Contents

Project Progression Record

Level 3 Extended Project Qualification

This form should be completed by the learner

Name: **Joon-Ho Son**	Centre: HRSFC
Candidate no: **1969**	Centre no: 22147

Current programme of study: A Levels		
Qualification Type GCE	Level A	Subject **Maths** **Further maths** **Physics** **Chemistry** **Biology**

Before starting this form you should have already explored various themes and ideas for your project and proposed a suitable topic to your mentor/supervisor

Date Project started: **19/01/16**
Project Topic: **The Smart Mirror: A demonstration of the Internet of Things**
Rationale: *why you want to do this project and how it links to progression plans/current courses/personal interest* **This project involves elements of electronic engineering, programming and woodworking, which are all potentially transferrable skills that could apply to my future career. Not only does this, but I will also (hopefully) end up with a useful product at the end of the project.**
Aims and Objectives: *What outcome(s) do you hope to achieve. What skills are you aiming to develop?* 1. **Learning a new programming language** 2. **Woodworking skills** 3. **Time management** 4. **Learning app development**

Initial Planning

What range of resources might you use/what types of research might you complete……e.g. specialist materials/workshop space. Books/academic journals etc. **Workshop for cutting and sanding wood, and cutting acrylic/glass.** **Online documentation and resources.** **One-way mirror.** **An android phone or Raspberry Pi or similar.**
What factors might you need to take into account when planning how to manage your project? **Which components should be prioritised to maximise time efficiency.** How will you plan to bring your project in for the deadline? You should create a timeline. **Create a Gantt chart, plan precisely and in detail.**
Supervisor Signed Date

Mid Project Review

Date: 21/6/16

How much progress have you made in meeting your outcomes? **I am almost entirely on schedule, according to my timeline.**
How has your project changed or evolved? **I have decided to create a hybrid app instead of a native app as originally intended. This has numerous advantages which I discovered during my research.** What have been the strengths/weaknesses with your project management so far? **Misjudging the time that would be required for exam revision.**
How are you evidencing the research you have done and how you are managing your project? **I am creating an extensive bibliography, referencing every source used.**

What do you still need to do? **The bulk of the features of the app are still yet to be fully implemented. I have not yet begun the construction of the mirror itself. However, I expect to progress rapidly from this point onwards.**
Supervisor Signed: Date:

End of Project

Where is the evidence within your project that demonstrates:	
Outcome	**Evidence** – where can it be found? E.g. log book, folder etc.
Exploration of a range of ideas for your project	**Introduction.**
How you have managed your time	**Timescales 1 & 2 (Gantt diagram) and planning.**
Use of a wide range of research and resources	**References, citations throughout.**
Evaluated research and selected appropriate research	**References and Source evaluation.**
How you made decisions and solved problems	**Design, Development, Construction, Evaluation, Diary.**
How you used technology – where appropriate	**Timescale, Design, and Development.**
Presentation given to an audience	**A comprehensive review and commentary on the marketplace is included in the Evaluation.**

Making links to HE/Career	**The artefact subject itself is strongly linked to HE. In particular see the Development section where programming is documented and Evaluation to see reflection on skills learned.**
How you have reflected on your project management skills and the quality of your outcome.	**Evaluation, Planning & Timescale 2.**
On-going evaluation	**Post-exam review. Planning & Timescale 2, PPR.**
Final Evaluation	**Evaluation.**
If not completing a dissertation – 1500 word supporting statement	**Design, Development and Construction provide documentation of my process.**

Date of Submission:
Student Signature:
Supervisor Signature:

This form should be used to record the progress of each learner.

A copy of this form must accompany each learners work when it is submitted for Moderation

Introduction

Project Ideas

Before settling on this idea, I thought about other project titles that would be a combination of my interests and relevance to my university application.

One idea I initially considered was building a mechanical claw machine, not unlike the ones seen at arcades. I thought this would be an excellent opportunity to explore engineering in more detail, also requiring some programming knowledge to program the claw's actions. However, in the end I decided that the idea was too expensive and I would prefer a project with a greater focus on Computer Science.

An alternative approach I thought about was a dissertation on Artificial Intelligence - inspired by Nick Bostrom's book *Superintelligence* which I had read some time earlier. This would have been a dissertation style EP and it would have discussed topics such as what we mean by *"AI"*, how we can achieve such a concept, and the potential implications of such a concept (ethical, political, existential). This would have given me the opportunity to explore a concept I am deeply interested in and would strongly support my application. In spite of this, I ultimately felt that I wanted to undertake a project that had more emphasis on a practical programming aspect.

Rationale & Background

The Internet of Things is a term used to describe pervasive and ubiquitous computing, a world where the environment *is* the interface. This brings benefits, but also caveats. Redesigning architecture, but more fundamentally, the way that we think. Often we find ourselves reaching to check the weather forecast for the day; ten minutes later and after checking every social media platform possible we have entirely forgotten the reason for unlocking our phone. Today, the ability to read data through the sensory overload of noise is a vital life-skill – and has almost become second-nature to current generations.

My project is a way for people to passively acquire critical information whilst minimising distraction by providing focused access to information. The product is a wall-mounted mirror with a display to show relevant data such as the weather, time, news, and more. The Smart Mirror is a means to disconnect ourselves from the *"distraction economy"* of modern society without sacrificing productivity.

Not only this, but I also have a secondary goal. Even living in a world dominated by technology, I find that most people have a very limited idea of what "coding" or "programming" is. I hope that by documenting my process I will somewhat demystify how a software developer thinks, what considerations that they must take into account when creating a product, and the tools that they use to make their ideas come to life.

Initially, I drew inspiration from an article [1] in the International Business Times. This was an open-source project created by Microsoft under the name of *"Magic Mirror"*, available to access publicly on GitHub. Unfortunately, the code base itself would not be much use to me as it a UWP Hosted Web App which is a platform that requires some proprietary software and would not run on any devices I own. However, some of the core concepts I used later.

I will begin by researching what features and functionality would be most essential to the device, then study how best to implement this functionality and prioritise these goals in order of importance and achievability. I will be producing documentation alongside the development cycle in order to log my methods and progress. Finally, it would be useful to write a reflective evaluation so that any future projects of mine can be completed to a greater standard, learning from any unforeseen problems encountered in this project.

Throughout, I will be using the IEEE referencing style as is standard in Computer Science. All artwork is mine unless stated otherwise.

Design

Hardware

After some rough sketches, I drew up some drafts of some potential designs. At this point I didn't know the exact dimensions of the mirror as I had not acquired the necessary components yet (LCD, Mirror, etc.), so I chose some arbitrary dimensions in the ratio 16:9 (the most common aspect ratio for TVs and monitors).

I listed some potential solutions, which I will not expand on for brevity.

- Raspberry Pi
- Android TV
- Android tablet
- Google Chromecast
- Amazon Firestick

After considering several designs, I decided that the most convenient way to tackle the "smart" component is via an Android phone which will be beamed to a TV via Chromecast. I feel like this is the most flexible and least restrictive choice. The fact that it works via a phone means that hardware purchases are kept at a minimum. Moreover, instead of using a dedicated controller like a Raspberry Pi, which implies a permanent installation, a streaming device can also be used for more general media purposes, like displaying photos.

Here is a labelled exploded-view diagram of what I visualise to be the final product, created using a program called "*Blender*".

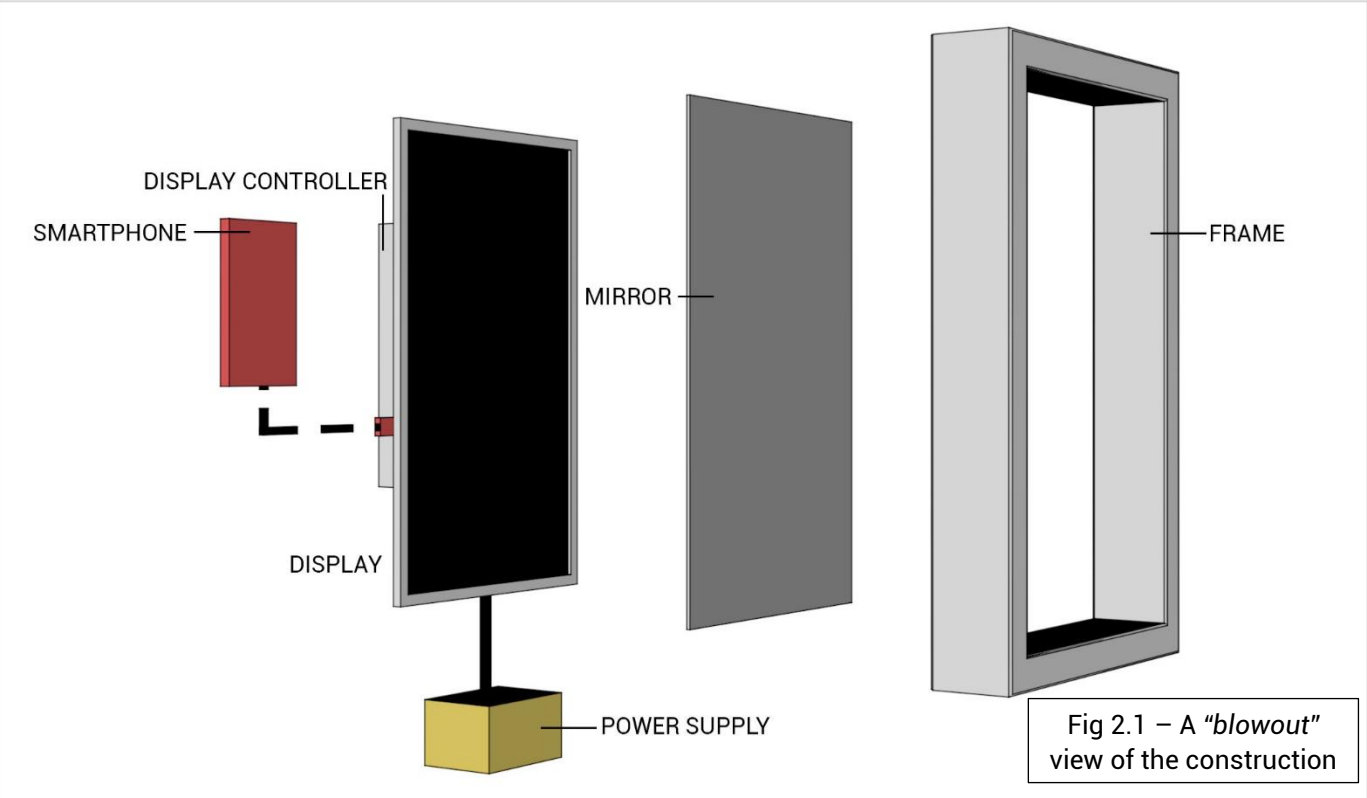
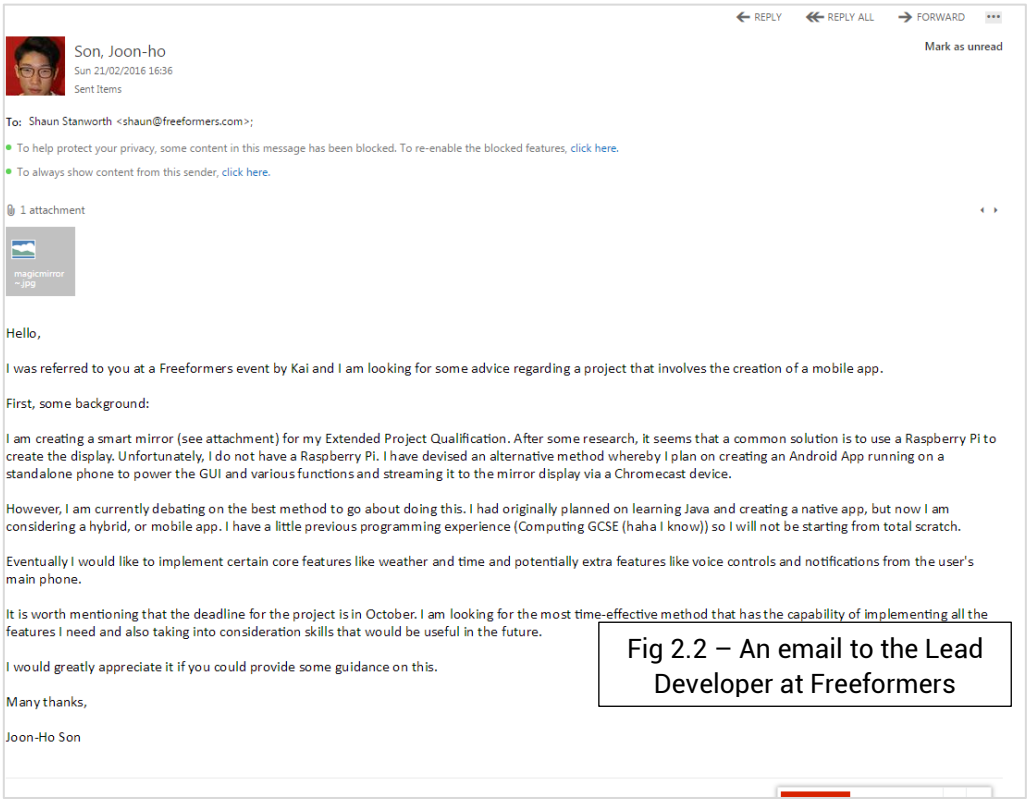


Fig 2.1 – A "*blowout*" view of the construction

Software

Initially, when I was first drafting this project I had intended to learn Java and native Android programming. An Android app running on a phone would power the display. However, in February I attended a *Freeformers* hosted "Learn to Code" event where I learned about alternatives to native app development: Mobile and Hybrid apps.

These involve using HTML, CSS and JavaScript technologies used for **webpages** instead of Java to create a mobile app. A drawback of mobile apps is that, because they are essentially a website, you must be connected to the internet to use them. Hybrid apps solve this by converting a mobile app into something that can be installed and used offline. I did some research on the pros and cons of each type and then eliminated those which were not relevant to this particular project [2]. I also got into contact with the Lead Developer at Freeformers to ask his advice.



REPLY REPLY ALL FORWARD

Mark as unread

Son, Joon-ho
Sun 21/02/2016 16:36
Sent Items

To: Shaun Stanworth <shaun@freeformers.com>;

To help protect your privacy, some content in this message has been blocked. To re-enable the blocked features, click here.

To always show content from this sender, click here.

1 attachment

magicmirror - jpg

Hello,

I was referred to you at a Freeformers event by Kai and I am looking for some advice regarding a project that involves the creation of a mobile app.

First, some background:

I am creating a smart mirror (see attachment) for my Extended Project Qualification. After some research, it seems that a common solution is to use a Raspberry Pi to create the display. Unfortunately, I do not have a Raspberry Pi. I have devised an alternative method whereby I plan on creating an Android App running on a standalone phone to power the GUI and various functions and streaming it to the mirror display via a Chromecast device.

