)
    row_countresults = len(dataresults)

print "Rows in results: "+repr(row_countresults)
print "Positive results: "+repr(positiveCount)
print "Negative results: "+repr(negativeCount)
if positiveCount>negativeCount:
    print "Gesture accepted"
    Status.show()#
    MainWindow.hide()
    Login.hide()
    NewUser.hide()
    NewPattern.hide()
    logPattern = False
    log.close()
    master.close()
    newInput.close()
    loggedIn = True
    idleTimer = 0
```

2.3.2.3 Auto Lock

To ensure that only registered and verified users access Looking Glass, I have implemented an auto lock function. For every frame when a registered user's face is not detected, an idleTimer increments by 1. Once the timer reaches 100, all functions stop and the system returns to an idle state where it waits for a registered user to interact. This timer can be reset to 0 by a user interacting with Looking Glass.

```
cv2.imshow("facial detection",securedFrame)
#when no face present
idleTimer = idleTimer+1
print("Face Not Detected after login" + repr(idleTimer))
if idleTimer >= 100:
    print("Idle timer hit limit")
    print("Looking Glass locked to prevent unauthorized access")
    idleTimer = 0
    MainWindow.show()
    Login.hide()
    Status.hide()
    NewUser.hide()
    NewPattern.hide()
    loggedIn = False
    logPattern = False
    log.close()
    master.close()
    newInput.close()
    break
```

2.3.3 Technologies

2.3.3.1 Python

Looking Glass could have been developed using Java, but I ultimately chose to use Python as I have never used Python before and I wanted to be challenged by learning a new language. Python was used to access the GPIO pins of the Raspberry Pi for the sensors, combined with OpenCV it enabled the recognition functions. It was also used to create the GUI.

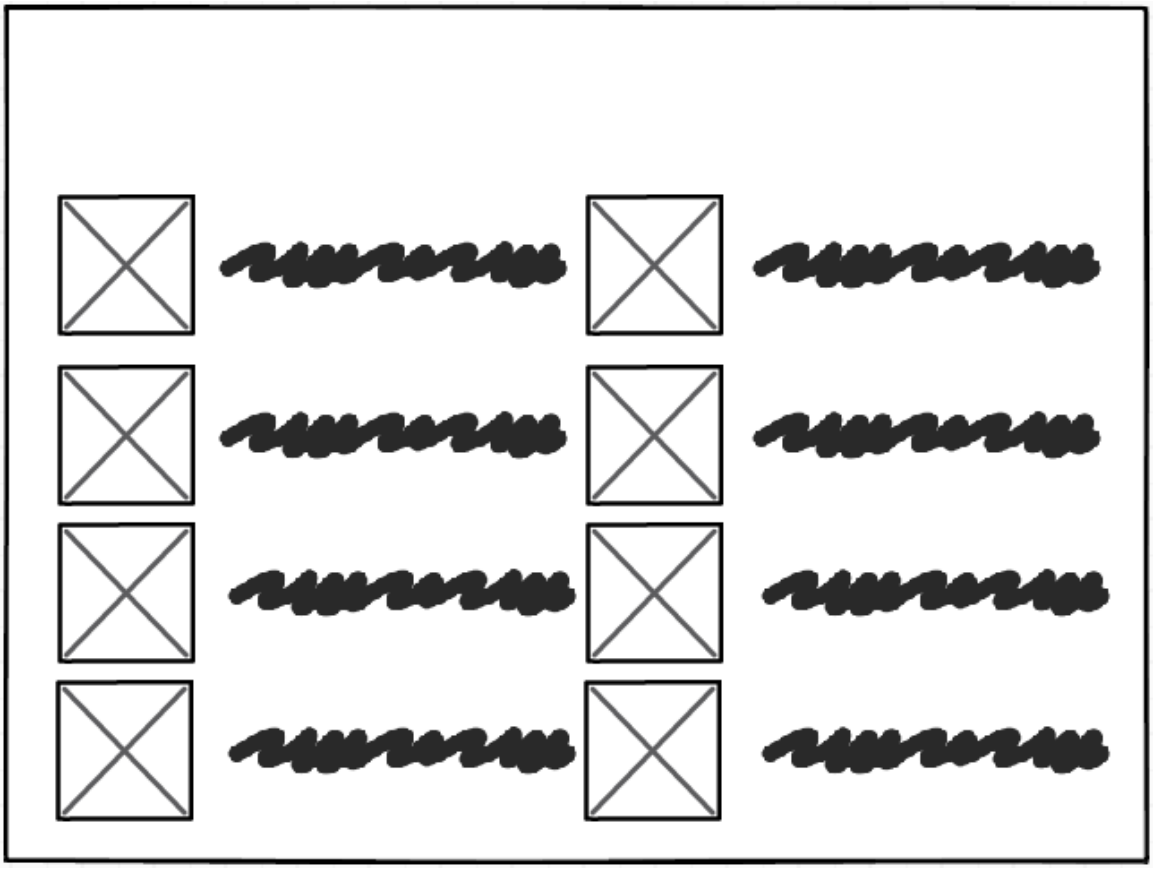
2.3.3.2 OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning library. It provides the ability to access and capture the input from a USB webcam and identify specified object from the captured input.

OpenCV also introduces the ability to create and train new Haar Cascade files from a large sample of images. From this I was able to create a cascade which can identify a user's hand.

