**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,** 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. 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.

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**

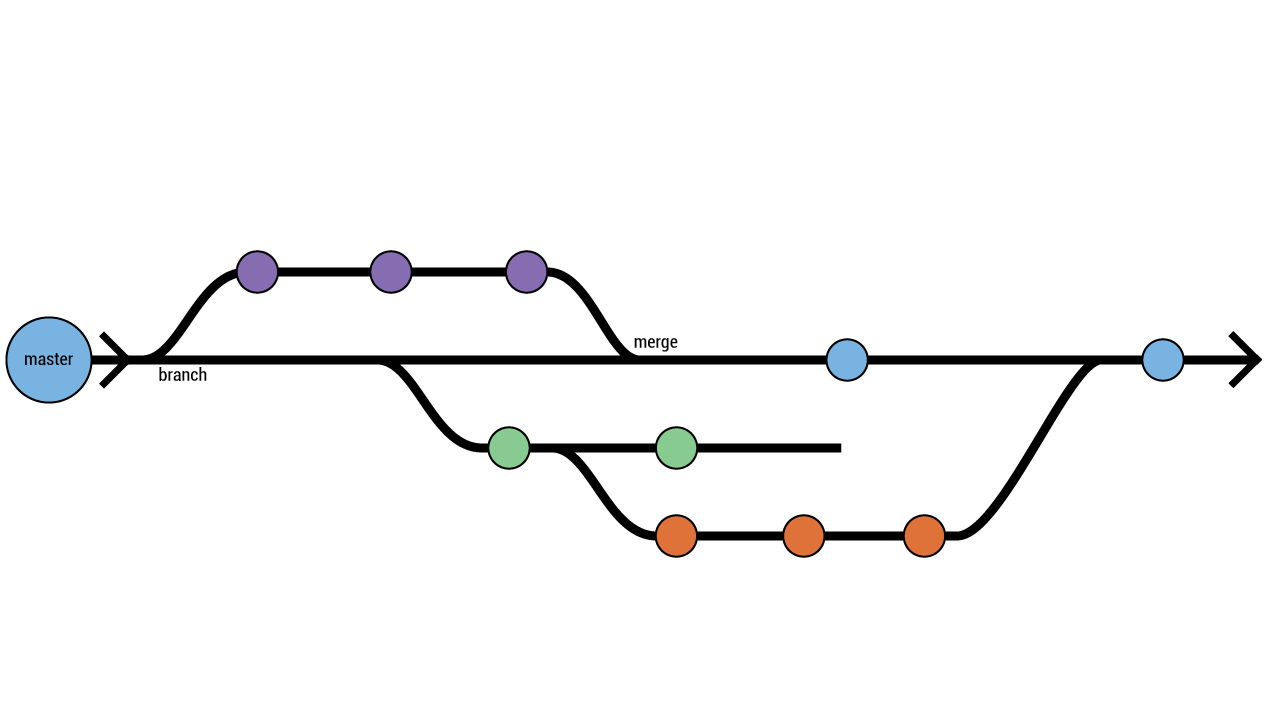
Git is a free, industry standard version control system used heavily in software development. It provides support for non-linear workflows by creating “forking” **branches** that can be worked on independently without affecting the **master** branch and are eventually **merged**. In this way, conflicts 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.