However, I am currently debating on the best method to go about doing this. I had originally planned on learning Java and creating a native app, but now I am considering a hybrid, or mobile app. I have a little previous programming experience (Computing GCSE (haha I know)) so I will not be starting from total scratch.

Eventually I would like to implement certain core features like weather and time and potentially extra features like voice controls and notifications from the user's main phone.

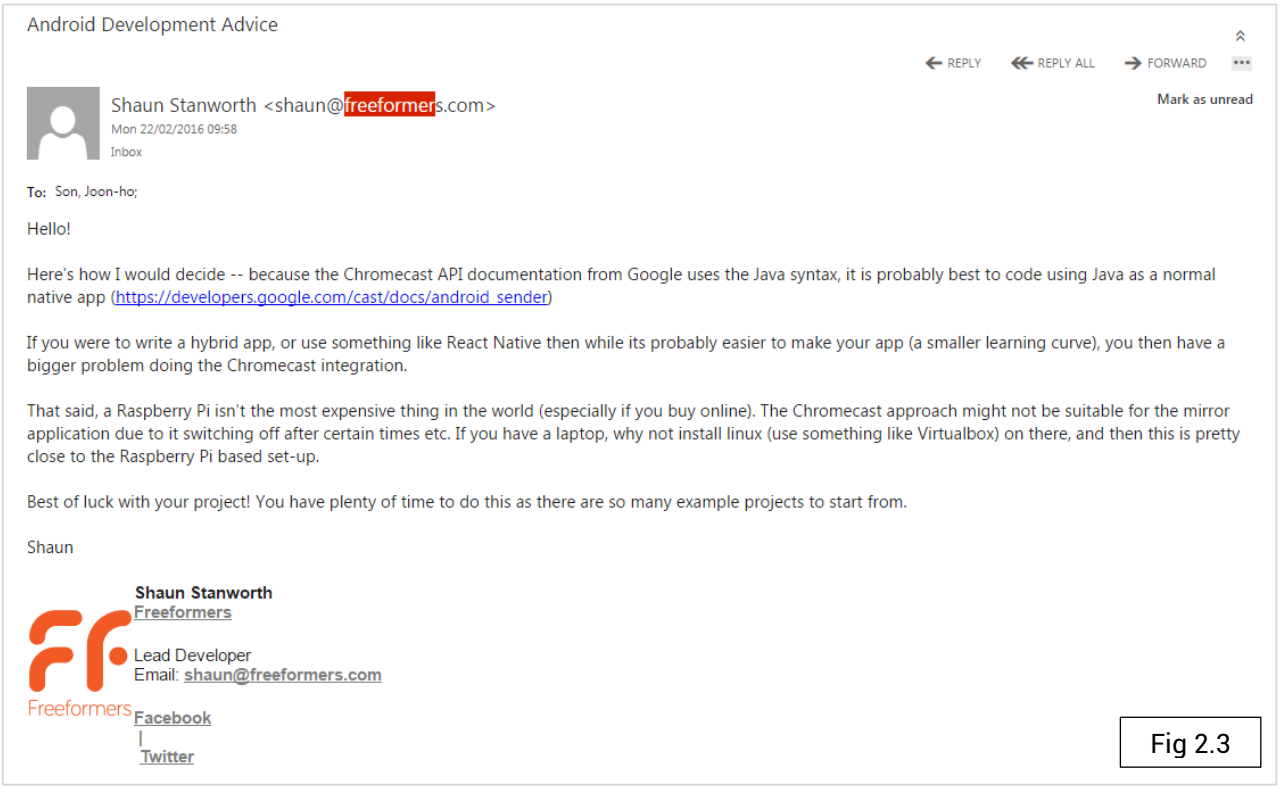
It is worth mentioning that the deadline for the project is in October. I am looking for the most time-effective method that has the capability of implementing all the features I need and also taking into consideration skills that would be useful in the future.

I would greatly appreciate it if you could provide some guidance on this.

Many thanks,

Joon-Ho Son

Fig 2.2 – An email to the Lead Developer at Freeformers



Android Development Advice

REPLY REPLY ALL FORWARD

Mark as unread

Shaun Stanworth <shaun@freeformers.com>
Mon 22/02/2016 09:58
Inbox

To: Son, Joon-ho;

Hello!

Here's how I would decide -- because the Chromecast API documentation from Google uses the Java syntax, it is probably best to code using Java as a normal native app (https://developers.google.com/cast/docs/android_sender)

If you were to write a hybrid app, or use something like React Native then while its probably easier to make your app (a smaller learning curve), you then have a bigger problem doing the Chromecast integration.

That said, a Raspberry Pi isn't the most expensive thing in the world (especially if you buy online). The Chromecast approach might not be suitable for the mirror application due to it switching off after certain times etc. If you have a laptop, why not install linux (use something like Virtualbox) on there, and then this is pretty close to the Raspberry Pi based set-up.

Best of luck with your project! You have plenty of time to do this as there are so many example projects to start from.

Shaun

Shaun Stanworth
Freeformers

Lead Developer
Email: shaun@freeformers.com

Freeformers
Facebook
|
Twitter

Fig 2.3

As I am not creating a conventional handheld app and it is a personal project, factors such as the ability to monetise apps by submitting them to an app store or being effective over multiple platforms are not a concern. Essentially, it comes down to if I prefer web development or native development. I found that

hybrid app development provided a comfortable compromise between being able to utilise the full capability of the device, the requirement of more general web developer skills, and reduced latency function over HTML5. This fact also means that the app can be used across most platforms, including a browser. The implication being that if you really wanted to use a Raspberry Pi, porting it would require little extra work. Ultimately, I decided that advantages of hybrid development outweighed native development, especially given the timescale and taking into consideration the skills I would develop.

There are definitely some obvious drawbacks with this design which I have considered, such as the fact that the phone must be actively connected to the display, the implication of which is that other apps cannot be used at the same time. However, I don't see this as a major concern for the reason that this project is primarily a proof-of-concept. Furthermore, hybrid app development means that it can very easily be ported to a platform that uses a dedicated controller i.e. not your phone.

There are a plethora of resources available both online and offline for learning web development. After a significant amount of research I compiled a list of those which I think best suit my needs.

Possibly one of the most popular online resource for learning programming is **Codecademy** [3]. Codecademy is a great, interactive tool to learn programming and provides courses for a large range of programming languages and even an entire section dedicated to web developer skills, encompassing basic HTML to Ruby on Rails web apps. Its interface includes an in-built code editor, and changes are shown in real-time.

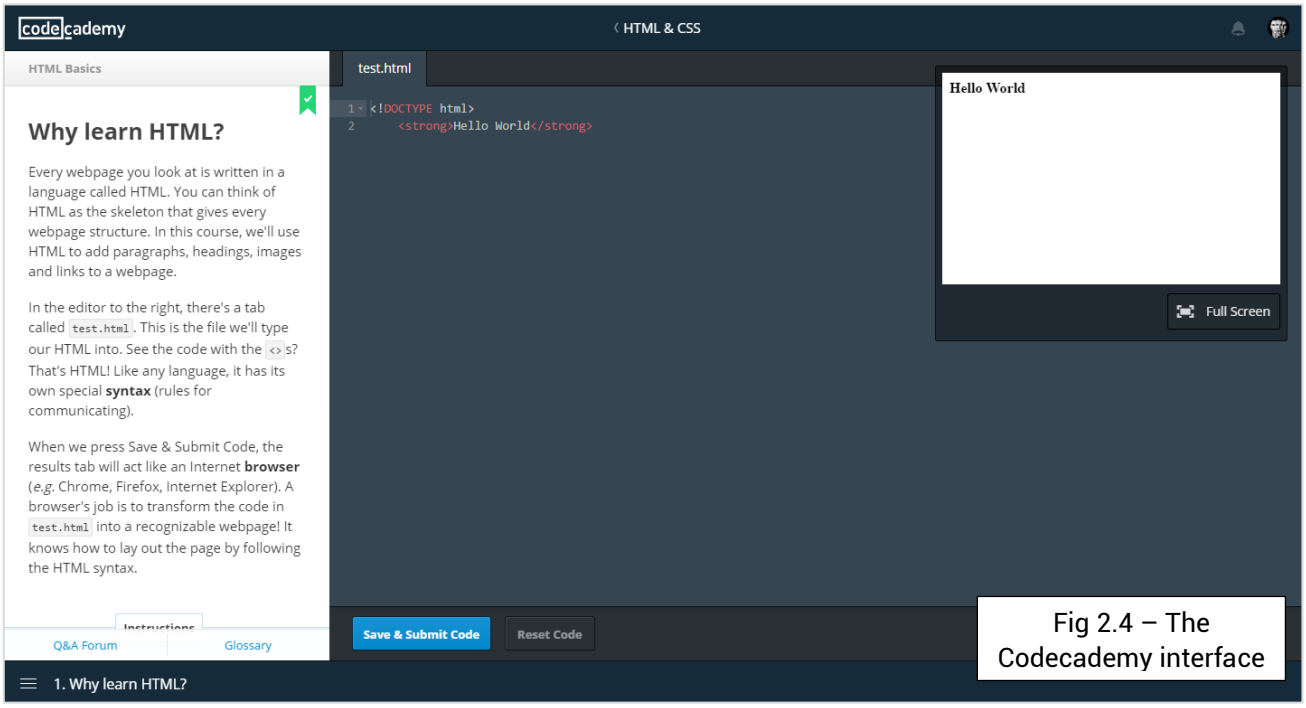


Fig 2.4 – The Codecademy interface

While Codecademy is very useful for getting to grips with the basics, there are many other ways for an aspiring programmer to learn. For one, it would be a cardinal sin in the software development community to neglect to mention **Stack Overflow** [4]. Stack Overflow is an incredibly popular community-moderated question and answer based forum for programmers. The true value of the website comes from the incredible breadth of answers available, and often it seems that every possible niche bug and error has already been covered in detail.

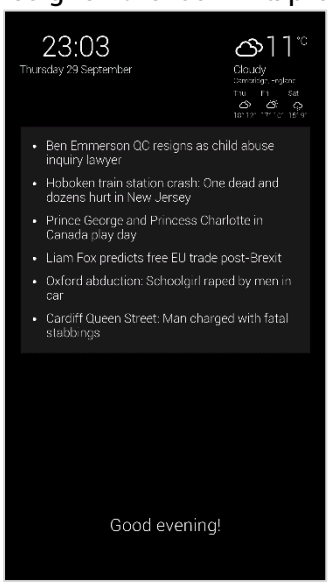
Fig 2.5 – An example of the stack**overflow** layout

I have chosen to not use any books as online media has every advantage against its offline counterpart. The most important instance being the fact that with the speed of development and changes made to the technological world it is inevitable that books will become outdated and of very little use very quickly – not to mention the cost.

Interface and Icon Design

While designing this interface I incorporated elements of both Apple's *Human Interface* [5] and Google's *Material Design* [6] guidelines. I thought the clean iOS design was a perfect match for the minimalist aesthetic of the mirror. Google's Roboto font provides a free and open-source alternative to Apple's San Francisco font, and the use of material design for the icon matches the rest of Google's unified Android look. I designed the icon in a program called "*Adobe Illustrator*".

Fig 2.6 – The icon I designed



To maintain minimum obscuration of the user's reflection I opted to keep the main view mostly empty, and then implement pop-up windows that will come up when the user asks, complete with slide-down and slide-up animations.. Here is an example of such a window.

Fig 2.7 – A window showing the news.

Planning 1

Before starting a project like this, it is important to have an idea of the timescales of each sub-task and to set yourself deadlines. This should be done before beginning any other aspect of the product. As well as tasks that need to be completed, I have included key periods, such as the examination period, during which my time to work on my EP will be severely limited. It is difficult to apply such a rigid structure to my project as I plan on approaching it with the *Agile* method (see development section) and this type of scope-bound sectioning is at odds to that methodology but it is still useful to draft an overarching plan. I feel that a Gantt chart is best suited to displaying this information.

Though I am aiming for the October hand-in date, in my mind the project must be essentially complete by the EP marketplace on 08/09/2016.

1. **Planning and Design**
   As part of this phase, I will draw up basic diagrams using CAD tools in order to further my IT skills and so I have an idea of what the final product will actually look like.

2. **HTML & CSS Basics**
   HTML and CSS are what hybrid apps are built from, and will be essential for my project. I have chosen to call this "basics" because improvement of programming skills are an ongoing process for any developer.

3. **JavaScript Basics**
   JavaScript is a very popular programming language used heavily in web development. Its role is to make things happen on the website, such as creating interactive dialogue boxes or displaying a slideshow of images

4. **Git Basics**
   Git is a version control software, essential for any development project, enabling features such as revision, merging and reverting (see development section for more on git).

5. **UI**
   UI stands for User Interface. It is essentially the interface that the user will see on the screen. I have given this a very long time period on my chart because it is a feature which is continuously adapted.

6. **Frame**
   A frame for my mirror needs to be constructed out of wood.

7. **Mirror**
   I will need to buy a one-way mirror that is either already cut to size or cut it myself.

8. **Time Widget**
   This is the most basic function of my mirror.

9. **Weather Widget**
   This is a core feature of my mirror. To be a useful device the mirror must display helpful information. At this point the app is what is known as the Minimum Viable Product (MVP).

10. **Additional Features**
    I have grouped these together as one item (as opposed to my core features, time and weather) because these are stretch-goals rather than primary targets. I will implement as many of these as possible with the remaining time I have, but not until my core features are complete. This is in accordance with my Agile development methodology. These additional features *may* include:

- Voice control
- RSS news feed
- Calendar/Agenda
- Compliment
- Face API
- Traffic information

Since this is the first time I have ever attempted anything like this before, it difficult to put an estimated completion time on these features, so I have just allocated as much time as possible.

11. **Examination period**
    From the end of Easter to the end of the exam period I will be putting my project on a hiatus in order to concentrate on revision.

12. **Holidays**
    These are periods where I hope to make more progress than a normal working week. This is with the exception of the half-term during the examination period during which I will be studying.

