PiMirror

Functional Specification

By Kieran Turgoose; 14355046

Date Finished: 23/11/2017

Contents

1. Introduction

1.1 Overview

The system to be developed for the Final Year Project is a Raspberry Pi operated Smart Mirror with webcam, that uses facial recognition to switch between provided profiles. The Smart Mirror is comprised of a monitor behind two-way glass, allowing for an interface to be shown through the mirror, as well as the mirror image itself. The interface can be used to display numerous widgets of the user's preference. The profiles allow the interface of the mirror to change to that user's pre-set widget configuration. There will also be an accompanying android application for settings management, and widgets configuration.

Idea Generation
The idea for this project originated through the desire to develop an IoT-based project. During the INTRA period of third year spent at SAP, a primitive Smart Mirror was shown during a developer's showcase, displaying the basic interface and some minor widgets. These widgets included the date, time and a scrolling news bar at the base of the screen. This was the main source for pursuing further research into this particular project.

Need for the System
This system can be a great addition to the ever-growing market of Internet of Things (IoT) household objects. We are seeing more and more IoT additions lately, from light bulbs and thermostats, to video and voice enabled doorbells. The Smart Mirror can be added to this market and provide use to the everyday adult, both as a mirror and as an information interface. Likely to be used in either an entrance hallway, bathroom, or bedroom, this Smart Mirror will save the user time throughout their day, most importantly in the mornings. When the user wakes up and uses their mirror they will be updated of the weather for the day, any important email/schedule updates, traffic jams that may hinder their journey, important news articles or even just pre-written memos for the user (shopping lists etc..).

Functions

- **Monitor Connection**

This Smart Mirror is designed by connecting the pi to a lightweight monitor, via HDMI, which will be placed behind two-way glass so that the screen will be visible through the glass, but that it will also be usable as a mirror.

- **Smart Mirror Interface**

The pi will host an interface that can provide functionality to the screen through widget-like overlays. These widgets are, for example, things such as; weather updates, scrolling news updates, local traffic, Gmail, to-do list, and calendar.

- **Facial Recognition**

A major component of this project will be to make use of a webcam, via USB, to detect when a user moves in front of the mirror, recognise the user from a pre-set collection of users, and then turn the interface on. Once the user is recognised the system will alter the widgets on the display to tailor it to that specific user's settings.

- **Android Application**

The user also needs some way of interacting with the Raspberry Pi in order to alter settings, this will most likely be an android application. However, gesture and voice recognition is being researched as a potential option, although this may be too tasking for the timeframe of the project. This application will consist of the functionality to login via a Google account, connect to the pi, change profiles, edit which widgets are assigned to your user, change the position of the widgets on the interface, and allow the user to turn off their interface if a second person is detected so that their data is hidden.

1.2 Glossary

SSH: Secure Shell (SSH) is a cryptographic network protocol for operating network services securely over an unsecured network. It allows the user to connect to and control the Raspberry Pi through linux shell commands over a network.

Raspberry Pi: Raspberry Pi is a small computer, ideal for developing small practical projects.

Internet of Things (IoT): The network of physical devices, vehicles, home appliances, and other items embedded with electronics, software, sensors, actuators, and network connectivity which enable these objects to connect and exchange data.

Google Tensorflow: An open-source software library for dataflow programming across a range of tasks. Used for machine learning and neural networks.

Eigenfaces: An appearance-based approach to face recognition that seeks to capture the variation in a collection of face images and use this information to encode and compare images of individual faces in a holistic manner.

Fisherfaces: An enhancement of Eigenfaces. Especially useful when facial images have large variations in illumination and facial expression.

Local Binary Patterns: Is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighbourhood of each pixel and considers the result as a binary number. Considered robust in handling lighting variations.

2. General Description

2.1 Product / System Functions

2.1.1 Raspberry Pi 3 Model B

The Raspberry Pi is the main computing power that will operate the Smart Mirror. It will connect to the monitor used for the mirror display via HDMI. The Pi will store the majority of the image processing code, written in Python using the OpenCV library. OpenCV is an open-source computer vision library for C++ and Python. The main functionality of the python code will be to detect motion, which will trigger the Smart Mirror interface to enable, and to recognise the face of the user and load the correct profile based on that user's data. This will be implemented using a USB-connected webcam. The Smart Mirror interface code will also be stored on the Pi. The Pi will be connected to the internet, and will have access to the database so that it can access the profile information necessary to recognise and change the user as they use the Smart Mirror. The Pi will also need to interact with the Android application. This particular model was chosen as it is the newest and fastest available Pi, it comes with added benefits such as; more RAM, built in Wi-Fi and Bluetooth.

2.1.2 OpenCV

There are many ways of implementing facial recognition into a project such as this; OpenCV in Python was chosen however as it seemed like the more manageable choice for a project of this scope. Alternative options, such as Google Tensorflow, may have been viable but were decided against due to the fact that documentation is not as openly found as with OpenCV, that OpenCV is open source and easily accessible, and also the developer has some minor past experience utilising OpenCV for other means.

OpenCv provides three algorithms to tackle the facial recognition problem, namely; Eigenfaces, Fisherfaces, and Local Binary Patterns Histograms. The algorithm chosen for the purpose of this project is Local Binary Patterns Histograms (LBPH), due to this particular algorithm not being affected by lighting changes. This choice may change is results aren't progressing as expected, but early research suggests that this will be the best method to use.

2.1.3 Smart Mirror Interface/Widgets

The Smart Mirror interface itself will be utilised by using 3rd Party open-source code called "Magic Mirror". This provides a black background for the monitor and also widgets that can be added to the interface. Some of the basic widgets include weather updates, scrolling news updates, local traffic, Gmail, to-do list, and calendar. This will be used so as to get my system up and running with profiles, facial recognition and launching the Smart Mirror when a user is detected. This interface is mainly written using a combination of HTML, CSS and JavaScript, and therefore the plan is to perhaps create a personal interface and some widgets later on in the development process after this is set up and tested, rather than utilising Magic Mirror for the duration of the project. Some potential future widgets such as Spotify could perhaps require an additional external speaker to be attached to the Raspberry Pi, this will be shown in the design sections as a precaution, in case it is required.