2.4 Graphical User Interface (GUI) Layout

The below images are mock-ups of the Looking Glass main screen. Each sensor will be displayed in an organised list along with its name and current status.



So that Looking Glass can display its information correctly through the mirror, only bright colours can be used. The black areas represents the reflective surface of the mirror.

2.5 Testing

As Looking Glass will be a security system, there will have to be a lot of testing to ensure it is safe to use in a home environment.

I will need to test the following aspects of Looking Glass:

- Facial Recognition
- Pattern Unlock
- Door/Window Sensors
- System Integrity

Facial Recognition:

I will need to minimise the risk of false positive results. This could be achieved through user testing. A possible test would be to get a group of people to try and unlock the Looking Glass. I will record the outcome of all interactions.

Pattern Unlock:

Similar to facial recognition, I will need to minimise the risk of false positive results. This will be tested in the same way.

Door/Window Sensors:

I will need to ensure the sensors stay connected to the main system. I will monitor their connection status over a given period of time.

The sensors also need to only react to when a window or door is actually open. I will need to thoroughly test that the sensors are not triggered by external sources.

To ensure that Looking Glass is fit for purpose, it must pass an overall System Test.

2.6 Customer testing

As part of customer testing, volunteers were invited to try out Looking Glass. At the beginning of the testing I gave a short overview as to how Looking Glass operates. I then invited the volunteers to try the system for themselves. Once they were finished using Looking Glass, I asked them to complete a short survey which is attached int the appendices below. The breakdown of results can be seen in the table below.

Customer Testing Results														
User	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**Total**	**Average**
User Interface	2	2	3	3	3	2	4	3	3	4	2	4	**35**	**2.92**
Ease Of Use	3	3	4	4	4	3	4	3	2	3	3	2	**38**	**3.17**
Facial Recognition Function	4	4	4	4	3	4	4	2	2	3	3	2	**39**	**3.25**
Gesture Recognition Function	2	2	1	1	1	2	1	2	2	1	1	2	**18**	**1.50**
Physical Appearance	3	4	5	5	4	5	4	3	5	3	4	3	**48**	**4.00**
Total User Score	**14**	**15**	**17**	**17**	**15**	**16**	**17**	**13**	**14**	**14**	**13**	**13**		
Average User Score	**2.8**	**3**	**3.4**	**3.4**	**3**	**3.2**	**3.4**	**2.6**	**2.8**	**2.8**	**2.6**	**2.6**		

The volunteers main comment about the interface was that it is quite simple, and they would like it to be more eye catching. With an average score of 2.92 / 5 it shows that the users found this element to be acceptable at best.

After being shown how to use the system, most volunteers agreed that Looking Glass is a simple to use system. The downfall however was the incomplete gesture recognition function. The average score for this was 3.17 / 5.

The volunteers seemed to enjoy the facial recognition aspect of Looking Glass. This was the second highest scoring item from the testing. One volunteer

suggested including the ability to recognise a face if the user is wearing over-ear headphones or a hood. Facial recognition scored a 3.25 / 5.

Gesture recognition scored the lowest on the survey with only 1.5 / 5. This was largely due to the function not being fully completed. As a concept, the volunteers liked the idea of using a pattern to unlock the system over traditional methods.

Physical appearance was the highest scoring element of the survey with a score of 4 / 5. The volunteers really liked the novelty of a screen built into a mirror.

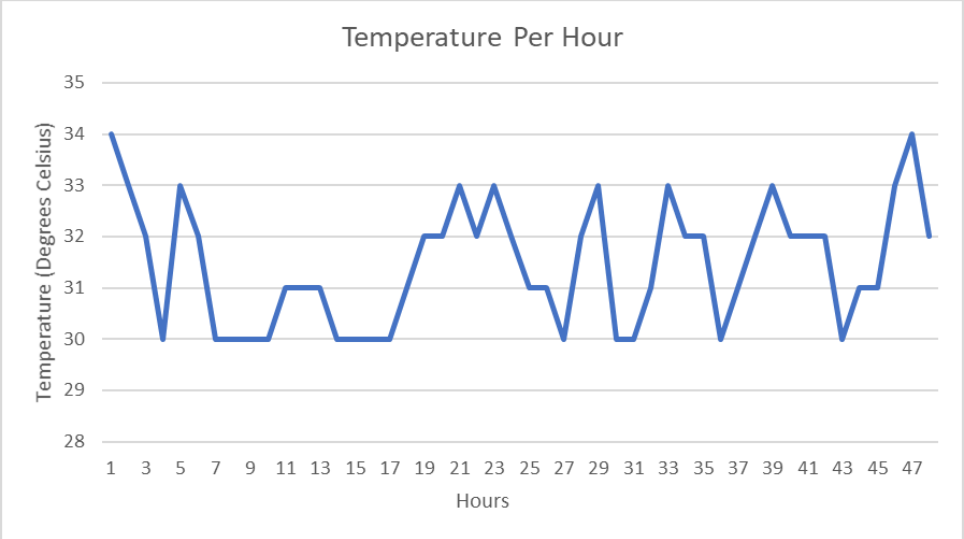
Overall, the volunteers gave a collective average score of 2.97 / 5. I believe this score would be higher if a feature complete system was demonstrated rather than the prototype.