Codecademy courses are structured into units and lessons. For example the HTML & CSS course is as follows:

UNIT 1: INTRODUCTION TO HTML

- HTML Basics
- Build Your Own Webpage

UNIT 2: HTML STRUCTURE: USING LISTS

- HTML Basics II
- Social Networking Profile

UNIT 3: HTML STRUCTURE: TABLES, DIVS AND SPANS

- HTML Basics III
- Clickable Photo Page

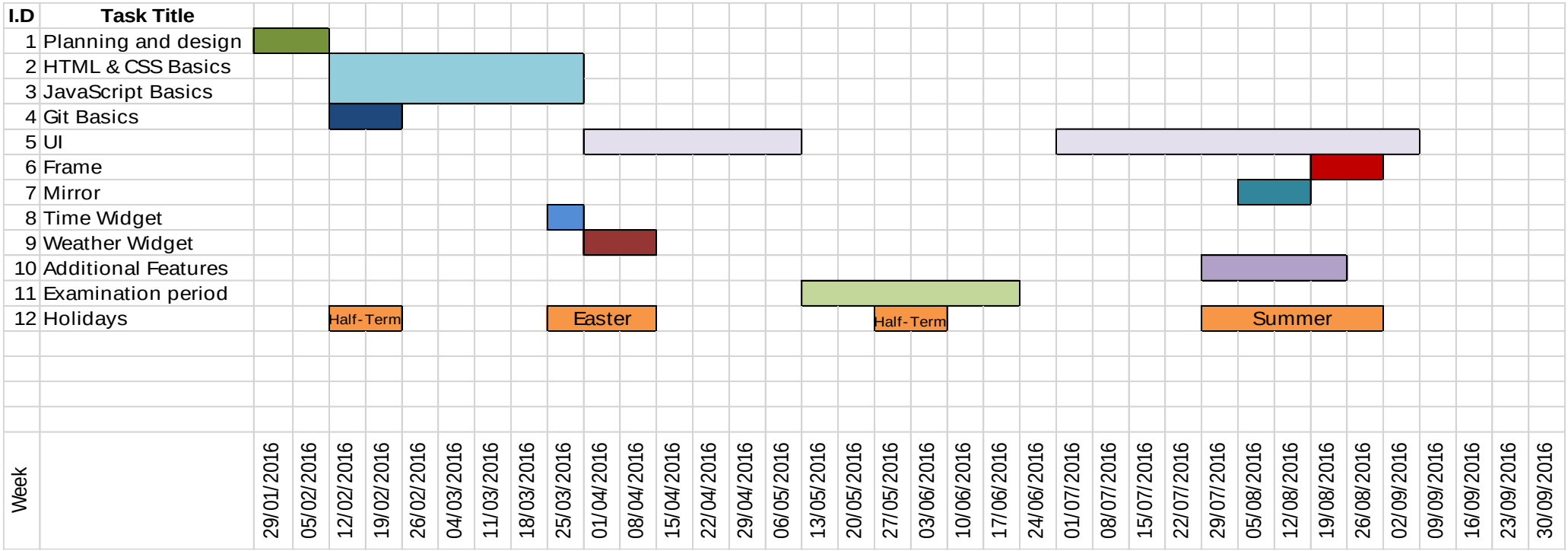
Etc.

During the normal school period I estimate that I will be able to complete 2 units of the HTML or JavaScript course every week. There are 6 units in the HTML & CSS course and 8 units in the JavaScript course. Therefore I shall assign 7 weeks to completing both the web and JavaScript courses. The git course is much shorter than the other courses should take just one dedicated week to complete as it is much shorter than any other so I'll spread the git course over 2 weeks in parallel to the other two courses. This is illustrated in my Gantt diagram.

I am also using a system of sticky notes on my wall which gives me a convenient way to keep track of features to be completed. Here is the state of the wall sometime midway through the project.

Fig 3.1 – Using sticky notes to track progress

Timescale 1


I.D
Task Title
1 Planning and design
2 HTML & CSS Basics
3 JavaScript Basics
4 Git Basics
5 UI
6 Frame
7 Mirror
8 Time Widget
9 Weather Widget
10 Additional Features
11 Examination period
12 Holidays
Half-Term
Easter
Half-Term
Summer
Week
29/01/2016
05/02/2016
12/02/2016
19/02/2016
26/02/2016
04/03/2016
11/03/2016
18/03/2016
25/03/2016
01/04/2016
08/04/2016
15/04/2016
22/04/2016
29/04/2016
06/05/2016
13/05/2016
20/05/2016
27/05/2016
03/06/2016
10/06/2016
17/06/2016
24/06/2016
01/07/2016
08/07/2016
15/07/2016
22/07/2016
29/07/2016
05/08/2016
12/08/2016
19/08/2016
26/08/2016
02/09/2016
09/09/2016
16/09/2016
23/09/2016
30/09/2016

Planning 2

(Post-exam)

1. **Planning and Design**
   The bulk of this phase was completed early on in the project, perhaps even a little earlier than suggested on my timeline. However, I have continued to add and develop this section as the project has progressed.

2. **HTML & CSS Basics**
   I completed the HTML and JavaScript courses in the week preceding Easter, so pretty much on time and in accordance with my timescale.

3. **JavaScript Basics**

4. **Git Basics**
   This was a very short course and was completed in the space of about a week.

5. **UI**
   I have a basic app up and running that can display HTML and JavaScript elements. However, due to exams there is little more than this to show. I have not yet begun to put different widgets together into a single interface, but rather have several separate test apps.

6. **Frame**
   As planned, I have yet to begin work on the construction of the mirror.

7. **Mirror**

8. **Time Widget**
   This was a very simple task to complete and was finished according to schedule, in early Easter.

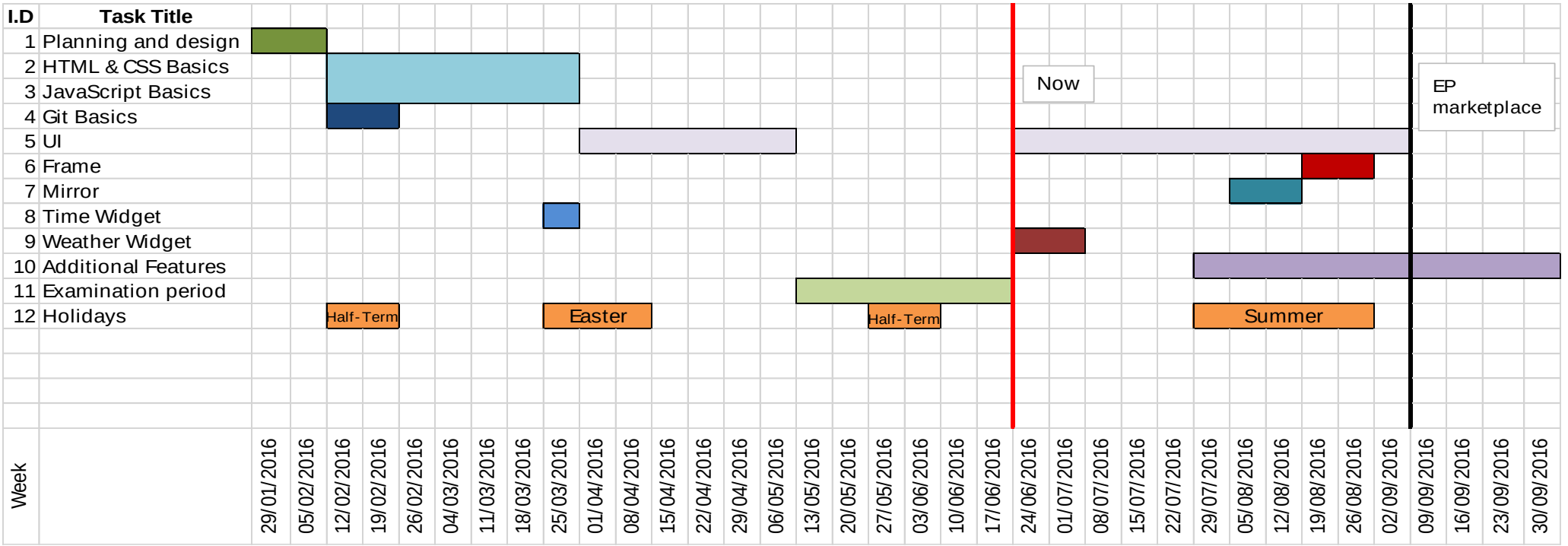
9. **Weather Widget**
   I had scheduled this to be worked on during the Easter however this was not possible due to revision.

10. **Additional Features**
    I have extended this period from the original timeline as it makes sense to keep improving the project right up until the finish date.

11. **Examination period**
    I misjudged when I would need to begin my revision which was short-sighted of me. So I have extended the examination period to include Easter which means the weather widget must be implemented post-examinations, much later than originally.

12. **Holidays**

Timescale 2


I.D
Task Title
1 Planning and design
2 HTML & CSS Basics
3 JavaScript Basics
4 Git Basics
5 UI
6 Frame
7 Mirror
8 Time Widget
9 Weather Widget
10 Additional Features
11 Examination period
12 Holidays
Half-Term
Easter
Half-Term
Summer
Now
EP marketplace
Week
29/01/2016
05/02/2016
12/02/2016
19/02/2016
26/02/2016
04/03/2016
11/03/2016
18/03/2016
25/03/2016
01/04/2016
08/04/2016
15/04/2016
22/04/2016
29/04/2016
06/05/2016
13/05/2016
20/05/2016
27/05/2016
03/06/2016
10/06/2016
17/06/2016
24/06/2016
01/07/2016
08/07/2016
15/07/2016
22/07/2016
29/07/2016
05/08/2016
12/08/2016
19/08/2016
26/08/2016
02/09/2016
09/09/2016
16/09/2016
23/09/2016
30/09/2016

Development

Development Methodology

For the duration of this project I will be taking a project management approach widely used in the software development industry called **Agile development**.

Agile development emphasises a living project which undergoes continuous improvement and evolution. At the core of its philosophy are what could be called the "tenants" of Agile: **adaptive**, **iterative** and **streamlined** [7]**,** meaning concise documentation is favoured over reams that lay out every detail.

This is at odds to something like the Waterfall model which promotes rigid, sequential phases [8] [9]. The principal danger with this system the possibility that a poor design choice will not be discovered until the final stages of development or testing. Consequently, if there is a change in the requirements, redesign, redevelopment and retesting is very costly. Instead, Agile demands flexibility as opposed to order [10].

The project is **time-boxed**, meaning it is planned by time instead of features. This involves allocating a fixed time period to each planned activity as opposed to **scope-boxing** which is the opposite. This is critical to my EP as one of the major aims of the qualification is being able to manage time effectively.

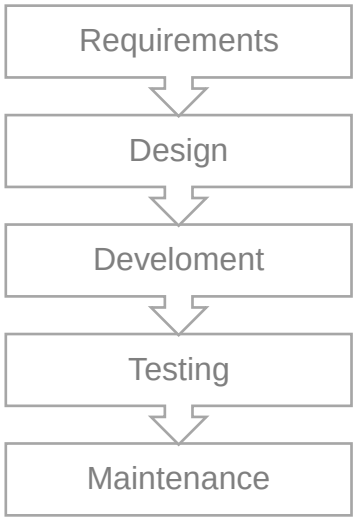


Fig. 7.1 – The Waterfall model

Git

Git is a free, industry standard version control system used heavily in software development. It provides support for non-linear workflows by creating "forking" **branches** that can be worked on independently without affecting the **master** branch and are eventually **merged**. In this way, conflicts can be avoided. This is especially useful when several different people, or even groups of people are working on the same project. It also enables me to manage all of my code in one repository.

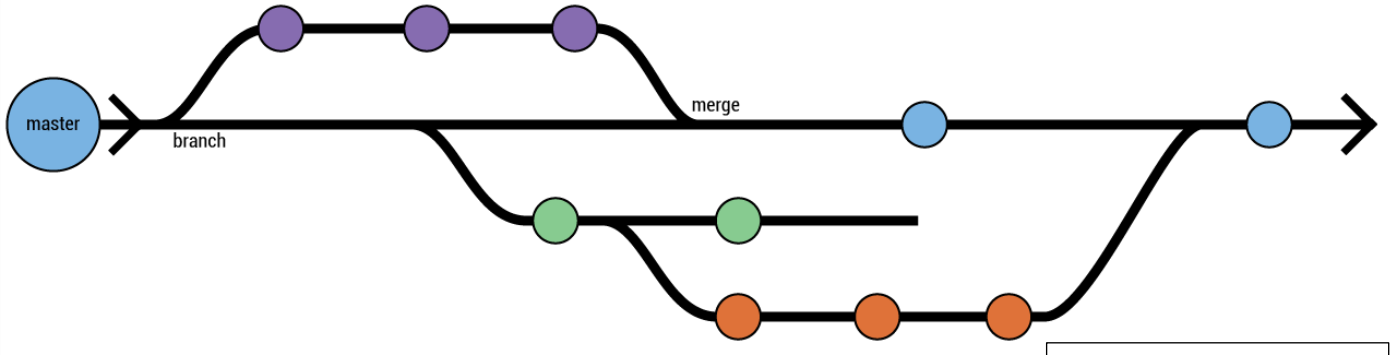


Fig. 7.2 – git branch and merge workflow

Github is an online project hosting service. This enables developers to upload their projects into a remote location so it can be accessed by other developers – whether they are part of the development team or independent. The uploaded projects are called **repositories**, or **repos** for short.

I used Codecademy and github's own tutorial, trygit, to familiarise myself with github.

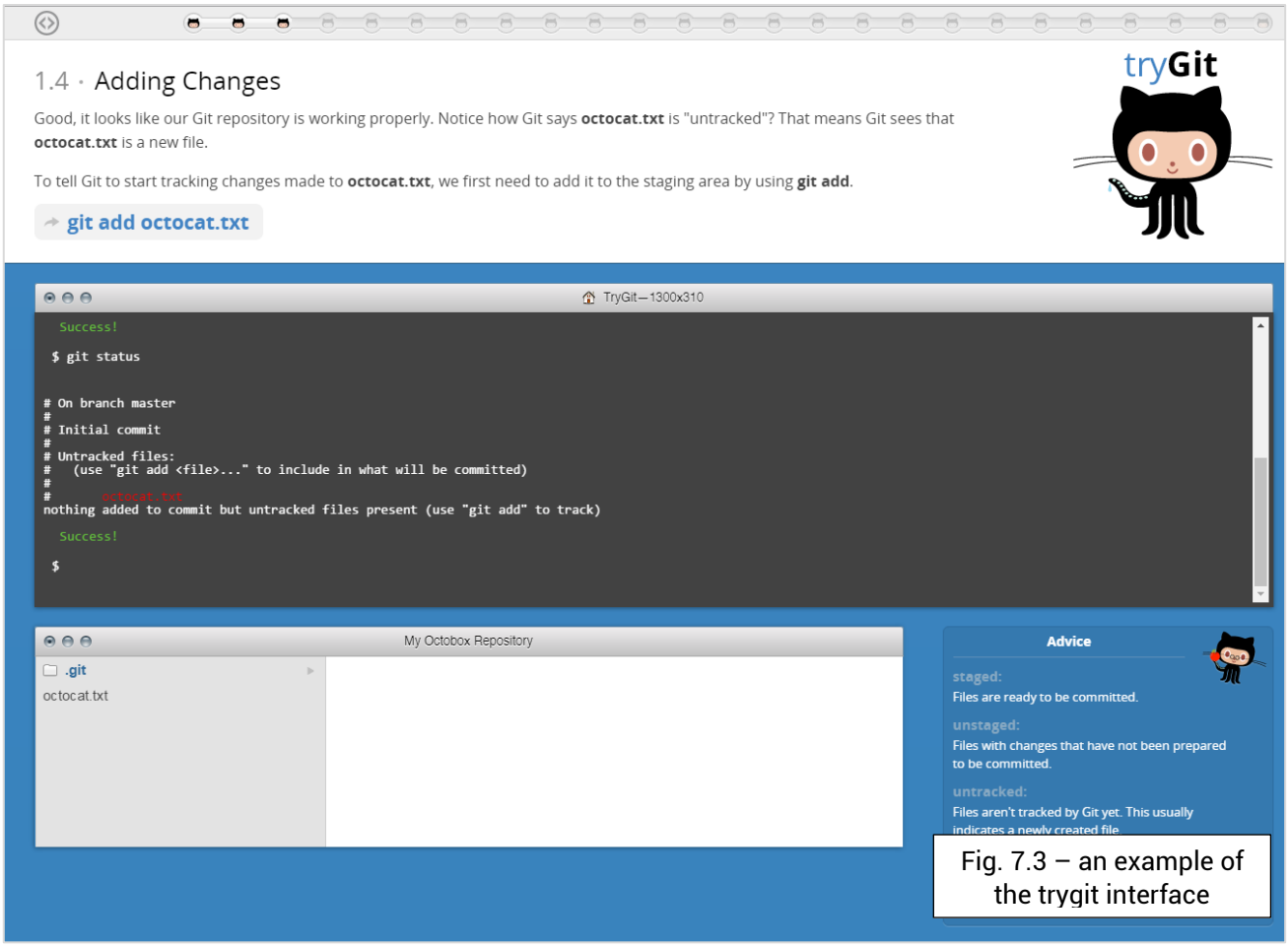


Fig. 7.3 – an example of the trygit interface

Github also has a feature to open issues related to the functioning of the code. This lets the developer know that something is wrong in their code and also allows people to provide troubleshooting advice to the person experiencing the issue.