2.1.4 Android Application

The Android application to be developed will have the functionality of controlling the Smart Mirror interface. It will be written in Java. This will mainly be for the ability to assign different widgets to each profile as the user desires. The application will connect to the Raspberry Pi via SSH commands, this will allow variables in the code on the Pi to be edited depending on which options are selected on the phone. Using the application will also allow the user to reposition the widgets they have selected on the interface itself. The application will also have the basic functionality of logging in/out of your account and allowing the user to disable the interface from appearing when they stand in front of the mirror. This is for situations whereby the user potentially may not want other people seeing their data, or that they just want the full mirror view available to them.

2.1.5 BackEnd / Database

A database will be required for the storing of user's profiles and profile data, widget names assigned to each profile and likely the user's pictures that are required for the facial recognition software. This database is likely going to be a MySQL database. The user's profile data and widget names will need to be accessed by the android app and the pi in order to change each profile correctly as the differing users are recognised by the Smart Mirror.

2.2 User Characteristics and Objectives

For this system, the user community can have a rather large variety. This system is not focused on any particular group of people, in that essentially anybody will be able to use it regardless of their age or gender, and once set up, should be fairly intuitive to use even for those with limited technological experience.

User Community

- **Working Adults and Students**

While not being exclusive to this group, this product would likely be more desired by working adults. They, alongside students, would benefit more from this product due to the nature of a lot of the widgets, such as Gmail, Calendar, traffic and news reports; and that it, in theory, would be more useful for people who have little time in the morning, i.e. before work/University. Younger users such as children perhaps might not have as much use from this particular product for most of the widgets that are currently in mind for development. However, alterations and additions to widgets could certainly cater this product more towards children if the opportunity arose. Possible widgets such as YouTube and Spotify would cater to most users, regardless of their age.

- **Households with Multiple Occupants**

This product could also be catered more towards households with multiple occupants. While not being worthless to single occupancy households, the fact that only one product would be required for multiple people within a household is certainly a positive for the occupants with regards to the possible price point. The functionality of this product would not alter whether there were multiple occupants or just a single occupant, therefore this preference is purely from a financial standpoint, if this were to become a purchasable product.

Expected Expertise
With regards to the expected expertise for use of the system, it is very minimal. Basic understanding of smartphone applications would be all that was required for the Android application, and the Smart Mirror requires no interaction other than standing in front of the mirror. Therefore, users who would potentially have issue with this product, such as the elderly and very young children, i.e. under 10 years; may not be able to fully utilise this product. This would mainly be due to inexperience with smartphone applications, and even a lack of understanding behind the concept of the facial recognition in the mirror. However, this problem is seemingly getting less frequent due to the vast nature of technology in today's everyday life.

Objectives and Requirements
From the perspective of the selected main user of the system, the working adult, there are many key requirements of the system that they would expect.

- Reliably working facial recognition.
- Quick loading of widgets/interface.
- Up-to-date data (emails, news, etc...)
- Ability to log out.
- Ability to disable interface.
- Easy and quick use of app.
- Visually pleasing interface.

Desirable Characteristics (Wish List)
These characteristics would certainly make for a more appealing product and are something that could potentially be implemented with further development.

- Hands-free interactions: Voice/Gesture control
    - No requirement for Android application.
- Touchscreen functionality.
- Wi-Fi cameras instead of USB connectors.
- Transparency adjustment
    - Alter the transparency of the widgets on the interface
- Spotify/YouTube integration.
- Security
    - Absolute security of data.
- Snapchat-like facial features
    - As an added functionality to cater to a younger audience.

2.3 Operational Scenarios

Potential Scenarios:

- Setting up the app to connect to the Smart Mirror.
- Turning off interface from app.
- Changing widgets from app.
- Standing in front of the mirror when off.
- Standing in front of the mirror when someone else is already there.

For these scenarios, the first three, which describe an action the user can perform on the Android application, will be showcased through use cases; whereas the rest will be explained through an example scenario.

2.3.1 Use Cases

Use Case 1	**Setting Up Application**
Goal in Context	Set up the application and connect it to your Smart Mirror
Preconditions	Smart Mirror; Google Account; Android Phone
Success End Condition	Application connected to Smart Mirror account
Failure End Condition	Application fails to connect to Pi/Google account
Actors	User
Step	**Action**
1	Download the application from the Google Play Store
2	Upon prompt, login to application via Google account
3	Go to Settings
4	Enter the Raspberry Pi IP address into the available field
5	Upon prompt, confirm the address
Branches	
5B	Do not confirm the address, back out instead

Use Case 2	**Turning off the Interface**
Goal in Context	Activating setting to turn off interface if multiple people are detected
Preconditions	Application connected; User logged in; Interface on
Success End Condition	Interface setting option will be turned on
Failure End Condition	Setting doesn't work / Unable to turn the setting on
Actors	User
Step	**Action**
1	Open the application
2	Upon prompt, login to application via Google account
3	Go to Settings
4	Select option for turning off interface for multiple people

5	Upon prompt, confirm the option
Branches	
2B	Do not / Cannot login
5B	Do not confirm the option, back out instead

Use Case 3	**Managing Widgets**
Goal in Context	Selecting the widgets to display from the application
Preconditions	Application connected; User logged in
Success End Condition	New widget configuration will be loaded to the interface
Failure End Condition	Widgets will not update / Cannot be altered
Actors	User
Step	**Action**
1	Open the application
2	Upon prompt, login to application via Google account
3	Go to Widgets
4	Select widgets to display on the interface
5	Select the position on the interface
6	Confirm selection
Branches	
2B	Do not / Cannot login
6B	Do not confirm the option, back out instead

2.3.2 Example Scenarios

- **Standing in front of the Smart Mirror when it is turned off:**

In this scenario, a user has set up their Smart Mirror and has configured which widgets they want displayed for their interface. The user enters the room, the camera connected to the Smart Mirror detects movement and starts to run the facial recognition software to detect a face. As the user stands in front of the mirror the camera detects their face and correctly recognises them as the user, based on the previously trained images provided by the user. The Pi then proceeds to boot up the interface with widgets displayed as configured by the user.

· **Standing in front of the Smart Mirror when somebody else approaches it:**
For this scenario, it must be noted that the user who is already using the Smart Mirror has the setting enabled which turns off the interface if another face is detected.