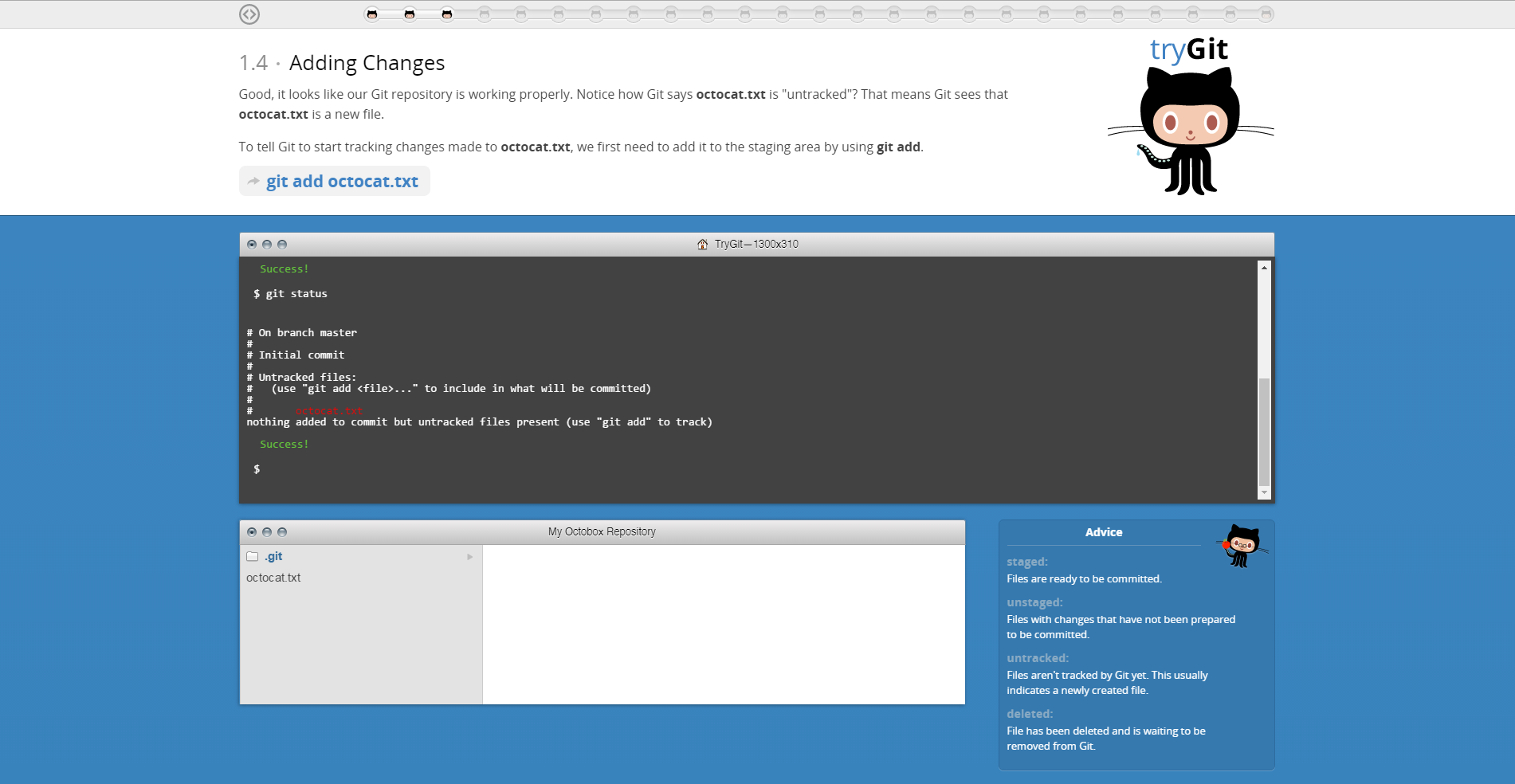


Fig. 7.3 – an example of the trygit interface

**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 <https://github.com/apache/cordova-plugin-geolocation>
* Speech Recognition <https://github.com/macdonst/SpeechRecognitionPlugin>
* Speech Synthesis <https://github.com/macdonst/SpeechSynthesisPlugin>

**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 <http://simpleweatherjs.com/>
* Google Distance Matrix API <https://developers.google.com/maps/documentation/distance-matrix/>
* Jquery Library <https://jquery.com/>

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.