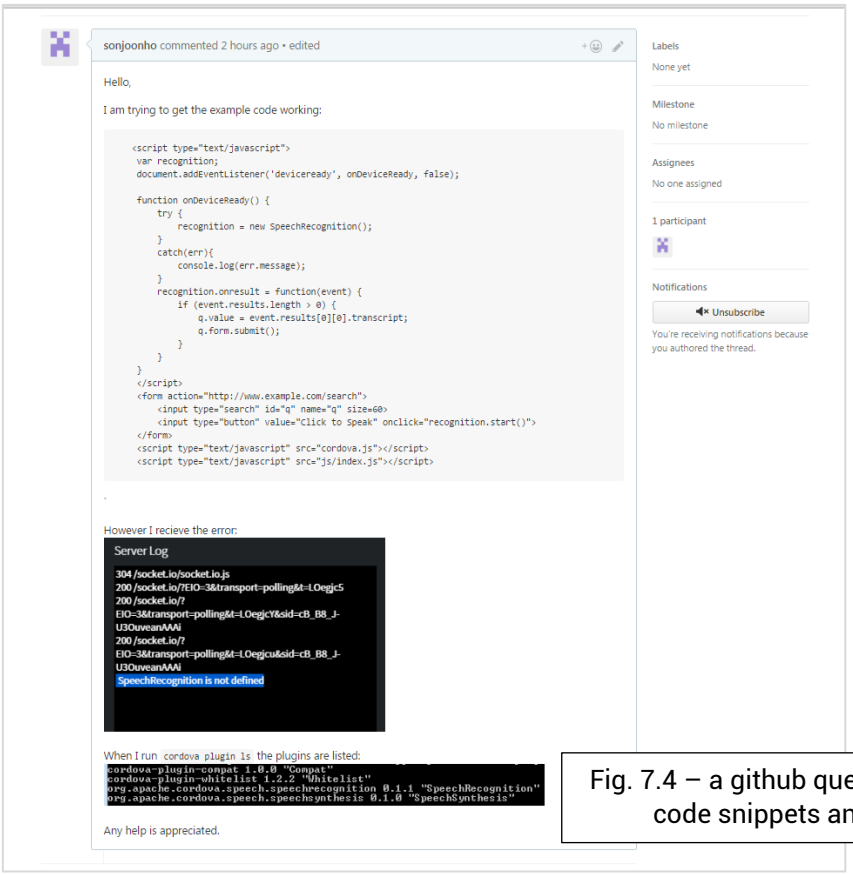


Fig. 7.4 – a github query I made, including code snippets and screenshots

Apache Cordova

Apache Cordova is a mobile application development framework that wraps a HTML and JavaScript app into an offline native container. This allows developers to access hardware functions of the device which would be otherwise unavailable to non-native developers.

These native functions are enabled by the use of plugins.

List of Cordova plugins used in this project

- Geolocation [11]
- Speech Recognition [12]
- Speech Synthesis [13]

Libraries and Application Programming Interfaces

A library is a collection of subroutines that can be called from your own program. In addition to reducing time cost, the functions contained in a library are often fully optimised meaning that they will probably run faster than if you wrote it yourself. For example, the JQuery library is a set of prewritten JavaScript blocks of code that can be used and reused while assembling your website.

APIs provide a means to interact with an application or library hosted on a different server. They often return raw data that can be manipulated by the developer in whatever way is necessary. Making a request to the Google Maps Distance Matrix API returns data such as travel time and distance.

List of APIs used in this project:

- simpleWeather API [14]
- Google Distance Matrix API [15]
- Jquery Library [16]

Though there are various GUIs available to aid with Cordova development (e.g. Phonegap Desktop App), my preferred method of interaction with Cordova is the Command Line Interface (CLI) due to the greater control and flexibility it provides the user. Various commands are entered into the CLI to perform certain operations. For example, this is what creating a new project looks like. My inputs are marked by a $ preceding the line.

```
Joonho@JOONHO C:\Users\Joonho
$ cd C:\Users\Joonho\Google Drive\College\EP\Apps

Joonho@JOONHO C:\Users\Joonho\Google Drive\College\EP\Apps
$ cordova create SmartMirrorApp com.sonjoonho.smartmirrorapp SmartMirrorApp
Creating a new cordova project.

Joonho@JOONHO C:\Users\Joonho\Google Drive\College\EP\Apps
$ cd SmartMirrorApp

Joonho@JOONHO C:\Users\Joonho\Google Drive\College\EP\Apps\SmartMirrorApp
$ cordova platform add android
Adding android project...
Creating Cordova project for the Android platform:
        Path: platforms\android
        Package: com.sonjoonho.smartmirrorapp
        Name: SmartMirrorApp
        Activity: MainActivity
        Android target: android-23
Android project created with cordova-android@5.2.1
Discovered plugin "cordova-plugin-whitelist" in config.xml. Adding it to the project
Fetching plugin "cordova-plugin-whitelist@1" via npm
Installing "cordova-plugin-whitelist" for android

               This plugin is only applicable for versions of cordova-android greater than 4.0.
telist will be built in.


Joonho@JOONHO C:\Users\Joonho\Google Drive\College\EP\Apps\SmartMirrorApp
$
```

Fig. 7.4 – An example of the CLI

This creates a framework upon which the developer will create their app.

The next series of commands involve initialising a new local (only on your hard drive) git repository and then uploading it github to create a remote (online) repository.

```
git init

git add *

git commit –m “Initial commit”

git remote add origin
https://github.com/sonjoonho/SmartMirrorApp.git

git push –u origin master
```

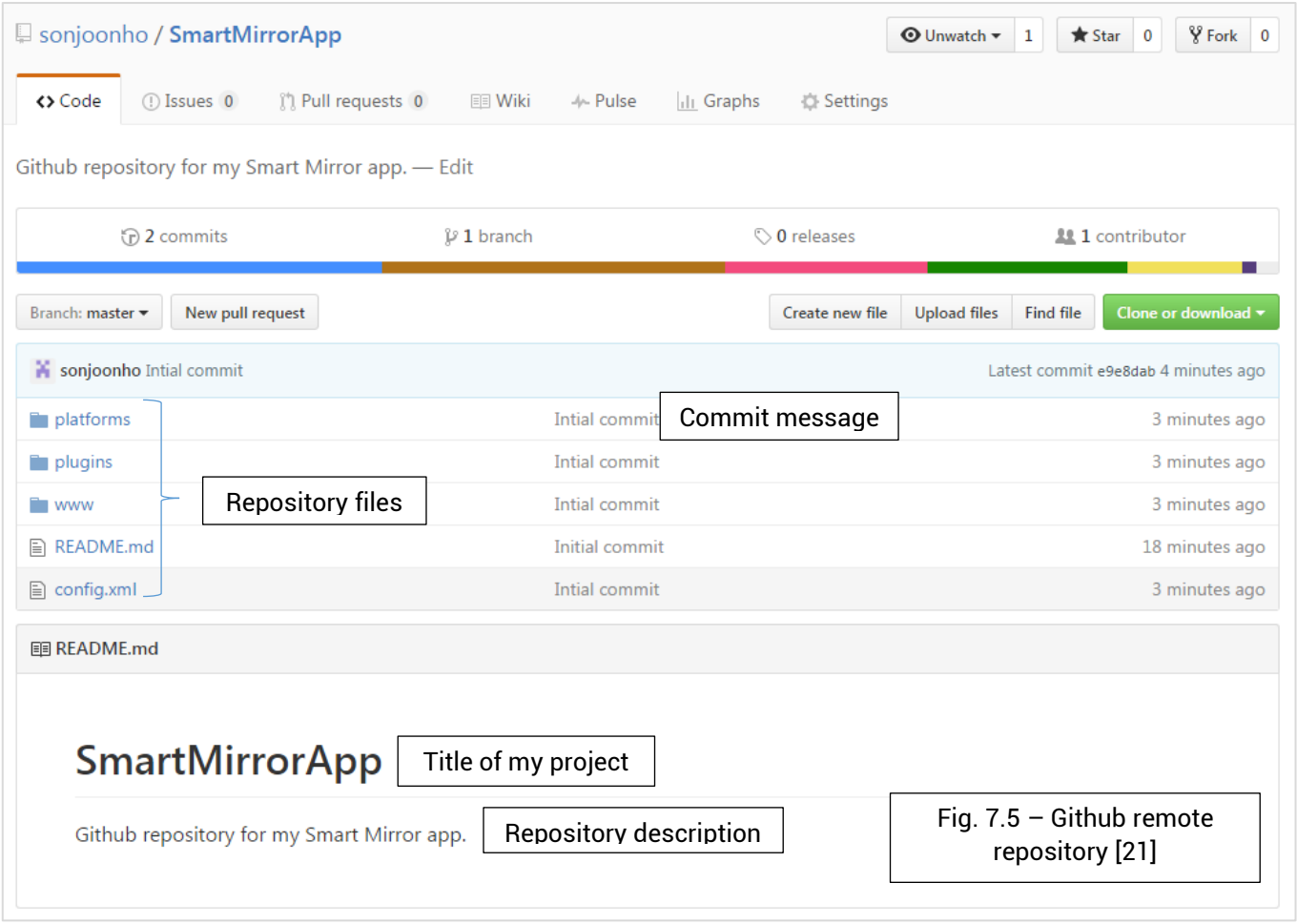


Fig. 7.5 – Github remote repository [21]

The Minimum Viable Product (MVP) is used by companies to gauge customer interaction about a product and to receive feedback. It includes the minimum set of features necessary to deploy a product and is used as a strategy to avoid wasting time on unwanted features [17]. Here is the basic MVP prototype I created compared to the final product. Note that the final has inverted colours (black background, white text). This is necessary to ensure the text shows up properly on the surface of the mirror.

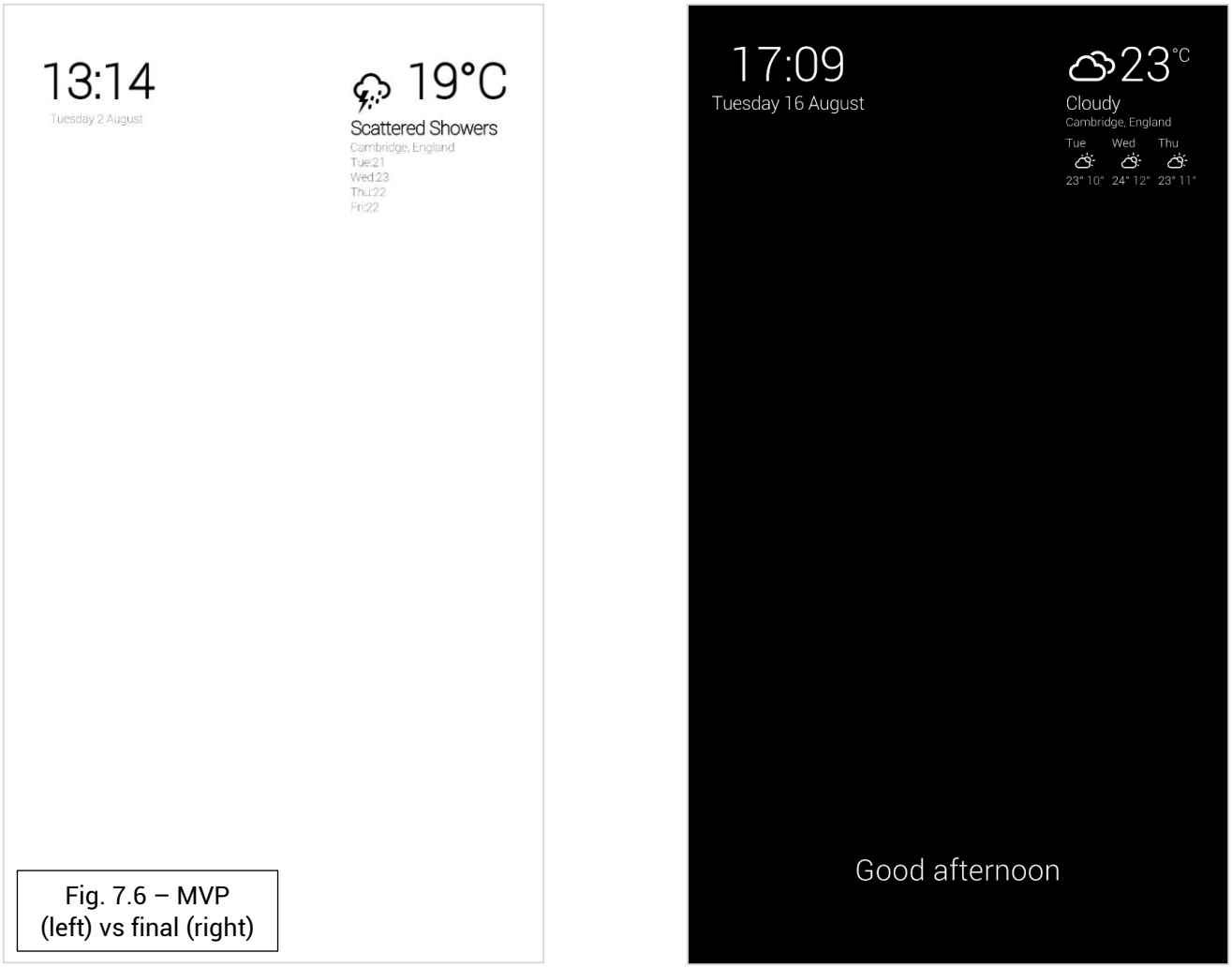


Fig. 7.6 – MVP (left) vs final (right)

The Console

Programming is all about problem solving – and the console is one of the most useful tool for a developer in diagnosing errors. JavaScript allows the developer to log messages to the console from the code. This means that you can output messages to tell you what the code is doing, what is working, and what is throwing errors. Otherwise, it is likely that a function could be not working and you will have no idea why. It will also tell you what *type* of error it is. As an example, a **syntax error** includes things like spelling mistakes and incorrect use of capital letters. This gives an indication of what to look for. Concretely, the console aids the developer in tackling bugs methodically, by isolating the source of the problem.