2.7 Evaluation

System evaluation was carried out by monitoring the results of the sensors and the temperature of the RaspberryPi.

The onboard temperature of the Raspberry Pi was recorded once an hour over the course of 48 hours while Looking Glass was left in an idle state. As shown in the below graph, the temperature stayed between 30°C and 34°C for the 48-hour period. As there were no alarming fluctuations or extremely high temperatures, I am comfortable with passing Looking Glass in this test.

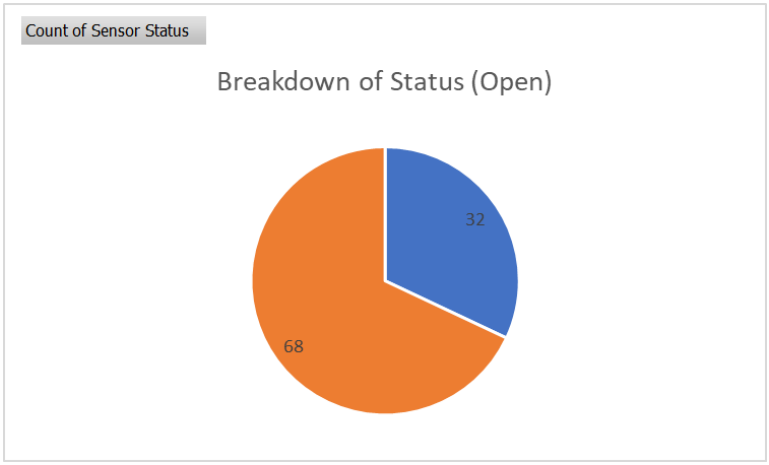
Table 1: Temperature per Hour While Idle



To test the efficiency of the sensors, I ran two tests which recorded the state of the sensor every second for 100 seconds. In the first test I left the sensor in an opened state.

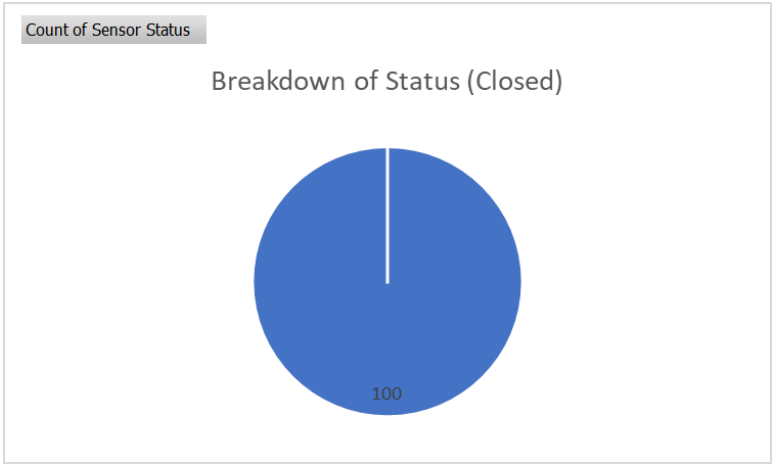
As we can see below, the sensor has a 68% accuracy rating while in an open state. This result is disappointing as it would have been preferable for the sensors to have a higher accuracy. Because of this data cleansing will have to be introduced to combat the false negative results.

Figure 2: Sensor Status Results While Open



The second test was setup to show the sensor in a closed state for 100 seconds. This test was a complete success as the sensor had a 100% accuracy score in this test.

Figure 3: Sensor Status Results While Closed



3 Conclusions

Advantages: The user interface of Looking Glass was made to be responsive so that it can be displayed on a screen of any size. This will allow users to customize the look of their Looking Glass. It also opens up the possibility to sell Looking Glass as a DIY kit for enthusiasts.

Disadvantages: The facial and pattern recognition functions only work in a well-lit environment. If the camera is in direct light or in darkness, the functions will not work.

Opportunities: As there is a limit to the amount of GPIO ports on a Raspberry Pi, wireless sensors which communicated with Looking Glass over Wi-Fi could be a viable alternative. The ability to connect to Looking Glass through a companion smart phone application would give the user the ability to view their sensors as well as rename the current sensors.

Limits: A Raspberry Pi can support up to eight sensors through GPIO ports. If a user needs to monitor nine or more areas, they would need a second Looking Glass. This is obviously not ideal for the user.

4 Further development or research

With more resources, where could the results of this project lead to?

I believe Looking Glass has huge potential to expand into a marketable product. If I had more time to develop Looking Glass I would add the following features and functions:

- **Wireless Sensors**
  This would allow users to have more than eight sensors attached to Looking Glass.
- **Companion Application**
  This app could be used to view the status of the sensors while the user is out of the house. They could also have the ability to update all user settings of Looking Glass.
- **Better Trained Haar Cascades**
  The most common comment on Looking Glass from the test group of volunteers was that the pattern recognition was not reliable enough. This is mainly due to the Haar Cascade file used for recognition. If time allowed, I would have trained the cascade file to be more accurate.
- **Remove User**
  If Looking Glass is used in a rental environment, a landlord may wish to remove old tenants from the userbase. Currently there is no function within Looking Glass to achieve this, but it could be easily implemented in the future.