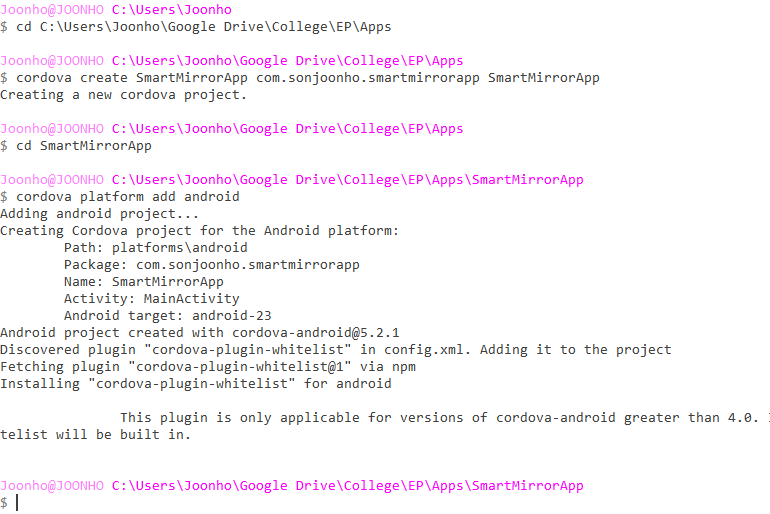
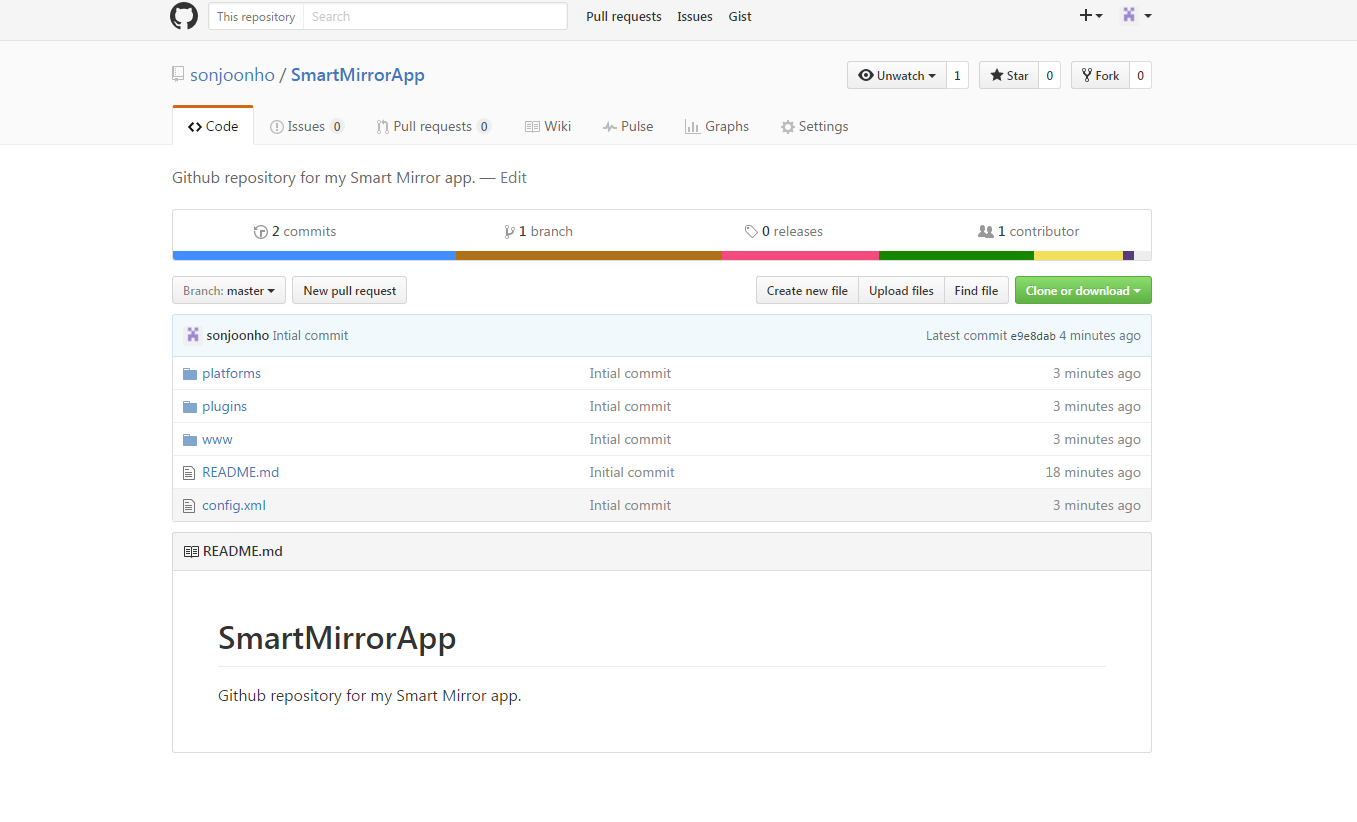
****

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.