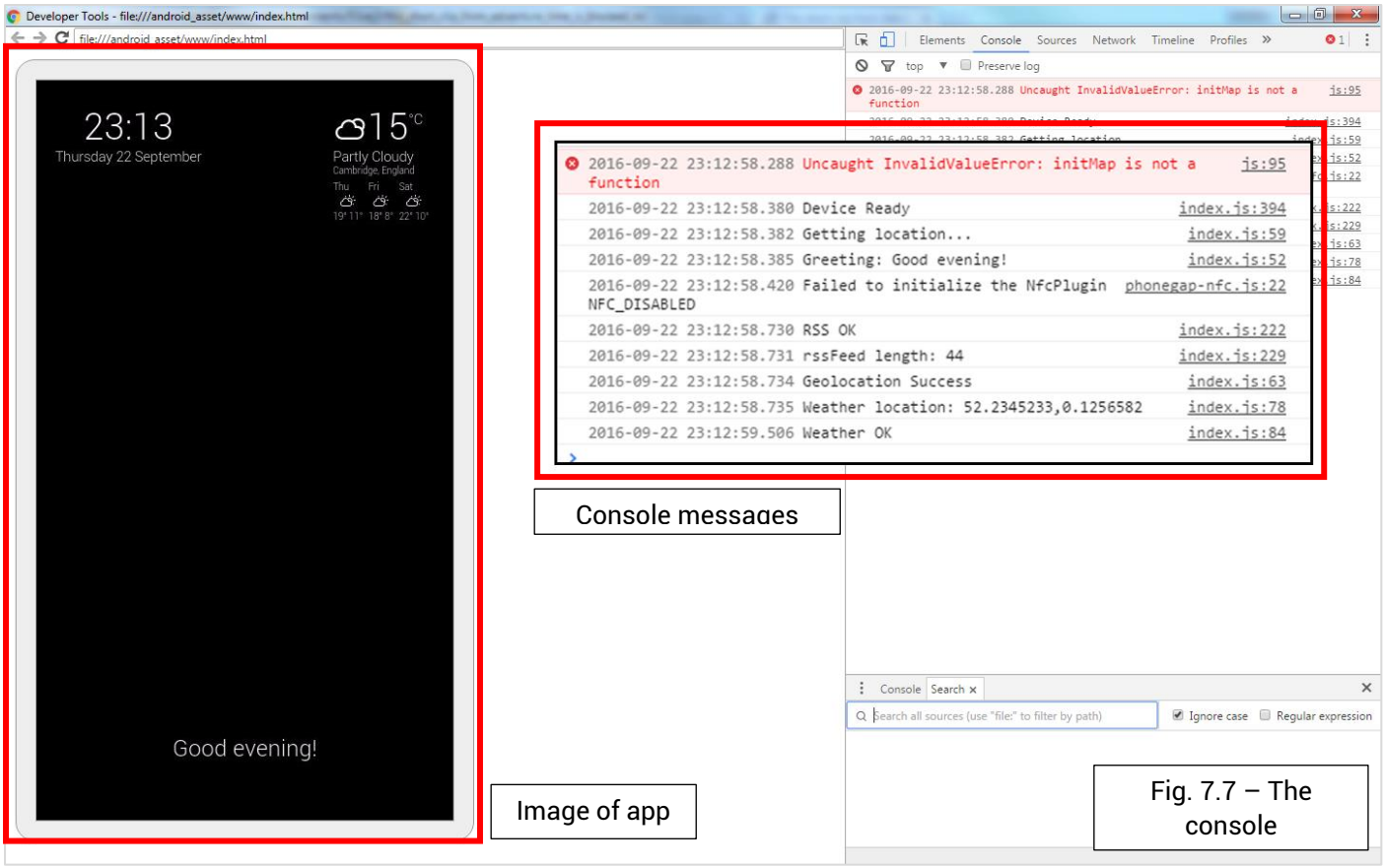


Fig. 7.7 – The console

Chromecast Integration

You may notice that in the development pictures the interface is oriented with elements positioned vertically, however in any photos of the actual display the interface is oriented horizontally.

Programmers often use comments and syntax highlighting to improve the readability of their code. Comments are annotations included in the code that have no effect on the execution on the code but provide useful documentation. They are marked in the code by preceding special symbols, which are language specific. In JavaScript, comments are marked by two slashes.

Syntax highlighting means colouring text depending on what kind of purpose it serves in the code. For example: functions may be highlighted green, statements in blue, and integers in purple. The style of highlighting depends on the code editor.

Also note the indentation of lines inside the curly brackets to show the structure of the code.

```
//this function multiplies two numbers together
function multiply(a, b) {
      var a = 5;
      var b = 6;
      console.log(a * b);
}
```

Fig. 7.8 – Example code

During this commentary I have not detailed the entire development process as it would go *far* beyond the confines of the word limit (which I have already exceeded by quite a bit) and it is almost impossible to document the thought process when writing code in a comprehendible manner. Instead I have tried to outline the key points which, I hope, will give greater insight into the world of software development. What follows is the complete, commented code with syntax highlighting. I have also created a flowchart to illustrate how a function works (see the Marketplace Review). The full code is attached in the appendicies.

NFC

Near-field Communication (NFC) is a set of communication protocols that enable two electronic devices, one of which is usually a portable device such as a smartphone, to establish communication by bringing them within about 4 cm (2 in) of each other. [18]

Since my mirror is powered by the phone display it would be convenient for the app to open by itself, instead of the user having to open it manually. NFC tags can be programmed to perform this action. My idea was to combine an NFC tag with a wireless charger to create a *"dock"* where the user can place their phone and the mirror interface will start automatically and will also charge wirelessly simultaneously. I was able to get the NFC demo running at my marketplace, but I did not have access to a wireless charger as they are rather expensive and as a results out of my budget range.

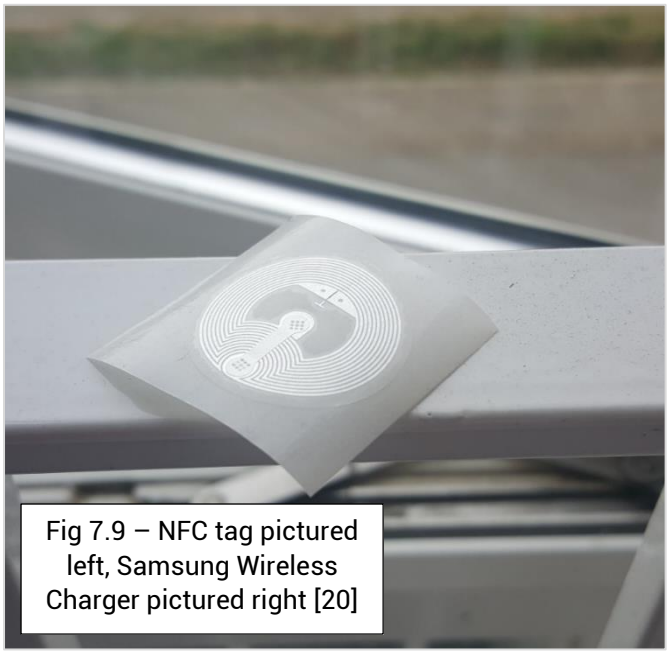


Fig 7.9 – NFC tag pictured left, Samsung Wireless Charger pictured right [20]

Complete List of Features

- Displays current time and date
- Displays current weather and forecast
- Weather in other locations
- Greeting
- Speech recognition
- News
- Translation
- Geolocation
- Time to work
- Distance between…
- Music player

“What's the weather in [location] ?"

“The weather in [location] is [weather[."

“How do I say [phrase] in [language] ?"

“[translated phrase]." [translated phrase is displayed]

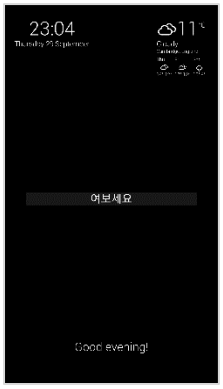


Fig 7.10

“Show me the news."

“Here are today's items." [BBC News feed is displayed]

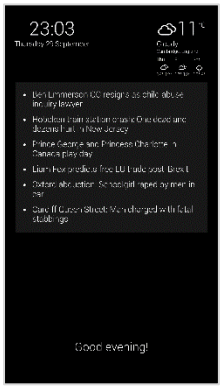


Fig 7.11

“Close."

[News feed window closes]

“Time from [origin] to [destination]."

“[time] minutes."

“Distance from [origin] to [destination]."

“[distance] kilometres."

“Play music."

[music plays]

Visual Timeline of Development

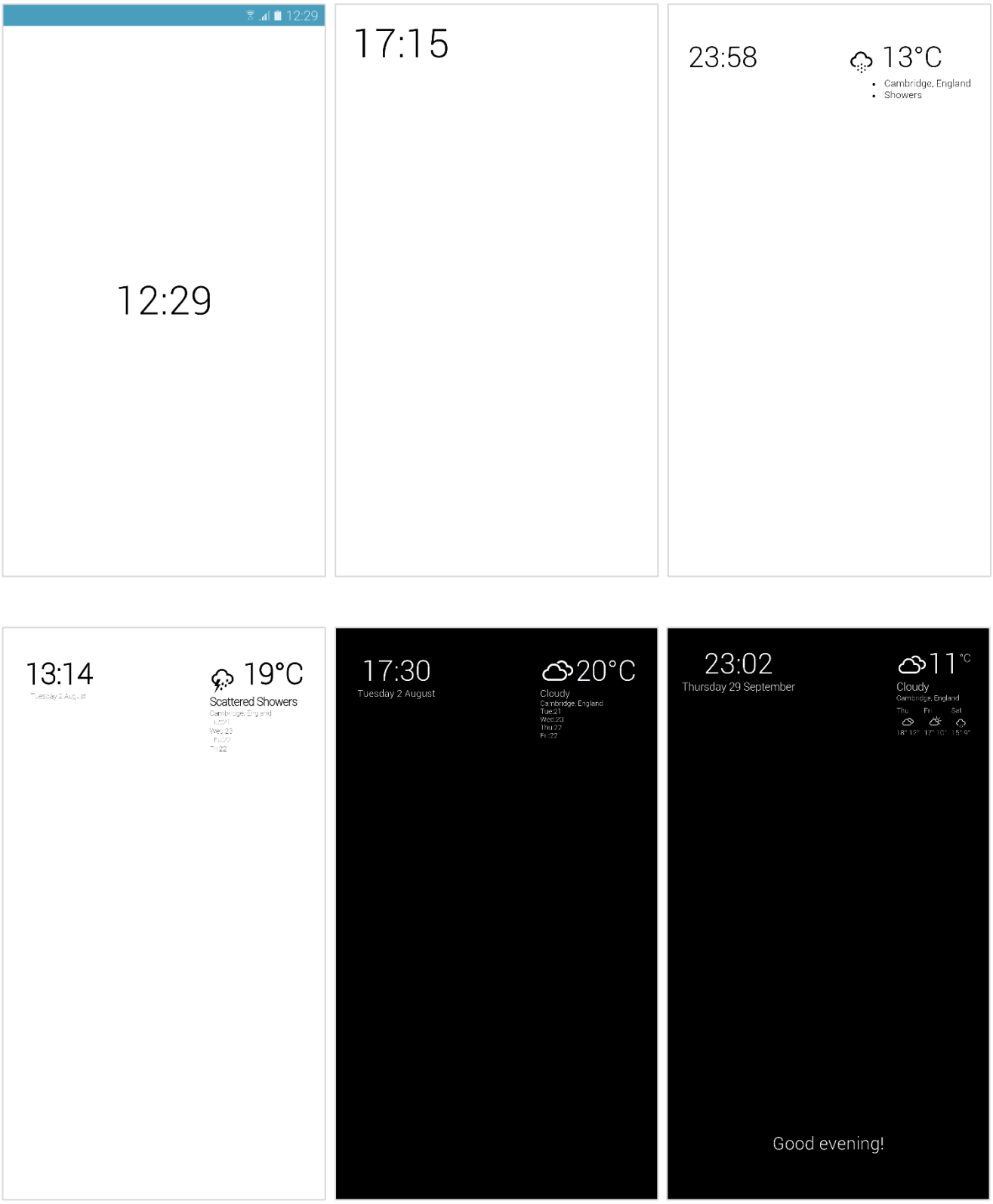


Fig 7.12

Construction

Very fortunately, I was able to acquire a TV for a mere £20 from a friend of mine. The TV model is a **Technika 19-248**. These were the results of the teardown.