The user stands in front of the Smart Mirror, fixing his tie as he gets an email update. As he is reading the email he doesn't notice his roommate enter the room. The roommate comes over to talk to the user and once he walks in proximity to the mirror, the camera, which is still running facial recognition software to detect faces, calculates that there are now two faces present in the image. The Pi immediately kills the interface so that the user's information is protected from the roommate. As the roommate begins talking, the user breathes a sigh of relief as they had important reminders on their interface relating to a surprise birthday present for the roommate.

2.4 Constraints

For the system being developed the following constraints can be applied.

· **Speed**
Speed is one of the key factors of most IoT products. Speed is needed throughout most of the functionality of these systems, therefore if they are not up to what the user is expecting they can be quickly shunned in the marketplace. For this product, speed will be essential for many of the features, namely: the speed at which the user is recognised as they stand in front of the mirror; the speed in which the interface is loaded and up-to-date widgets are displayed; the speed in which the interface is refreshed and the profile widgets are updated after changes are made from the app. These are all integral to a satisfying user experience.

· **Hardware Performace**
With regards to the hardware, our system will be utilising a Raspberry Pi 3 model B. This particular model has 1GB of RAM. 1GB is most likely too little to train the facial recognition images on. Therefore, the likely option will be to train the images on a laptop or PC, and then deploy the software on the Raspberry Pi. There is also the consideration of the webcam to be used for the camera feed. Higher quality cameras of 1080p resolution or higher are likely to provide the best results for both efficiency and effectiveness of the facial recognition. This is something that needs to be taken into consideration when purchasing the webcam.

· **Camera Hardware**
One unfortunate constraint is related to the type of camera that can be used in conjunction with the Raspberry Pi. The Raspberry Pi camera module is an obvious choice, it is small and has good camera quality. Unfortunately, it also means that the python code would need to be catered for use of this camera, meaning that it would only function with this particular camera. For this reason, it seems more practical to use a USB webcam as the choice is more diverse for the user. However, there is also the case that the Raspberry Pi does not support every make of webcam, with a list of tested compatible webcams available on the Raspberry Pi website. While a non-listed webcam may well work, it is surely not worth the risk and so this list will be consulted when making the final camera choice.

· **Facial Recognition**

Facial Recognition is a key part of this project and therefore must be working effectively in order to supply a satisfying product. However, since I am using an open-source library, “OpenCV”, and given the timeframe and resources behind this product, this, more primitive facial recognition, is not likely to be near industry standards e.g. iPhone X’s “FaceID”. There are other discrepancies such as camera quality, as mentioned above, and also the quality of the user’s images. There will need to be a certain number of images of the user’s face from multiple viewpoints and angles. This ideal number will be calculated at a later date, but in theory, the more variety in pictures, the better quality the learning algorithm should be.

· **Ease of use for the app**

The design of the Android application must be centred around the user and the ease at which they can fully utilise it. Smartphone application design principles will be consulted, but considering the application will have limited functionality there should be no issues in making all its functions easy to find and use, regardless of the user’s technological ability.

3. Functional Requirements

3.1 Interface with widgets:

- **Description:** This functional requirement is to ensure that the Raspberry Pi is connected to the monitor, and that the Smart Mirror interface can be loaded onto the screen with widgets enabled.
- **Criticality:** This requirement is the most critical for this system. Without this functionality, there can be no Smart Mirror. Without the following requirements below there can still be a functioning system, albeit a lesser quality one. However, if this requirement is not fulfilled then there is no system to present.
- **Technical Issues:** This requirement requires multiple facets of the system to work together, i.e. the numerous widgets, and the interface itself. Ensuring this functionality is developed and configured correctly could possibly be a stumbling block for the implementation phase.
- **Dependencies with Other Requirements:** For the overall system to work to its highest standard, this requirement must interact with two other requirements that will follow; “Profiles” and “Android Application”. This is due to different widget configurations being assigned to different profiles that should be able to change, and the widget configuration being altered by the user on their Android application. However, just to get the widgets working on the interface, it doesn’t depend on any of the other requirements.

3.2 Android Application:

- **Description:** This requirement is to allow users the functionality of logging in with their Google account, and altering the widget configuration on the Smart Mirror interface.
- **Criticality:** This requirement is the second most critical for this system, because without it there would be no control over what widgets can be displayed on the interface. This would mean that no set widget profiles would be allocated to users but that each time they went to the Smart Mirror they would have to edit the widgets to their preference.

- **Technical Issues:** This requirement requires development of an Android application, which in itself can be quite taxing, and also linking the application to be able to edit the widget code via SSH into the system.
- **Dependencies with Other Requirements:** This requirement depends on the Smart Mirror interface being up and working properly with widgets available for selection. The application itself can be developed as a standalone but its functionality requires the Smart Mirror in operation.

3.3 Profiles:

- **Description:** This requirement is to allow multiple users to have pre-set widget configurations. This would mean that, using the Android Application, a user would be able to simply login as themselves and the interface would change to their pre-set configuration; rather than having to change each widget individually, as described in the last requirement.
- **Criticality:** This requirement is critical in that it allows multiple users to share the system with ease and efficiency. Without set profiles the users would spend far too much time having to organise which widgets they want every time they use the Smart Mirror.
- **Technical Issues:** This requirement requires setting up multiple interfaces loads, one for each user. In theory, this should not be too tasking of a task. However, linking these profiles to automatically login through the Google account on the Android application could prove difficult and requires more research.
- **Dependencies with Other Requirements:** This requirement depends on the previous two dependencies as it runs on the Smart Mirror interface and is loaded via the Android application.