- **Add Multiple Users**

  In its current state, Looking Glass will detect if there are no users registered and will force a user to register on initial start-up. There is no option currently to add additional users.

- **Update Pattern**

  Like the new user function, Looking Glass will detect if there is no stored pattern and force a user to set up a new pattern. There is currently no option to update the master pattern.

5 References

Hadiono, K. (2015). *How to install a USB webcam in Raspberry Pi - Ask Xmodulo*. [online] Ask Xmodulo. Available at: http://ask.xmodulo.com/install-usb-webcam-raspberry-pi.html [Accessed 10 Oct. 2017].

Opencv.org. (2016). *About - OpenCV library*. [online] Available at: https://opencv.org/about.html [Accessed 10 Oct. 2017].

Qt.io. (2018). *Qt | Cross-platform software development for embedded & desktop*. [online] Available at: https://www.qt.io/ [Accessed 12 Apr. 2018].

Rosebrock, A. (2017). *Raspberry Pi: Deep learning object detection with OpenCV - PyImageSearch*. [online] PyImageSearch. Available at: https://www.pyimagesearch.com/2017/10/16/raspberry-pi-deep-learning-object-detection-with-opencv/ [Accessed 10 Oct. 2017].

sentdex (2016). *Haar Cascade Object Detection Face & Eye - OpenCV with Python for Image and Video Analysis 16*. [video] Available at: https://www.youtube.com/watch?v=88HdqNDQsEk [Accessed 27 Feb. 2018].

sentdex (2016). *Training Haar cascade object detection - OpenCV with Python for Image and Video Analysis 20*. [video] Available at: https://www.youtube.com/watch?v=eay7CgPlCyo [Accessed 27 Feb. 2018].

6 Appendix

6.1 Project Proposal

Project Proposal

Looking Glass

Aaron Beveridge, x14469052, x14469052@student.ncirl.ie

BSHCYB4 - BSc (Honours) in Computing

Cyber Security

17/10/2017

Objectives

Looking Glass will be a disguised home security system. It will comprise of a Raspberry Pi, a monitor, a sheet of two-way glass and door/window sensors.

Looking Glass will look like an ordinary mirror until a user activates it. When a user unlocks the Looking Glass by passing the two-step authentication, they will be able to see the status of all the door/window sensors around their home. This two-step authentication process will consist of facial recognition and an augmented reality (AR) pattern unlock.

A user will be able to arm the sensors from the connected app on their smart device. If a sensor is triggered while armed, a notification will be sent to the user's smart device.

For Looking Glass to work, I will need to achieve the following:

- Assemble the hardware components.
- Implement object and facial recognition.
- Design an AR pattern unlock system.
- Implement a method to link sensors to the main Looking Glass system.
- Develop the Looking Glass application for smart devices.
- Implement a notification system.

Background

I have always been fascinated with the possibilities of the RaspberryPi. One project which I particularly admired was the "Magic Mirror", which essentially comprises of a RaspberryPi, a LCD screen and a sheet of two-way glass. When text is displayed on the monitor, it shines through the glass and appears on the reflective side. Typically, magic mirrors display information such as live twitter feeds, local weather or traffic and Google Calendar reminders.

My inspiration for this project came from a conversation I had with my Uncle. My uncle was telling me about his recent break in. Thieves broke into his home and disabled the alarm box before it could alert authorities. If the thieves were unable to locate the alarm box, they would have been caught. This gave me the idea to combine the "Magic Mirror" concept with a home security system. Only the home

owners would know that the mirror contains the security system as it would not display anything unless activated.

Once I complete Looking Glass, I would like to partner with the Gardai or companies who offer home monitoring services such as Eir. Rather than just a notification about a sensor being triggered, I would like to send a notification to the Gardai or the home monitoring service. This will give users peace of mind if they are leaving their home vacant for an extended period of time.

Technical Approach

I will be building upon MagicMirror², an open source, modular API used as the backbone of most magic mirror projects (https://magicmirror.builders/). MagicMirror² provides templates for developing new modules. There are also facial recognition modules which I can build upon for my project.

As of yet, there is no object recognition module for MagicMirror². I plan to use Google's TensorFlow API to overcome this problem (www.tensorflow.org). TensorFlow requires a user to teach it what objects to identify.

For the proposed AR pattern unlock, Looking Glass will try to identify a user's closed fist. If it detects their fist, Looking Glass will highlight it on screen. Looking Glass will then record the pattern that is made with the user's fist. If the pattern matches the pre-set pattern, the system will unlock.