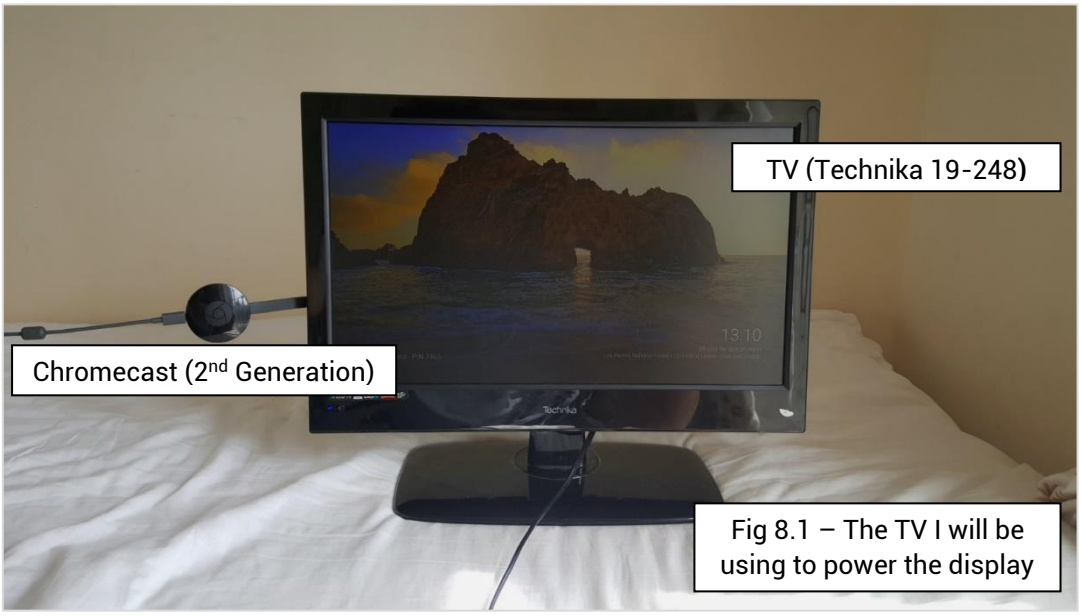


Fig 8.1 – The TV I will be using to power the display

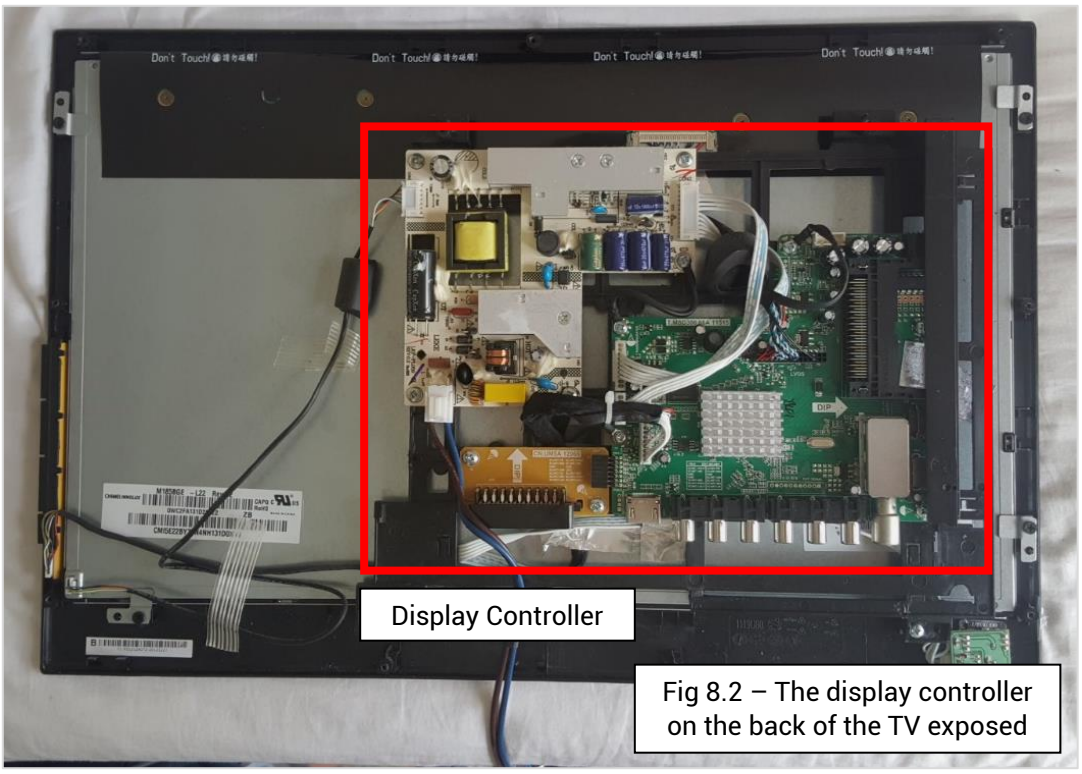


Fig 8.2 – The display controller on the back of the TV exposed

The next course of action is to acquire a one-way mirror (also sometimes confusingly called a two-way mirror) that will create the desired futuristic mirror-display effect. For this I investigated three methods of achieving this: glass, acrylic and film.

I concluded that acrylic one-way mirrors provide an optimal compromise between cost and quality.

Upon acquiring the mirror I made the unfortunate discovery that it was a few millimetres too big for my LCD.

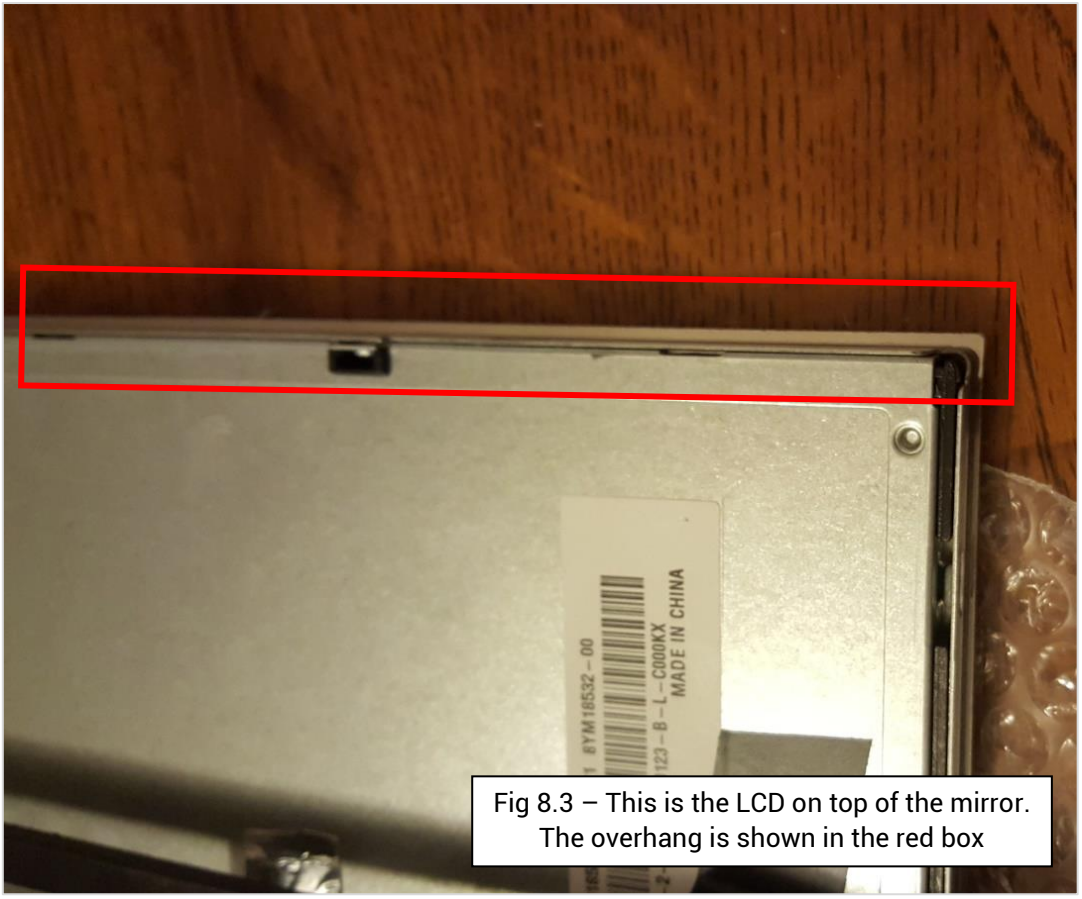


Fig 8.3 – This is the LCD on top of the mirror. The overhang is shown in the red box

To fit a frame around the device I needed a precise fit, without any overhang. I considered my options:

- Attempt to cut the few millimetres very precisely
- Try and sand the edges down

It did not take much thought before I decided that neither of these ideas would be very suitable. It would be an impossible task to cut with such precision and without causing damage to the mirror with the tools I have on hand. Sanding was likely to create an uneven edge, somewhat ruining the clean and minimalistic aesthetic of the mirror. Instead, I went into college to meet with an electronics teacher (and enthusiastic home-engineer) who had extensive experience in his own personal projects. He suggested using thin wooden splints to fill out the overhang. This would be many times safer than cutting the mirror and required minimal cost.

I bought some thin pieces of basswood and cut them to size with a box cutter.

Fig 8.4 – Wooden strips of basswood

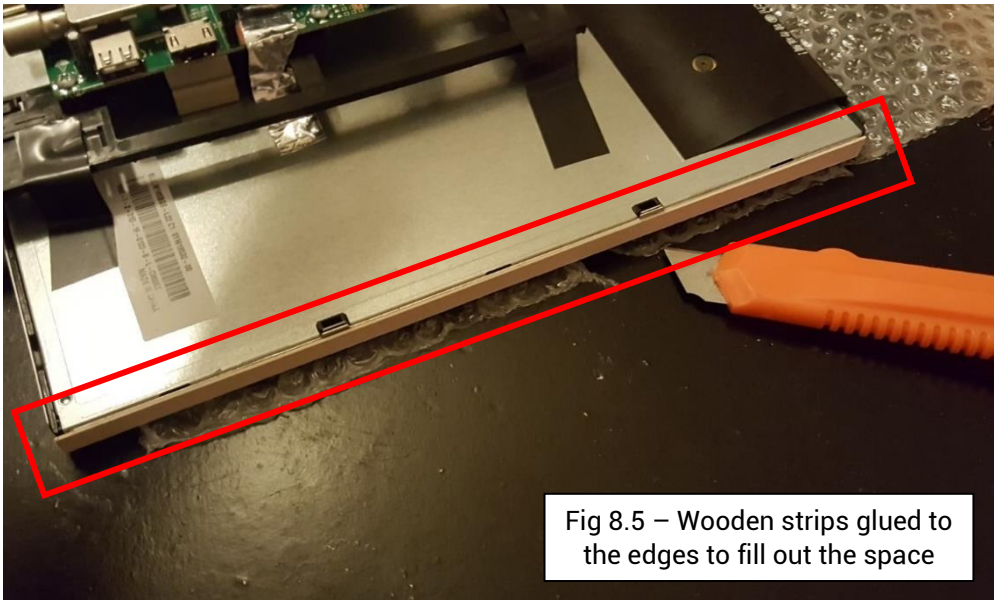


Fig 8.5 – Wooden strips glued to the edges to fill out the space

Before sticking the mirror to the bezel I had to remove some excess plastic on the display controller board that was bigger than the LCD, meaning it could not stand up properly.

After solving this problem, I was next faced with the task of sticking the mirror to the LCD itself. I had previously decided to glue the mirror to the bezel (metal part surrounding the screen) with a type of superglue. However, I decided to get a second opinion from a friend of mine who is an engineer. He agreed on my decision to use glue, and recommended a brand called Loctite.

Before sticking the two together, it was imperative to clean and polish the surface of the mirror as much as possible, as any imperfections will be permanent. I wore gloves to prevent fingerprints and used a microfiber cloth clean it.

At this point I faced a major issue when I discovered some damage done to the backlight circuit on the LCD (the one that literally says "*Don't touch*" on it). I think this was caused by the wooden strip that I glued on top of it, even after taking care not to glue directly onto the circuit. I felt that adding a solid wooden frame on top of that would not help matters so I ultimately decided to abandon the idea completely. Although this is an unfortunate departure from my original plan, I believe it is a necessary one, or I risk the entire project.