3.4 Enabling code through Movement Detection:

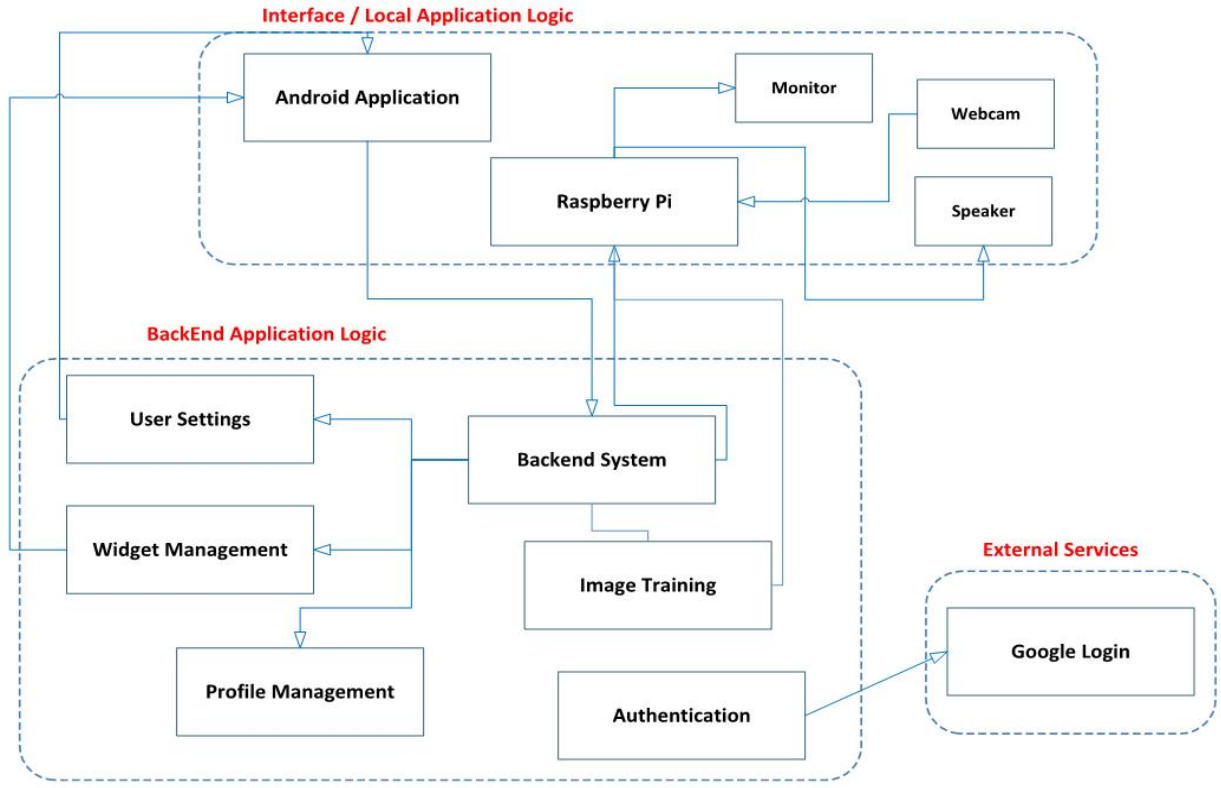
- **Description:** This requirement enables the Smart Mirror interface. When the camera detects movement, it will begin powering up the interface so that it loads up the currently assigned configuration. If the user logs into the mirror before entering the room, the interface will load their profile upon bootup. This allows for the interface to not always be enabled which will save power to the Raspberry Pi as well as computational power.
- **Criticality:** This requirement is critical in that it saves valuable computational power for the Raspberry Pi, which is already rather limited. Without this functionality, the Smart Mirror interface would always be turned on.
- **Technical Issues:** This requirement requires the correct profile to be loaded upon boot. This could prove quite difficult if motion is triggered and then the user logs in to his profile. While this sounds simple, the implementation in code could prove tricky.
- **Dependencies with Other Requirements:** This requirement depends on the first requirement mainly. As without the other two, it could still boot the interface, just without the ability to change the configuration. However, without the first requirement there would be no interface to load.

3.5 Facial Recognition to Load Profiles:

- **Description:** This requirement allows for the correct profile to be loaded to the interface based solely on the user standing in front of the Smart Mirror. Once motion is triggered the camera will start the facial recognition software and once the user is detected the interface will load the user's pre-set widget configuration.
- **Criticality:** This requirement is the least critical of the five in that while being a perfect feature for a system such as this, it is not absolutely essential for the system to run. The same effect can be obtained through the Android application, although it will be more time consuming and less efficient and natural for the user.
- **Technical Issues:** This requirement requires potentially the largest area for technical issues to arise, as running facial recognition through a rather primitive open source library such as OpenCV will likely not be as reliable as the industry standards we have come to expect from today's tech. Enabling the software to "learn" the user's face will require a lot of research and effort in order to be confident in the quality of prediction.
- **Dependencies with Other Requirements:** This requirement depends on each of the above requirements, given the way that it will be implemented in this system (triggered by movement). However, it could work by always checking for faces, instead of only triggering upon movement, although this would be far less efficient.

4. System Architecture

4.1 Component Diagram



- **Local Application Logic**

This section covers all the high-level interactions between each physical component and the application itself. The Android application receives data from “User Settings” and “Widget Management” in order for it to have up-to-date information on the current state of these components; whilst it also sends any changes to these values to the “Backend System” so that it can update the user’s preferences as they alter it. The Raspberry Pi will receive data from the webcam in the form of a stream of images, and will also output to the monitor for the Smart Mirror interface, and the speaker for any sound. The Raspberry Pi also receives data from the “Backend System” for the current user data, and from the “Image Training” component which is the trained images of the user.

- **BackEnd Application Logic**

This section represents the different relationships between the lower level components of the system. Here the “Backend System” acts as the link between the Local and Backend sections. The “User Settings” module allows the user to edit settings such as turning off the interface when a second person is detected. The “Widget Management” component allows the user to change which widgets they want to display on the interface along with their orientation on the screen. The “Profile Management” component is where the user can logout of their profile and where they can delete their account from the system. “Image Training” is the Python code which will train the user’s images so that they will be recognised when they step in-front of the Smart Mirror. “Authentication” allows the user to login to their Google account.