Special resources required

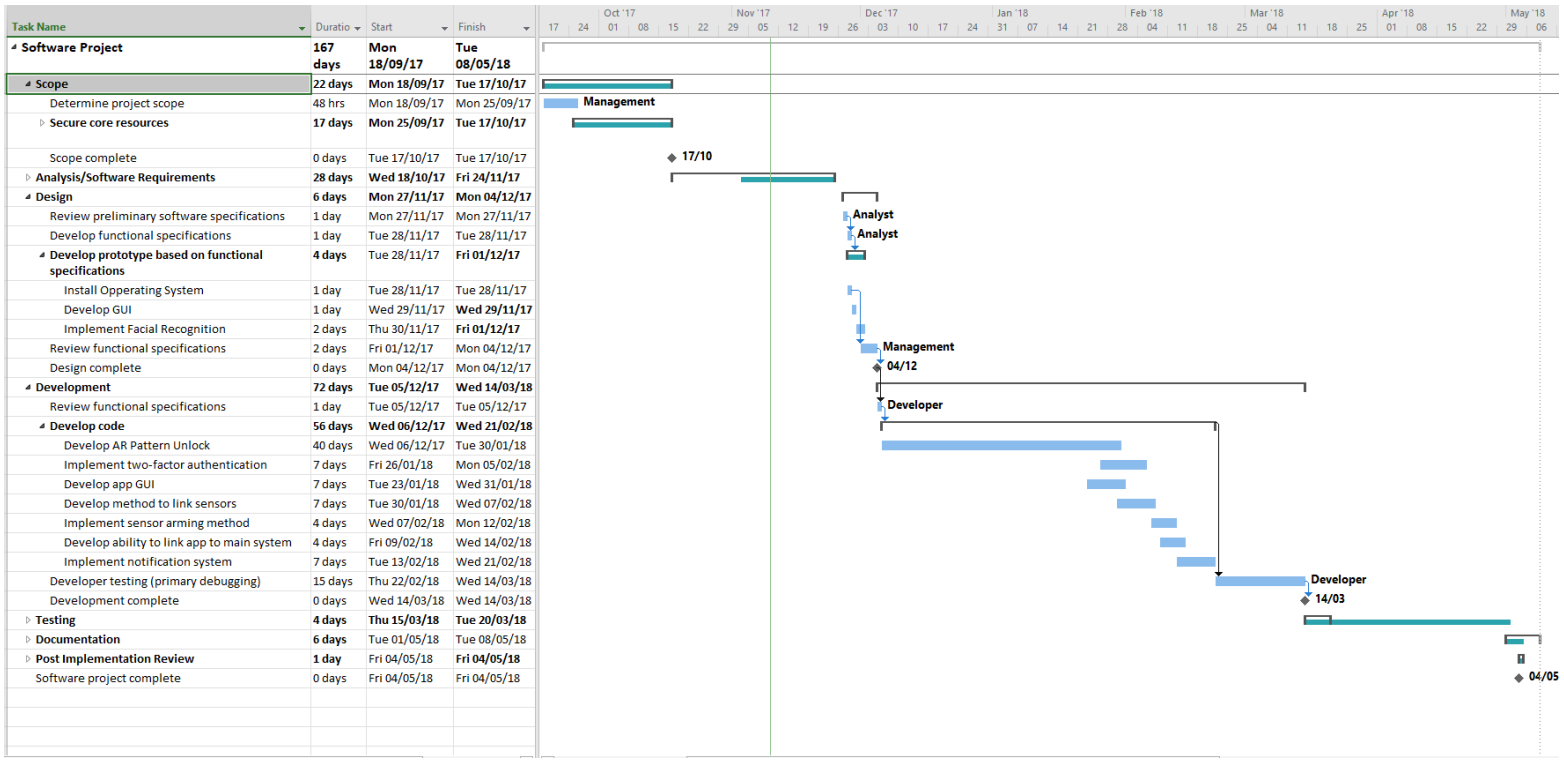
The special resources which I will require to complete Looking Glass will be:

- A RaspberryPi
- A LCD monitor
- A webcam
- A sheet of two-way glass which is larger than the monitor
- Wooden planks to construct the housing of Looking Glass
- Door/Window sensors

A RaspberryPi will be provided by the National College of Ireland. I already own a suitable monitor and webcam. The glass will have to be sourced from a glazier's

shop or online while the materials for the housing can easily be purchased in a hardware store or garden centre. The sensors can be purchased online.

Project Plan



Task Name	Duratio	Start	Finish
Software Project	**167 days**	**Mon 18/09/17**	**Tue 08/05/18**
Scope	**22 days**	**Mon 18/09/17**	**Tue 17/10/17**
Determine project scope	48 hrs	Mon 18/09/17	Mon 25/09/17
Secure core resources	**17 days**	**Mon 25/09/17**	**Tue 17/10/17**
Scope complete	0 days	Tue 17/10/17	Tue 17/10/17
Analysis/Software Requirements	**28 days**	**Wed 18/10/17**	**Fri 24/11/17**
Design	**6 days**	**Mon 27/11/17**	**Mon 04/12/17**
Review preliminary software specifications	1 day	Mon 27/11/17	Mon 27/11/17
Develop functional specifications	1 day	Tue 28/11/17	Tue 28/11/17
Develop prototype based on functional specifications	**4 days**	Tue 28/11/17	**Fri 01/12/17**
Install Opperating System	1 day	Tue 28/11/17	Tue 28/11/17
Develop GUI	1 day	Wed 29/11/17	**Wed 29/11/17**
Implement Facial Recognition	2 days	Thu 30/11/17	**Fri 01/12/17**
Review functional specifications	2 days	Fri 01/12/17	Mon 04/12/17
Design complete	0 days	Mon 04/12/17	Mon 04/12/17
Development	**72 days**	**Tue 05/12/17**	**Wed 14/03/18**
Review functional specifications	1 day	Tue 05/12/17	Tue 05/12/17
Develop code	**56 days**	**Wed 06/12/17**	**Wed 21/02/18**
Develop AR Pattern Unlock	40 days	Wed 06/12/17	Tue 30/01/18
Implement two-factor authentication	7 days	Fri 26/01/18	Mon 05/02/18
Develop app GUI	7 days	Tue 23/01/18	Wed 31/01/18
Develop method to link sensors	7 days	Tue 30/01/18	Wed 07/02/18
Implement sensor arming method	4 days	Wed 07/02/18	Mon 12/02/18
Develop ability to link app to main system	4 days	Fri 09/02/18	Wed 14/02/18
Implement notification system	7 days	Tue 13/02/18	Wed 21/02/18
Developer testing (primary debugging)	15 days	Thu 22/02/18	Wed 14/03/18
Development complete	0 days	Wed 14/03/18	Wed 14/03/18
Testing	**4 days**	**Thu 15/03/18**	**Tue 20/03/18**
Documentation	**6 days**	Tue 01/05/18	Tue 08/05/18
Post Implementation Review	**1 day**	Fri 04/05/18	**Fri 04/05/18**
Software project complete	0 days	Fri 04/05/18	Fri 04/05/18

Technical Details

The operating system which I will run on the Raspberry Pi will be Raspbian, a Unix-like operating system. I will mainly be developing in Java for both the Looking Glass app and main system. I plan to implement modules from https://magicmirror.builders/ for facial recognition as well as Google TensorFlow for recognising a user's fist.