Fig 8.6 – Loctite super glue

Fig 8.7 – Cleaning the surface of the mirror

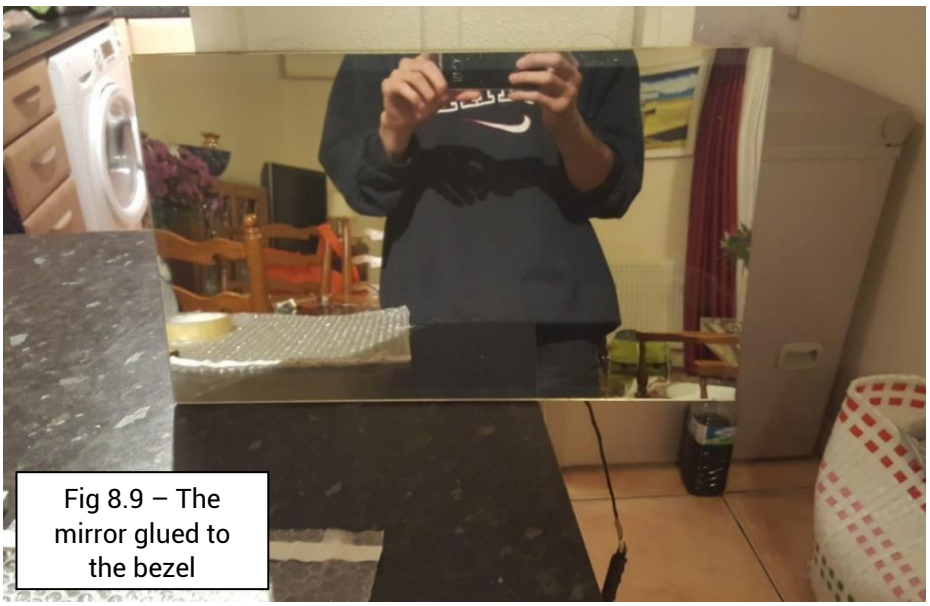


Fig 8.9 – The mirror glued to the bezel

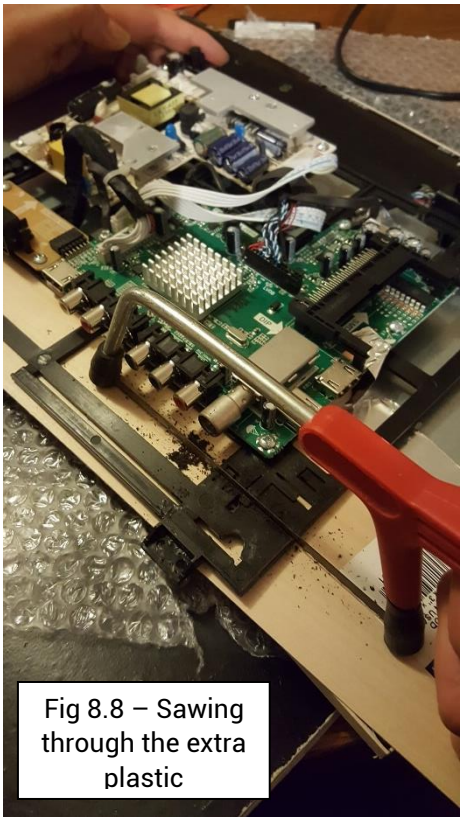


Fig 8.8 – Sawing through the extra plastic

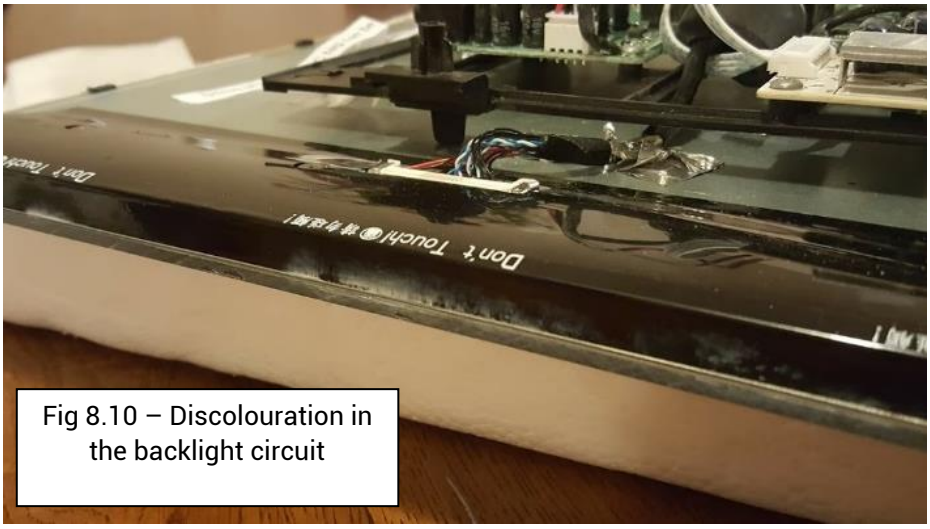


Fig 8.10 – Discolouration in the backlight circuit

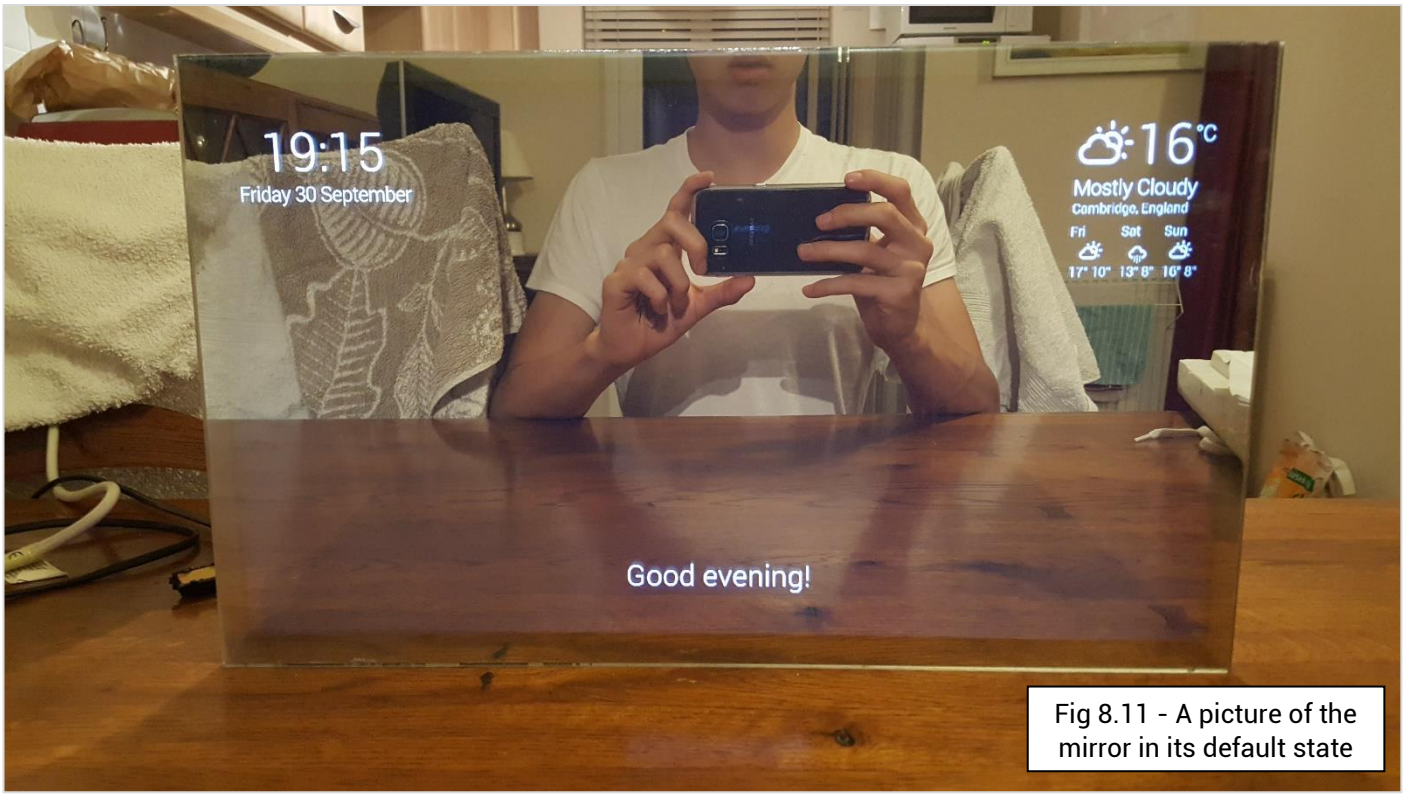


Fig 8.11 - A picture of the mirror in its default state

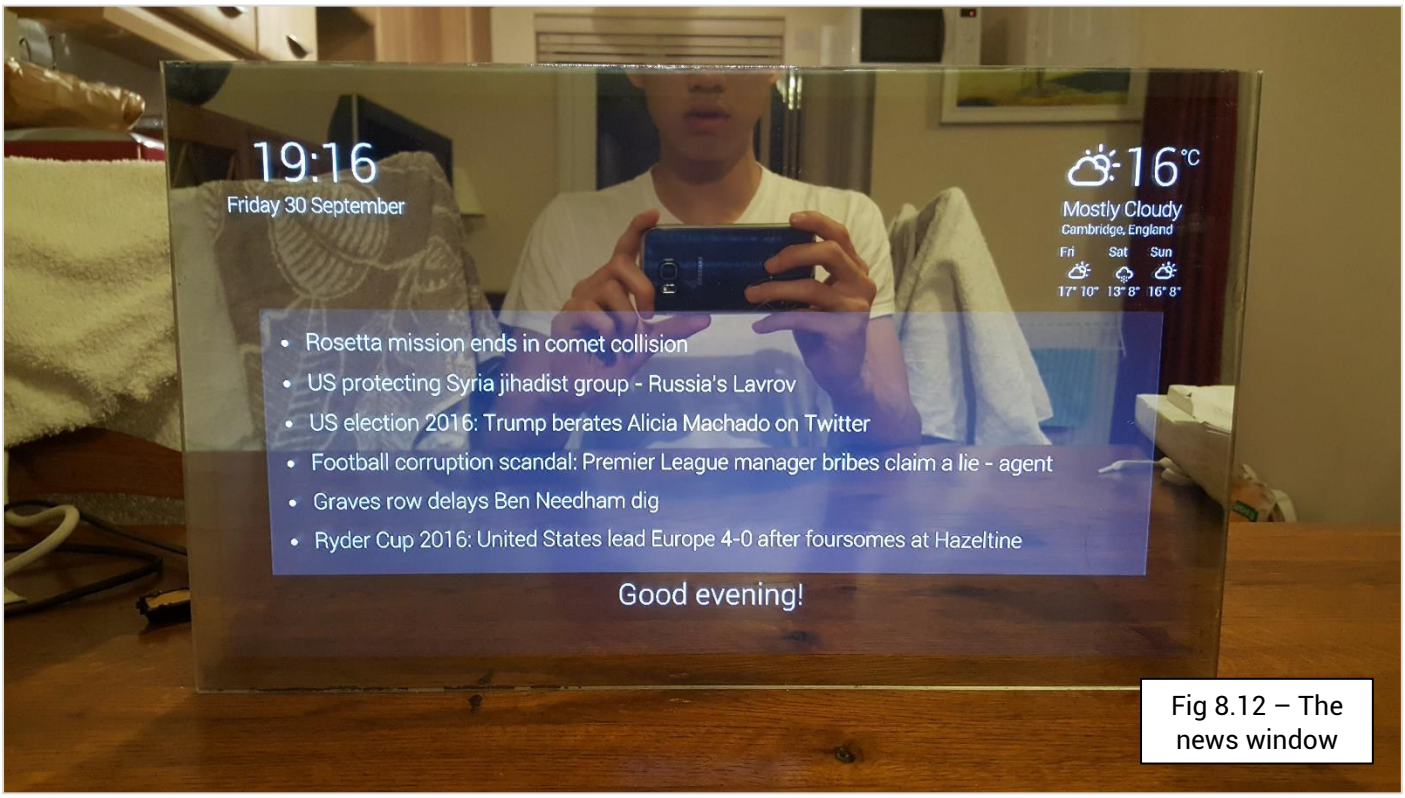


Fig 8.12 – The news window

Evaluation

Marketplace Review

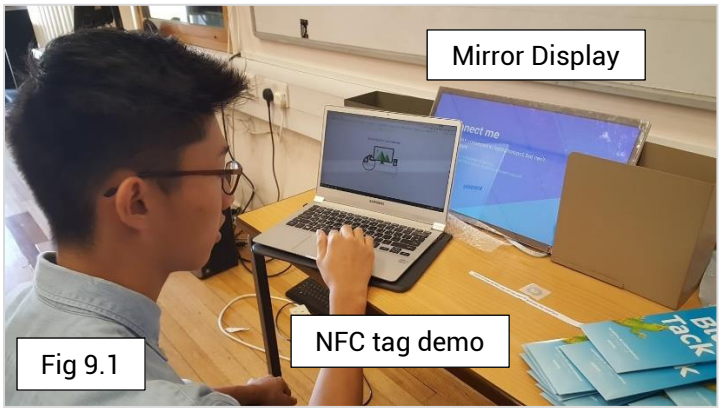


Fig 9.1

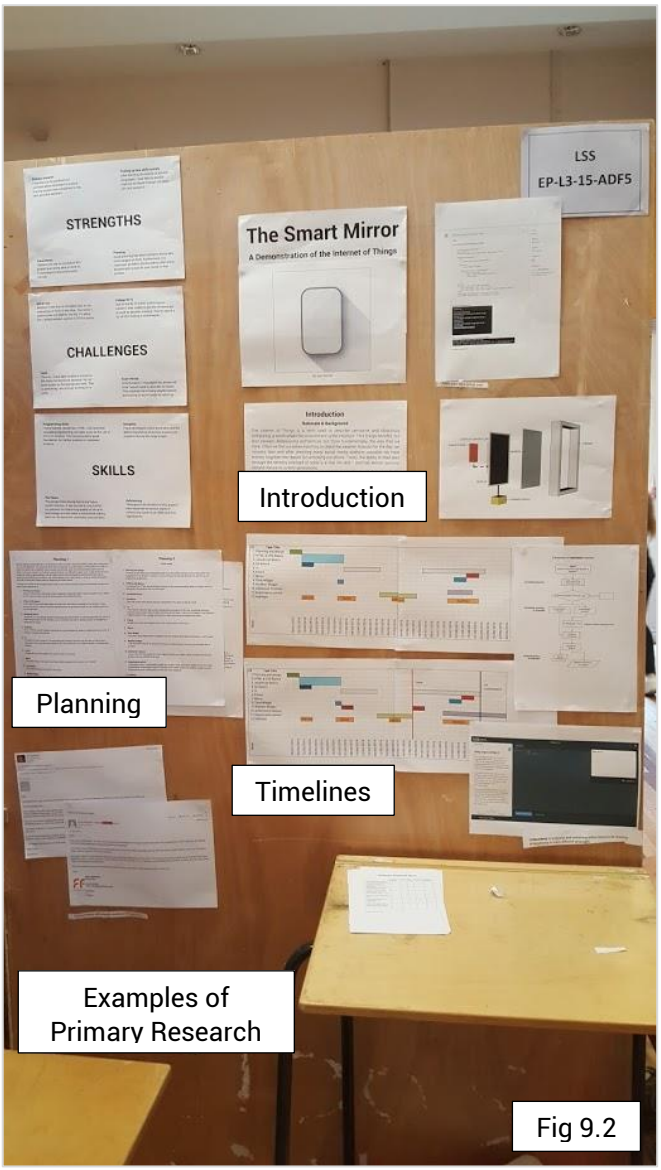


Fig 9.2

Overall, I would say my marketplace was a success. It was not without its challenges, however. I faced some major technical issues while I was setting up my mirror. The problem was due to the fact that my Chromecast could not connect properly to the school's Wi-Fi. I go into further detail about this in the *"Challenges"* section later on.

In spite of this, I eventually managed to get a basic demo of the interface working, albeit with many of the major features not available. Unfortunately, as a result of this I was not able to show off the speech recognition features which is perhaps one of the more desirable or more impressive aspects of the mirror.

Apart from the main mirror, I also had an NFC tag demo working which was a small victory in a day plagued by unexpected difficulties.

However, as eye catching as it was, I felt that the mirror was not necessarily the most important part of the project but instead the process I went through to get there and the lessons I had learned as a result. I prepared various materials to showcase these achievements. These materials were taken from my final write-up which was largely complete by the time of the marketplace, but with some text had to be cut as to reach a reasonable compromise between readability (font-size) and quantity. Despite this I tried to be selective with what material I included by attempting to cover as many of the marking points as possible as comprehensively as possible.

I feel that I was able to effectively deliver a clear presentation on every aspect of my project and answer any questions that visitors to my stall had. Moreover I got the impression that people were drawn to the stall because of the unique artefact I had on display which satisfying to see after many months of hard work.