- **External Services**

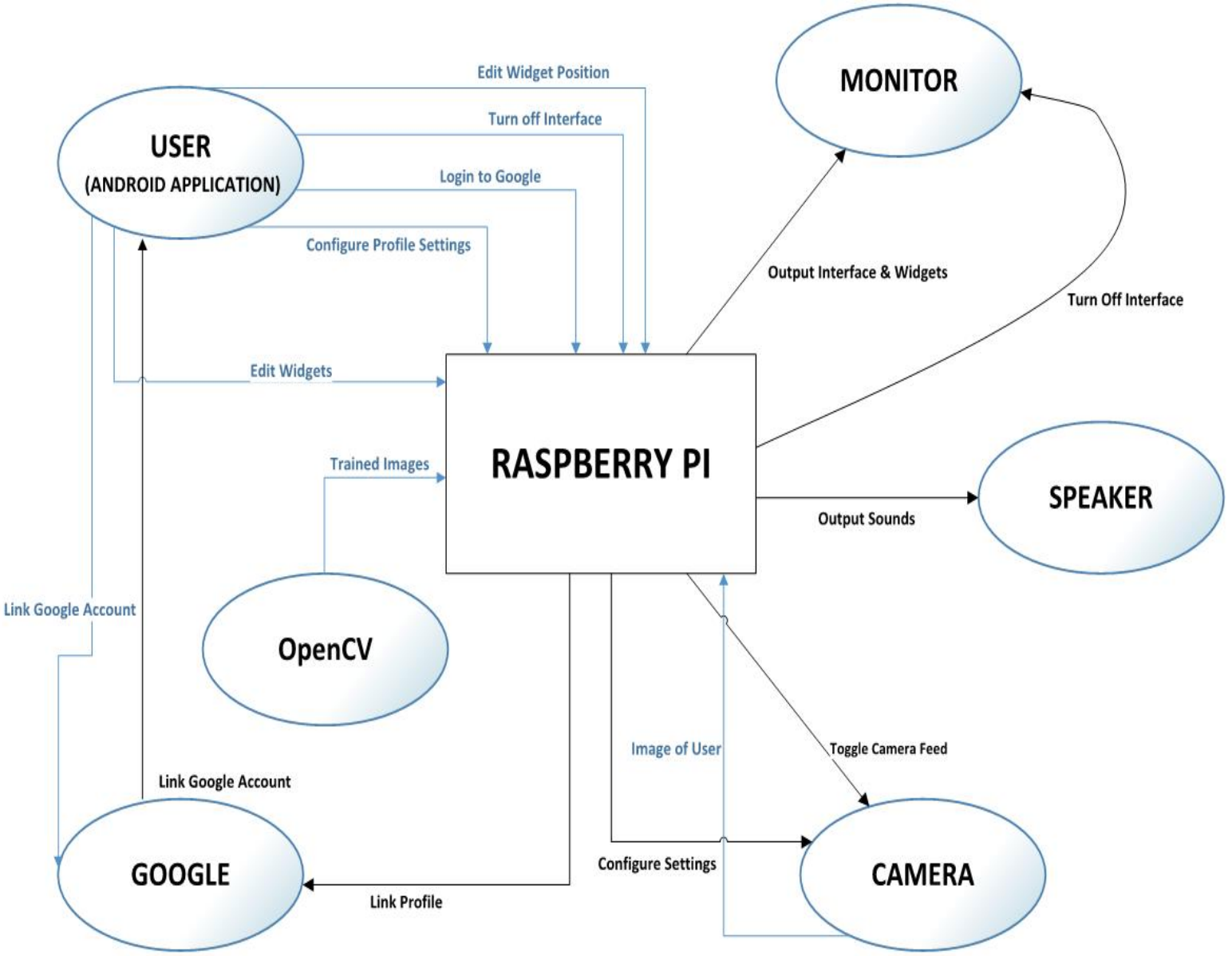
This refers to the 3rd party component used in this system, that of Google’s login service.

5. High-Level Design

5.1 Context Diagram

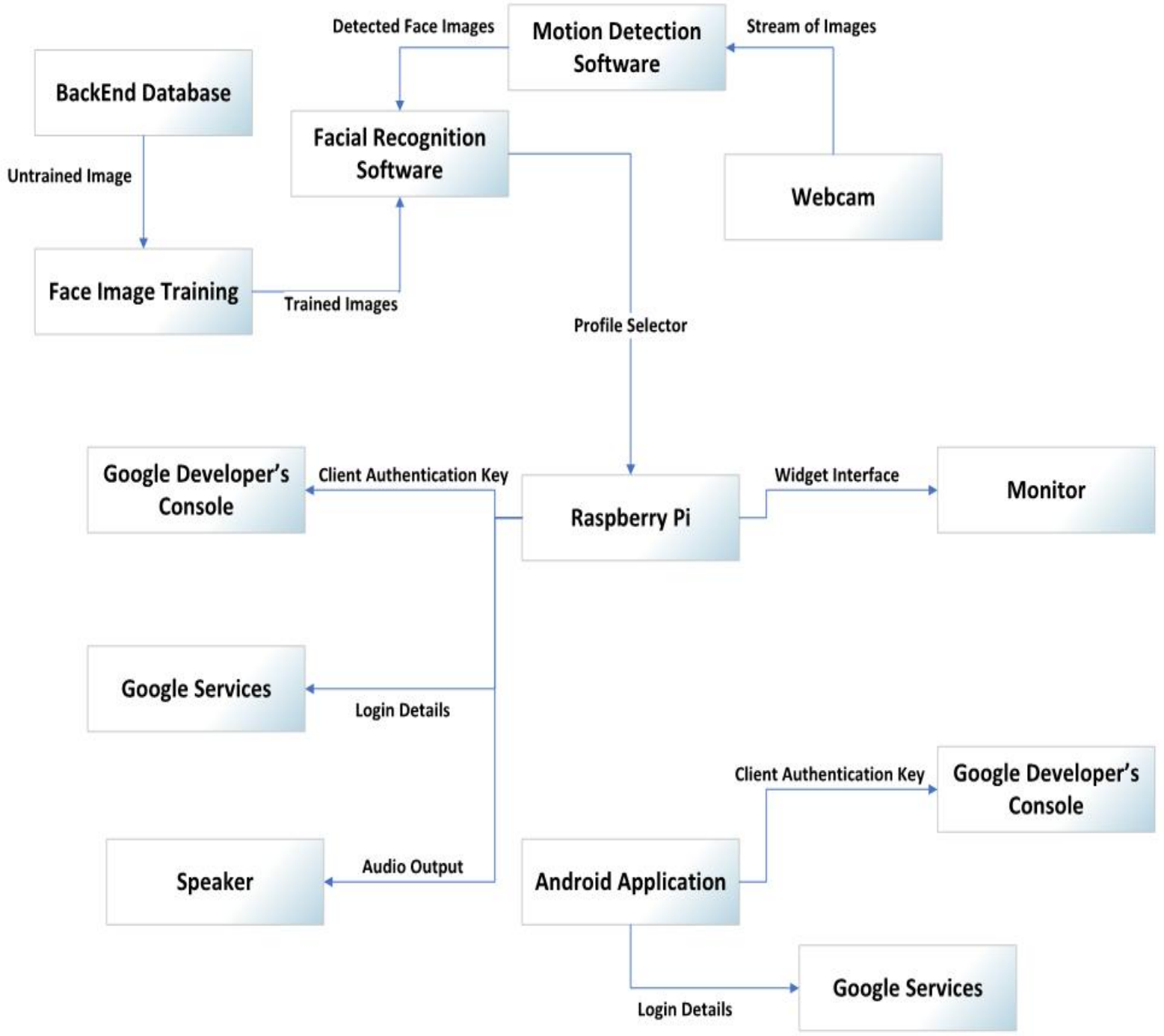
Below is a basic context diagram illustrating how our system will be connected and what data will be transferred. This data is displayed in more detail in the Data Flow Diagram that follows. This diagram shows a good visualisation of which parts of the project must be linked to each other, and so is perfect for the first step of the high-level design.