Evaluation

As Looking Glass will be a security system, there will have to be a lot of testing to ensure it is safe to use in a home environment.

I will need to test the following aspects of Looking Glass:

- Facial Recognition
- Pattern Unlock
- Door/Window Sensors
- Notification System
- System Integrity

Facial Recognition:

I will need to minimise the risk of false positive results. A possible test will be to get a group of people to try and unlock the Looking Glass. I will record the outcome of all interactions.

Pattern Unlock:

Similar to facial recognition, I will need to minimise the risk of false positive results. This will be done in the same way.

Door/Window Sensors:

I will need to ensure the sensors stay connected to the main system. I will monitor their connection status over a given period of time.

The sensors also need to only react to when a window or door is actually open. I will need to thoroughly test that the sensors are not triggered by external sources.

Notification System:

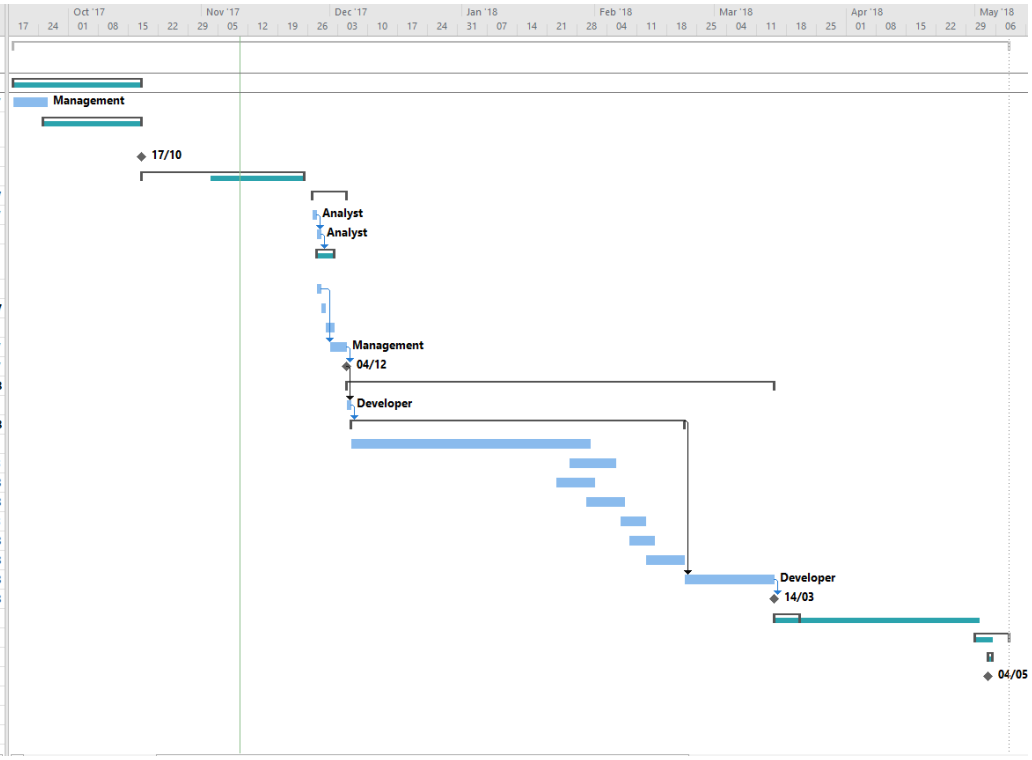
Firstly, I will need to ensure that the notification method is triggered when a sensor is triggered. This can be done by writing to a log file.

Next, I will test that the notification is being received by the app. Again, this can be done with log files.

Finally, I have to ensure that the notification is being displayed by the app. I will have to test this with multiple devices.