Despite the technical nature of my project, my presentation demographic was a largely non-technical audience. Therefore I had to apply a degree of abstraction when putting together my presentation materials. A prime example of this was a flowchart I created of a sample function in my code. This flowchart is pictured on below, with the original code. Code is close to meaningless to non-programmers

– even with comments. The aim of the flowchart was to provide a visual and easy-to-digest medium to show what the code is doing.

```
259     //This function translates text given to it
260     function translate(to_translate, to_language) {
261
262         //Converting the language spoken by the user into Google Translate language codes
263         //Again, it only supports spanish and korean in its current state
264         switch(to_language) {
265             case 'spanish':
266                 to_language = 'es';
267                 break;
268             case 'korean':
269                 to_language = 'ko';
270                 break;
271             default:
272                 to_translate = "I don't know that language"
273                 to_language = 'en'
274         }
275
276         //This actually works by querying the google translate website and then taking the translation directly from the HTML
277         //This bypasses the need to mess around with more APIs
278         var link = 'https://translate.google.com/m?hl='+to_language+ '&sl=en'+'&q='+to_translate.replace(' ', '+');
279         console.log(link);
280         $.ajax ({
281             url: link,
282             success: function(page) {
283                 var translation = $(page).get(7);
284                 translation = translation.innerHTML;
285                 console.log(translation);
286                 $('#translate').html('<h2>'+translation+'</h2>');
287                 $('#translate').slideDown('slow');
288                 setTimeout(function() {
289                     $('#translate').slideUp('slow')}
290                     , 5000);
291                 speak(translation, to_language);
292             }
293         })
294     }
295
```

Fig 9.3 – Code snippet depicting *translate* function

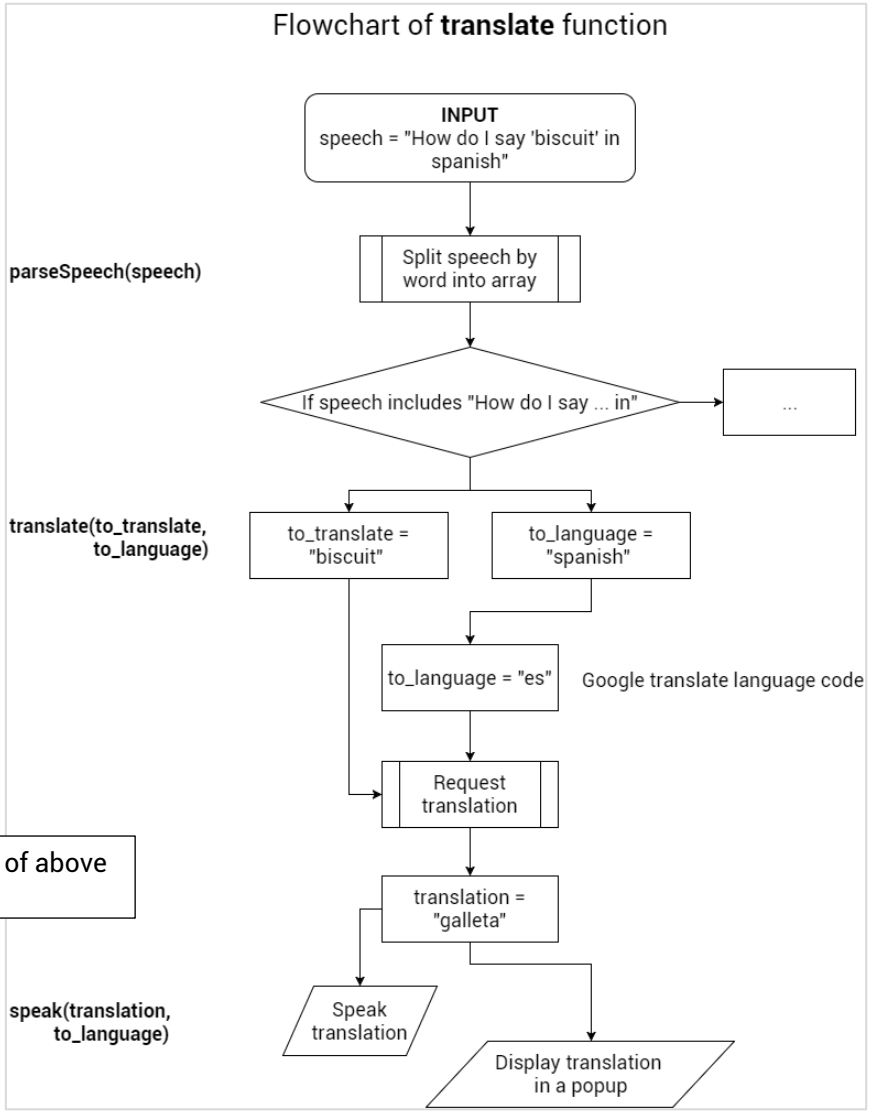


Fig 9.4 – Flowchart of above code

Audience Feedback Sheet

	Strongly agree	Agree	Not sure	Disagree	Strongly disagree
The display was clearly presented and informative.	𝍸 III	II			
I seemed clear and engaged in what I was presenting.	𝍸 III	II			
I have shown my project management process.	𝍸 III	II			
I have clearly presented the strengths and weaknesses of my project.	𝍸	𝍸			
It is clear why I chose this topic.	𝍸 I	III	I		
I have explained what I would have done differently.	𝍸	IIII	I		
I have shown to have learnt new skills through working on this project.	𝍸 IIII	I			

Comments: (optional)

"The smart mirror was really cool!"

"The mirror looked fancy and the wiring was lit [colloq. cool]"

Fig 9.5 – Audience feedback sheet

In order to analyse feedback received at my marketplace I created an Audience Feedback Sheet to compile responses. The figure above shows the total tally of all the responses I received.

Overall, the response was overwhelmingly positive and I am pleased with the result. I received a majority (~97%) *Strongly agree* and *Agree* in all areas of my presentation. My weakest section was "*I have explained what I would have done differently*". While I clearly showed challenges I faced in my project, perhaps I could have more clearly demonstrated how these challenges would influence future projects.

Strengths

In my opinion, one of my greatest strengths during this project was the ability to pick up new skills quickly. Before beginning the project I had little to no experience in any of the programming languages or techniques used. This enabled me to get started on the actual development quickly, so I could hit deadlines and targets punctually.

In order to meet these targets I was required to work consistently. I would not have been able to accomplish what I did by cramming it just before the deadline. The task simply demanded a level of complexity and organisation that could only be achieved by constant and steady progress. Of course, this excluded the exam period where revision took precedence.

Consistency is one thing, but having set targets to work towards is as, if not more, important. Good planning has been critical, allowing me to track my progress and prioritise the most important tasks. Not only is setting deadlines in advance important, but also being able to reflect and alter your plans dynamically to best fit your needs in that context. It is inevitable that your current conditions and ideas will change so a degree of adaptability is needed.

The project would be vastly different if it were not the help and expertise of professionals I asked for assistance. I reached out for opinions when I needed them to ensure that I could complete whatever task I was working on within the allocated time and to the best possible standard.

Challenges

One of the few oversights in my initial planning was due to the exam period. Unfortunately I had misjudged the amount of time I would need to dedicate to revision. Exam preparation took a lot longer than I had anticipated, resulting me in being slightly behind schedule and having to work harder to catch up.

Whether it was due to mistakes in my measurements or inaccuracy in my ruler, the one-way mirror I ordered was cut to slightly erroneous dimensions – being several millimetres too big in each direction. However, I was able to find a suitable solution to this (as detailed in the construction section) and in the end it had no great detrimental effect on my project. This could have been avoided by measuring more precisely or contacting the manufacturer.

One major issue I faced was the inability to pair my mirror and phone in college. The school's MAC authorisation system means that only approved devices have access to the internet. I had to devote a large amount of time to finding a workaround – which I finally did. The solution was to use a phone to create a mobile Wi-Fi hotspot that my Chromecast could connect to. This was less than ideal as the mobile network connection was patchy in that area. As my entire project depended on the Chromecast functioning this issue was pretty much unavoidable.

Some way through the summer I decided to add native Chromecast integration to my app. I theorised that this could potentially enable capabilities such as the ability to have the screen casting run in the background so that the user could do other things on the phone while the interface was still running. However, none of the solutions I tried worked, and instead I just wasted a significant amount of time trying to get it to run properly. If I were to recreate the project, I would surely use a Raspberry Pi or similar device as opposed to a Chromecast.

Another unexpected challenge I faced was to do with my frame that I had planned to build around my mirror. After gluing the basswood strips to the edges of the LCD I noticed some damage to the backlight circuit. While there was no apparent effect on the actual functioning of the device, I feared that adding a solid wooden frame on top of it could destroy my entire project. As a result I decided to abandon this idea, despite scheduling it in my timeline prior.

Skills

Foremost, I have learned JavaScript, HTML, CSS and other critical programming concepts such as the use of APIs and Git from scratch. This has provided excellent programming practice, providing good preparation for a Computer Science degree where I will have a head start with my programming skills. Not only this but this project in particular has allowed me to pursue my passion of improving quality of life with technology and forced me to communicate my ideas with clarity. It is a prime example of the type of thing I hope that a Computer Science will lead to. Furthermore, it has provided a significant talking point on my personal statement which I believe will give my application a competitive edge against others.

As a result of being forced to keep a strict diary and extensive planning, I have developed a solid work-ethic and learned the value of tracking your progress and producing detailed documentation as a result of this task. I feel like the final artefact would not even resemble its final form without the prior timescale materials I produced before I even began programming. Being able to set and hit deadlines is a vital skill that will be applicable to projects of any kind in the future, and even in general work and study.

I comprehensively researched different development methodologies and selected the one that resonated with me the most. I then applied this to my own project in a way that I think was very successful. In this way, there has been an element of self-discovery by which I have found how I can optimally approach a project.

Future Plans

I fully intend on continuing development on my mirror, and even creating further iterations of software and hardware. It would be nice to finish off some features I didn't get the chance to fully implement and also polish some that are already present.

An example of another feature I would like to add would be facial recognition. This would allow the mirror to detect the person using the mirror and display personalised information depending on the preferences set by that user.

If I were to make an effort to transform this software into a commercial product I would put some thought into how I would design an interface so that the display is customisable. For example, I could implement options for what widgets and features the users would want and perhaps even a drag-and-drop system to position elements on the display. Commercialising the hardware would be an entirely different issue, and much more complex. Certainly I would have to fabricate custom display controllers which, I presume, is a lot of work.

References

[1] O. Hughes, "How to build a smart mirror: Microsoft reveals secrets of DIY Magic Mirror project," International Business Times, 2 June 2016. [Online]. Available: http://www.ibtimes.co.uk/how-build-smart-mirror-microsoft-reveals-secrets-diy-magic-mirror-project-1563300.

[2] D. Cortez, "UpTop," 27 August 2015. [Online]. Available: http://www.uptopcorp.com/post/mobile-app-development-native-vs-hybrid-vs-mobile-websites .

[3] "Codecademy," [Online]. Available: https://www.codecademy.com/.

[4] J. Atwood and J. Spolsky, "Stack Overflow," [Online]. Available: http://stackoverflow.com/.

[5] Apple, "Apple Human Interface Guidelines," [Online]. Available: https://developer.apple.com/ios/human-interface-guidelines/ .

[6] Google, "Google Material Design Guidelines," [Online]. Available: https://material.google.com/ .

[7] C. Larman, Agile and Iterative Development: A Manager's Guide, Addison-Wesley Professional, 2003, p. 27.

[8] D. W. W. Royce, "Managing the Development of Large Software Systems," Monterey, 1970.

[9] Oxagile, "Waterfall Software Development Model," 5 February 2014. [Online]. Available: http://www.oxagile.com/company/blog/the-waterfall-model/.

[10] B. Boehm and R. Turner, Balancing Agility and Discipline: A Guide for the Perplexed, Boston: Addison-Wesley, 2004.

[11] Apache, "Cordova Plugin Geolocation," Apache, 31 March 2013. [Online]. Available: https://github.com/apache/cordova-plugin-geolocation.

[12] S. MacDonald, "Speech Recognition Plugin," 14 July 2013. [Online]. Available: https://github.com/macdonst/SpeechRecognitionPlugin.

[13] S. MacDonald, "Speech Synthesis Plugin," 14 July 2013. [Online]. Available: https://github.com/macdonst/SpeechSynthesisPlugin.

[14] J. Fleeting, 29 January 2016. [Online]. Available: http://simpleweatherjs.com/.

[15] Google, "Distance Matrix API," [Online]. Available: https://developers.google.com/maps/documentation/distance-matrix/.

[16] jQuery, "jQuery," [Online]. Available: https://jquery.com/.

[17] E. Ries, "Minimum Viable Product: a guide," 3 August 2009. [Online]. Available: http://www.startuplessonslearned.com/2009/08/minimum-viable-product-guide.html.

[18] C. Faulkner, "What is NFC? Everything you need to know," Techradar, 17 November 2015. [Online]. Available: http://www.techradar.com/news/phone-and-communications/what-is-nfc-and-why-is-it-in-your-phone-948410.

[19] "cordova-plugin-statusbar," [Online]. Available: https://cordova.apache.org/docs/en/latest/reference/cordova-plugin-statusbar/.

[20] Samsung, "Wireless Charging Pad," [Online]. Available: http://images.samsung.com/is/image/samsung/uk_EP-PG920IBEGWW_000000001_Front_black?$DT-Gallery$.

[21] J.-H. Son, "Smart Mirror App," 31 July 2016. [Online]. Available: https://github.com/sonjoonho/smartmirrorapp.

Source Evaluation

J. Atwood and J. Spolsky, "Stack Overflow," [Online]. Available: http://stackoverflow.com/	While a resource that anyone on the internet can contribute to could be considered dubious, due to the user-moderated nature of the website, answers and solutions are generally reliable. Not only this, but because of the number of experienced developers on the site, often they can provide a solution that is actually better than your own. Discussions of poor quality are *"downvoted"* so they are not visible.
C. Larman, Agile and Iterative Development: A Manager's Guide, Addison-Wesley Professional, 2003, p. 27.	Craig Larman is a highly respected computer scientist and "A Manager's Guide" is regarded as one of the foundation texts upon which on Agile development is based.
D. W. W. Royce, "Managing the Development of Large Software Systems," Monterey, 1970.	Dr Winston Walker Royce was a pioneer in the field of software development. He is best known for this very paper where he details and early version of what would become known as the Waterfall model. As this is essentially where the Waterfall model was born I would consider this to be an authoritative text. However, one could claim that it is not the most up-to-date resource as it was published over 4 decades ago.
Oxagile, "Waterfall Software Development Model," 5 February 2014. [Online]. Available: http://www.oxagile.com/company/blog/the-waterfall-model/	Oxagile is a software development company. The source cited is a blog post describing the waterfall model. The source itself cites Dr Royce's paper (see above) so perhaps there is a degree of redundancy in using this source. However I think it provides value in evaluating the strengths and weaknesses of the method and also provides more recent information, building on the original paper. It is written by a software developer for software developers so I would expect it to be written in full technical details.
B. Boehm and R. Turner, Balancing Agility and Discipline: A Guide for the Perplexed, Boston: Addison-Wesley, 2004.	Barry W. Boehm is a distinguished software engineer and professor of computer science. This is just one of many texts he has contributed to the field. In the book he writes a comprehensive and reasoned analysis on the essence on Agile development. It is also a relatively recent resource, despite being published 12 years ago.
C. Faulkner, "What is NFC? Everything you need to know," Techradar, 17 November 2015. [Online]. Available: http://www.techradar.com/news/phone-and-communications/what-is-nfc-and-why-is-it-in-your-phone-948410	This is a well-written and comprehensive article on NFC and its uses. Techrader is well-known as a decent publication on all areas of technology related news. The article is rich in detail and does an excellent job of producing technical information for a mainstream audience. Being the most recently written source out of all my references means it contains the latest information. Besides, NFC is already a well-established technology.

Glossary

API (Application Programming Interface)	A set of subroutine definitions, protocols and tools for building software applications.
CLI (Command Line Interface)	A means of interacting with a computer by directly typing in commands.
CSS (Cascading Style Sheets)	A style sheet language used for describing the presentation of a document written in a mark-up language.
Git	A version control system used for software development (see development section for more on git).
Github	Website to remotely host git repositories. (see development section for more on git)
GUI (Graphical User Interface)	A type of interface where users interact via graphical icons and visual indicators as opposed to text.
HTML (HyperText Markup Language)	A mark-up language used to create the structure of web pages.
Hybrid Application	An app built like a website, but packaged into a mobile app (see design section).
JavaScript	A programming language used alongside HTML and CSS to form the core of web development.
MVP (Minimum Viable Product)	An early build of a product including the minimum set of features necessary to deploy the product.
NFC (Near Field Communication)	A technology that essentially enable a type of Wi-Fi communication over the distance of about 4cm.
One-way mirror (Two-way mirror)	A mirror that is partially reflective and partially transparent.
Open-source	Computer software with its source code made publicly available.
RSS (Rich Site Summary)	A web feed used to publish frequently updated information (e.g. news headlines, blog entries etc.).