The diagram shows how the Raspberry Pi interacts with the external hardware and software that will be part of the system development. The only real interconnection outside of the physical hardware of the system will be to Google's authentication services, and the code used for training the images for facial recognition.



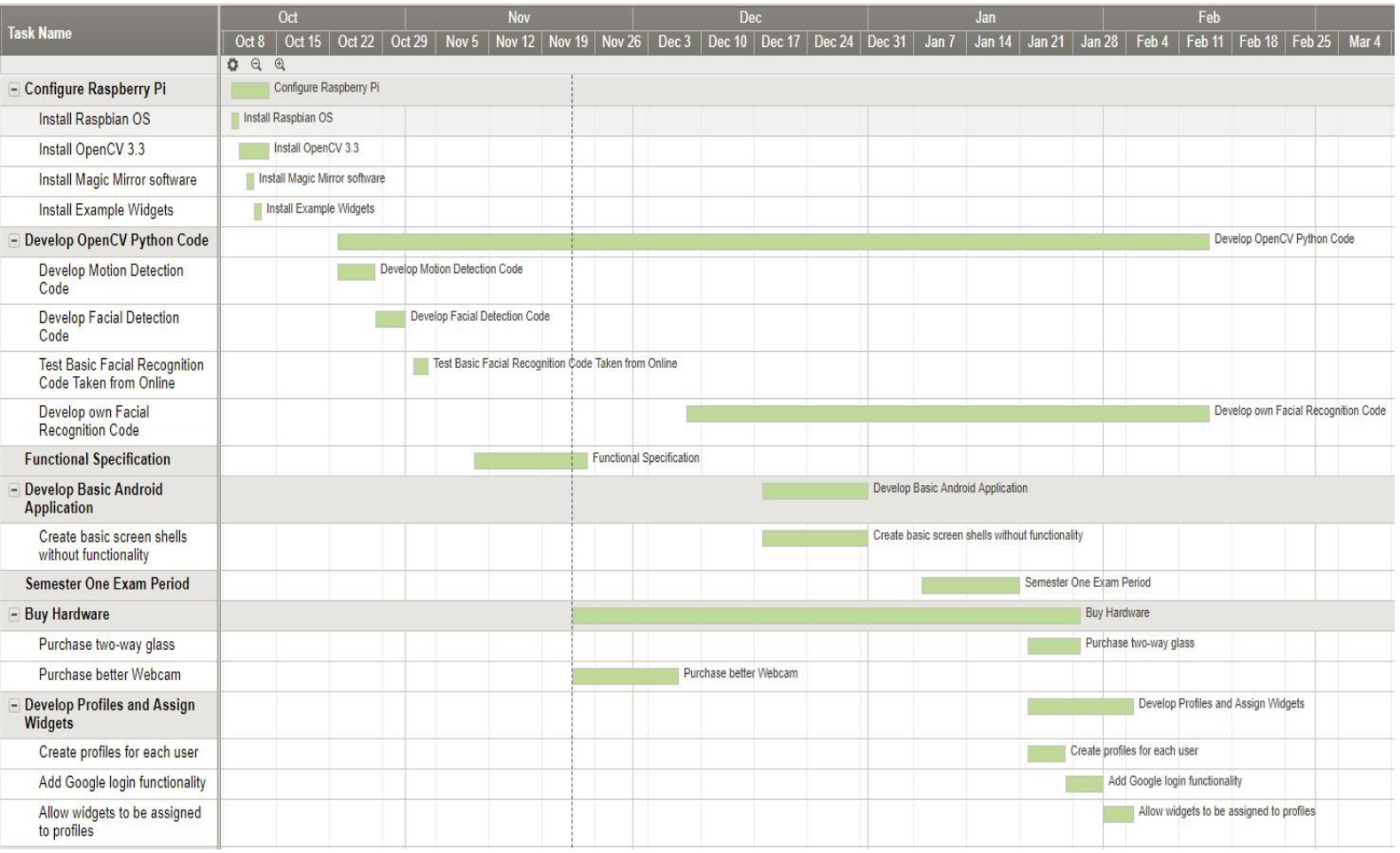
5.2 Data Flow Diagram

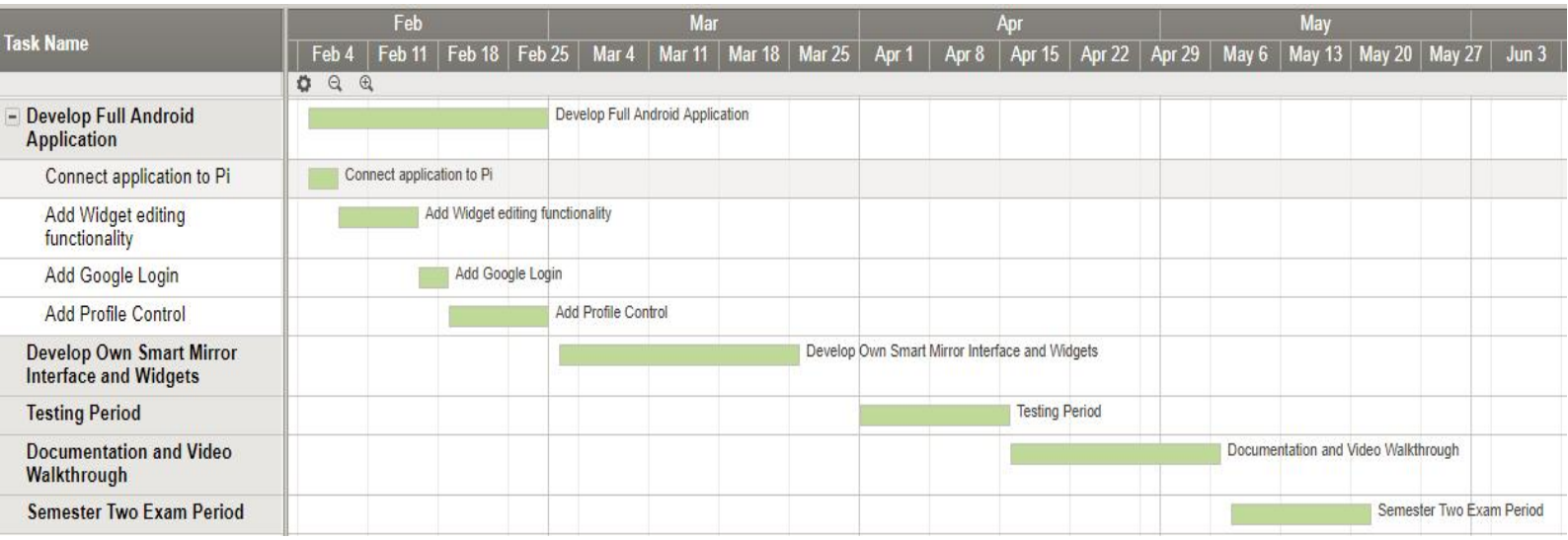
The Data Flow Diagram documents the various pieces of data sent between the