6.2 Project Plan

Task Name	Duratio	Start	Finish
Software Project	**167 days**	**Mon 18/09/17**	**Tue 08/05/18**
Scope	**22 days**	**Mon 18/09/17**	**Tue 17/10/17**
Determine project scope	48 hrs	Mon 18/09/17	Mon 25/09/17
Secure core resources	**17 days**	**Mon 25/09/17**	**Tue 17/10/17**
Scope complete	0 days	Tue 17/10/17	Tue 17/10/17
Analysis/Software Requirements	**28 days**	**Wed 18/10/17**	**Fri 24/11/17**
Design	**6 days**	**Mon 27/11/17**	**Mon 04/12/17**
Review preliminary software specifications	1 day	Mon 27/11/17	Mon 27/11/17
Develop functional specifications	1 day	Tue 28/11/17	Tue 28/11/17
Develop prototype based on functional specifications	**4 days**	Tue 28/11/17	**Fri 01/12/17**
Install Opperating System	1 day	Tue 28/11/17	Tue 28/11/17
Develop GUI	1 day	Wed 29/11/17	**Wed 29/11/17**
Implement Facial Recognition	2 days	Thu 30/11/17	**Fri 01/12/17**
Review functional specifications	2 days	Fri 01/12/17	Mon 04/12/17
Design complete	0 days	Mon 04/12/17	Mon 04/12/17
Development	**72 days**	**Tue 05/12/17**	**Wed 14/03/18**
Review functional specifications	1 day	Tue 05/12/17	Tue 05/12/17
Develop code	**56 days**	**Wed 06/12/17**	**Wed 21/02/18**
Develop AR Pattern Unlock	40 days	Wed 06/12/17	Tue 30/01/18
Implement two-factor authentication	7 days	Fri 26/01/18	Mon 05/02/18
Develop app GUI	7 days	Tue 23/01/18	Wed 31/01/18
Develop method to link sensors	7 days	Tue 30/01/18	Wed 07/02/18
Implement sensor arming method	4 days	Wed 07/02/18	Mon 12/02/18
Develop ability to link app to main system	4 days	Fri 09/02/18	Wed 14/02/18
Implement notification system	7 days	Tue 13/02/18	Wed 21/02/18
Developer testing (primary debugging)	15 days	Thu 22/02/18	Wed 14/03/18
Development complete	0 days	Wed 14/03/18	Wed 14/03/18
Testing	**4 days**	**Thu 15/03/18**	**Tue 20/03/18**
Documentation	**6 days**	Tue 01/05/18	Tue 08/05/18
Post Implementation Review	**1 day**	Fri 04/05/18	**Fri 04/05/18**
Software project complete	0 days	Fri 04/05/18	Fri 04/05/18



6.3 Monthly Journals

6.3.1 September

Reflective Journal

Student name: Aaron Beveridge

Programme: BSc in Computing – Cyber Security

Month: September

Overview

There was quite a lot of panicking this month, especially during the second week as quite a lot of time was devoted to our AI project. During the first two weeks, I gave very little thought to my project idea as I felt that my other assignments were more urgent. As soon as my AI project was uploaded, I shifted my attention to my software project idea.

I toyed with the idea of end-to-end email encryption and a program which could mass call spam callers so that they could not continue making anymore spam calls. I was quickly informed that this idea was highly illegal and already done before.

The idea I settled on is an extension of the “magic mirror” concept. These mirrors are controlled by a Raspberry Pi. The Pi is connected to a monitor. This monitor has a sheet of 2-way plastic overlaid. When the monitor displays white text on a black background, the text will shine through the plastic and appear on the reflective side of the plastic, creating the “magic mirror” concept. To build upon this, I planned to install a webcam in the mirror and voice detection software on the PI. When a user says, “save my outfit”, the mirror would take a picture of the user, identify the articles of clothing they are wearing, save this picture to the database of outfits, dissect the image and save a picture of each article of clothing to the wardrobe database. Through machine learning, the mirror would begin to learn a user’s own fashion sense. The mirror could then either suggest an entire outfit or recommend an outfit to match a previously worn article of clothing. To keep

all data secure, I planned to implement facial and voice recognition software as a two-step authentication process.

This idea was pitched to three lectures on the 6th of October. All three liked the idea, one even said she would buy the mirror if it was already available. The only criticism was that the project idea is as wide as it is complex and that it would probably be best suited to a team rather than one student. The project was accepted with revisions in the end. I was told to try implement an Augmented Reality (AR) feature rather than an object detection feature. I will have to give the mirror a catalogue of images of clothing, these images would then be mapped onto the user through the mirror.

I am happy with how this month has gone and I am excited to begin work on my mirror

My Achievements

This month I had multiple ideas for my software project, one of which was approved with revisions.

My Reflection

I felt that my time management needs attention. I have registered on Trello, a workflow organiser, as to keep track of upcoming assignments and uploads. Hopefully this will help keep me on top of college this year.

6.3.2 October

Reflective Journal

Student name: Aaron Beveridge

Programme: BSc in Computing – Cyber Security

Month: October

Overview

Since my last journal, I have talked with multiple lectures about my project idea. The consensus from them was that my project would be more suitable for a group of students rather than just myself, especially in regards to the augmented reality aspect of my project. From talking to my supervisor, I have decided to rework my project idea.