components of the system, in much more detail than the previous diagrams. Below is a complete overview of the system's expected components and the data they must send/receive for the system to complete its functions.



6. Preliminary Schedule

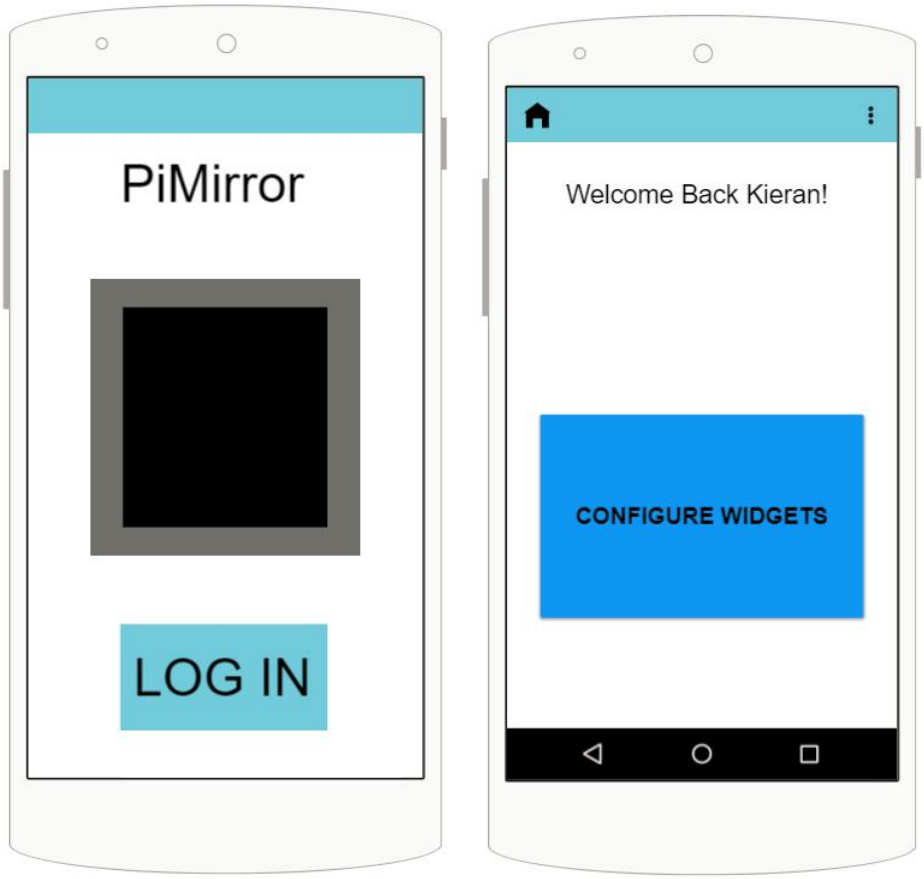
GANTT chart can be viewed at the following URL in full detail: "https://app.smartsheet.com/b/publish?EQBCT=1f5dc52b00f14f3e8769883f38bc8555".
Below is the exported version of the GANTT chart, created at smartsheet.com.

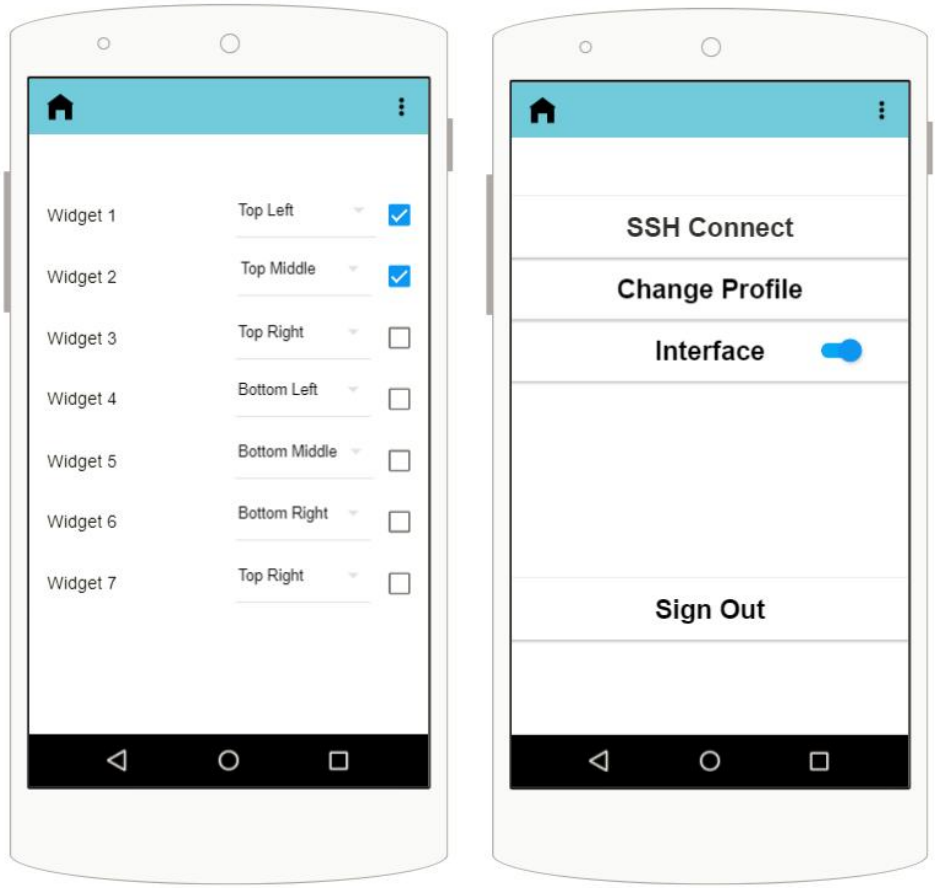




7. Appendices

7.1 Android Application Prototype Design





Shown above are four prototype potential designs of the Android Application. This is simply to get an idea of how the functionalities of the application will be laid out on the screen. The screens above show the following; (starting in the top-left moving across);

1. The Login screen which will be shown upon entry to the application.
2. The Home screen, shown after login, which allows you to configure your widgets via the main button in the centre of the screen.
3. The Widgets configuration screen, which allows you to select which widgets you want from the available list, and also change the location that they are displayed on the interface through the drop-down buttons.
4. The settings screen, allows the user to connect to the Raspberry Pi through the SSH connect button. Change the profile, and turn the interface on or off. It also gives the user the option to sign out of their profile.

7.2 Potential Problems

There are many potential problems that could occur/have already occurred during the duration of the project's development.

- **Installing OpenCV correctly:**

Installing the correct versions and modules needed for a project has proven to be a notoriously difficult task for a lot of people. Knowing this before starting the project allowed me to focus on getting this working first. This problem did occur already as although OpenCV was correctly installed on the Raspberry Pi; certain facial recognition methods and algorithms, that were used to test everything was working, failed. This problem alluded to the "face" module of OpenCV, which for no discernible reason, could not be installed on the version of OpenCV that was installed. After not being able to find anything online, with all documentation stating that it should be able to be installed, the newest version of OpenCV was installed which, after a complete wiping of the Pi, was able to successfully install the "face" module. This problem is therefore considered solved for now.

- **Facial Recognition effectiveness:**

One very possible potential problem that could occur is the effectiveness of the OpenCV algorithms to successfully predict the correct user. With this being a fairly primitive facial recognition method, it is not expected to be of cutting-edge industry standard. But the final effectiveness of the prediction should be of a reasonable accuracy, given the smaller sample size of users this system caters for.

- **Tricking Facial Recognition:**

Another very possible problem is that this method of facial recognition could potentially be tricked far easier than most industry standard methods. Again, this is due to OpenCV being a fairly primitive implementation. Events such as holding up a good quality picture of a user may be able to trick the system, but these problems are not realistically preventable given the hardware and software limitations, and also the limited timeframe for the project.

- **Training images on the Raspberry Pi:**

Training the images on the Raspberry Pi itself may not be possible due to the limited computational power of the device. This will mean that the facial recognition software will have to be trained on an external laptop or desktop computer, and then deployed onto the Pi. As of writing this specification, computations have been run for three users, consisting of two pictures each. So far, the Pi is able to handle this sort of computation but it is likely this will have to be accounted for as the user and image sets grow larger.

- **Speed of interface bootup:**

Again, due to the limited computational power of the Raspberry Pi, the speed in which the Smart Mirror interface initialises and boots up may not be an optimal speed for a cutting-edge product. The extent of this delay remains to be seen, but is not considered a major problem for the scope of this project. This problem is one that would likely be fixed with further development, after the system is shown to be able to work as advertised, with better hardware components.

- **Assigning profiles:**

Assigning user profiles with set widget configurations is something that is a main part of this project as it allows ease of use for users, particularly in cases where multiple users are sharing a single Smart Mirror device. Potential problems lie with swapping between these profiles smoothly, and loading/removing the correct data from the interface. Another roadblock, again, could be the speed at which this occurs, but this is a lesser problem.

7.3 Challenges and Learning Requirements

- **OpenCV documentation difficult to traverse:**

While OpenCV is a great resource for this project considering its open-source nature, this can also make it rather difficult to find the right documentation. Considering the number of changes that happen with each version release, which sometimes can render old functionality as obsolete, ensuring that something will work on the version currently installed can be tricky and time-consuming. This is also true for troubleshooting problems that occur within the system.

- **Python:**

Python is a language that has not been formally taught within the Computer Applications course for this class. Therefore, the limited knowledge personally gained in this programming language so far has been through self-experimentation and self-learning. This knowledge will have to be developed further when developing this project. The goal is to take knowledge from the other previously taught object-oriented programming languages, and apply similar methodologies to Python.

- **Facial Recognition algorithms:**

This part of the project is something that is entirely new, in that there is no prior knowledge or experience in any form of machine learning or facial recognition for the developer of this project. This will likely be the main learning challenge during the development of this system. This is the area into which most of the early research went into, before even choosing this project as the final idea. Choices have been made early, from this research, with regards to the algorithm to be used. Also, early tests have been conducted as mentioned above in “Training images on the Raspberry Pi” in the “Potential Problems” section. This has also been alluded to in the preliminary schedule.

- **Android application:**

For the Android application, there will be learning challenges with regards to the actual development of the application, with which the developer has limited knowledge of, although that Android applications are written primarily in Java should be of benefit given the number of modules taught on this language. Another learning requirement comes in how to connect to the Raspberry Pi, and edit the widget configuration from the application. From research, given the OS that is running on the Pi, the current solution is to simply SSH into the Pi’s IP Address, and edit tokens within the code relating to the widgets, through SSH commands.

7.4 References

Raspberry Pi 3 Model B: “https://www.raspberrypi.org/products/raspberry-pi-3-model-b/”

OpenCV – Version 3.3: “https://opencv.org/opencv-3-3.html”

OpenCV Local Binary Patterns: “https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html#id27”

Magic Mirror website: “https://magicmirror.builders/”

Raspberry Pi – Compatible webcam list: “https://elinux.org/RPi_USB_Webcams”

Two-Way Glass: “https://www.twowaymirrors.com/”