I have decided to shift my focus to the security aspects of the project. For my new idea, I will still use the mirror element and rework the login mechanism. I will still attempt to implement a two-factor authentication process to the mirror. The first step will continue to be facial recognition. The second step will change to a pattern unlock. Rather than the conventional touchscreen pattern unlock, I would like to combine it with object recognition. A user would present their closed fist to the camera and draw their pattern in the air. The camera would track the users fist and try to match it with a pre-set pattern.

Once the user has unlocked the mirror, the user will be able to view an overview of their home security system. This overview will show the status of sensors which are attached to door and window frames. If a sensor is triggered, it will be logged to the mirrors for a user to see. I would also like to implement a push notification feature if a sensor is triggered. The push notification will more than likely be a stretch goal.

My inspiration for this new idea came from my uncle. His house was recently broken into. The thieves found the alarm box and disabled it before continuing to

rob his house. My project would disguise the home security system as an ordinary mirror, preventing thieves from disabling it.

My Achievements

I have talked with my supervisor and we have concluded that I need to change my project idea. I have since came up with a new idea which will be discussed in greater detail with my supervisor in the coming days.

My Reflection

I feel like a lot of time has been wasted this month. Between all the other assignments and deliverables, I have had to do over the past few weeks, I have been up to my eyes with work and have not been able to devote enough time to this project. I hope to change this for this month as at the time of writing, I have exactly four weeks to build a prototype.

6.3.3 November

Reflective Journal

Student name: Aaron Beveridge

Programme: BSc in Computing – Cyber Security

Month: November

Overview

Since the October journal, I have completed all documentation required for the Mid-Point presentation and created a very early prototype of Looking Glass. I am happy with how my documentation has turned out so far, but I feel as if my prototype is on the disappointing side. I have created a mock-up of the user interface using HTML and JavaScript. Unfortunately, this prototype has no functionality apart from navigation between pages. I would have liked to have been able to show the sensors connected to the Raspberry Pi but my shipment from Amazon has been delayed.

Over the next few weeks I doubt I will have much time to focus on this project as the Christmas exams are around the corner. By the next journal upload I would like to have the object recognition elements introduced to the project.

My Achievements

I have completed my project proposal, project plan and technical report for the mid-point presentation. I have also created a basic prototype.

My Reflection

I like the direction that the project is going. I know I do not have much of the actual coding completed but just having the documentation and plan in place has put my mind at ease.

6.3.4 January

Reflective Journal

Student name: Aaron Beveridge

Programme: BSc in Computing – Cyber Security

Month: January

Overview

Over the past few weeks we have had quite bad snow. This allowed me to get a lot of work done for various projects. I have been able to install OpenCV on my Raspberry Pi as well has implementing facial detection. Now that I have facial detection working, I can build upon it to implement facial recognition and pattern recognition.

My Achievements

I have implemented facial detection. The camera will capture each frame and check for a face. If a face is present, a blue square will be drawn around each face.

My Reflection

I highly underestimated how long it would take to install OpenCV. Due to failed installs it took over four days just to get it correctly installed. I have created a disk image of my SD card at the point where OpenCV was successfully installed in case I have to revert to an earlier state.

6.3.5 February

Reflective Journal

Student name: Aaron Beveridge

Programme: BSc in Computing – Cyber Security

Month: February

Overview

This month was all about implementing as many of the remaining features as possible while also getting other assignments completed.

This month I was able to get the pattern recognition implemented using OpenCV and csv comparisons. The file I am using to detect hands is rather basic and has a high inaccuracy, so I will need to train my own file at some point.

My Achievements

I have implemented pattern recognition. The camera will capture each frame and check for a hand. If a hand is present, its coordinates are saved to a csv file. This csv file is compared against a master pattern. If the two csv files are similar enough, the method will return true.

My Reflection

I am excited by my progress this month. I now have both components of the two-factor login. I now need to finalize my GUI as well as linking my sensors to Looking Glass.

6.3.6 March

Reflective Journal

Student name: Aaron Beveridge

Programme: BSc in Computing – Cyber Security

Month: March

Overview

This month was quite exhausting as I had quite a lot of assignments to work on simultaneously. I have not made as much progress as I would have liked this month. I have not been able to connect the sensors over Wi-Fi. I believe that the wireless ESP board that I ordered online came damaged as the on-board light is not turning on when I boot the board. I have decided to wire the sensors directly to the Raspberry Pi. I have been able to receive true and false signals from the sensor based on whether the sensor is in an open or closed state.

My Achievements

I have connected the sensors to Looking Glass and received information from the sensors.

My Reflection

I am beginning to get worried about how much work I have left to complete Looking Glass. The GUI needs to be completed, the sensors need to update the status screen, the login needs to be implemented and an auto lock function needs to be added.

6.4 Other Material Used

6.4.1 Customer Testing Survey

Looking Glass Survey

For each item identified below, circle the number
to the right that best fits your judgment of its quality.
Use the rating scale to select the quality number.

Survey Item	Scale				
	Poor	Good			Excellent
1. User Interface	1	2	3	4	5
2. Ease of use	1	2	3	4	5
3. Facial Recognition Function	1	2	3	4	5
4. Gesture Recognition Function	1	2	3	4	5
5. Physical Appearance	1	2	3	4	5
6. Any other comments					