Repository description

Title of my project

Fig. 7.5 – Github remote repository

Commit message

Repository files

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. Here is the basic MVP prototype I created compared to the final product.

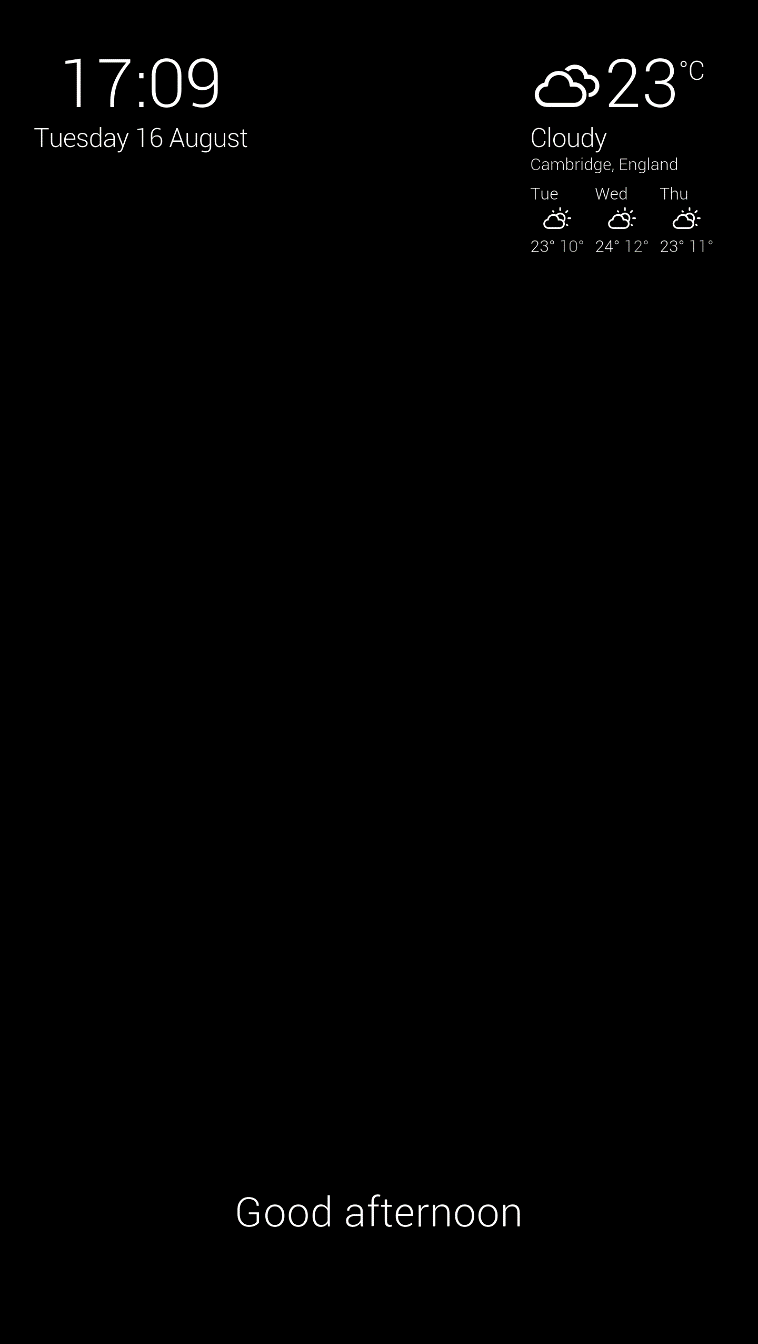


Fig. 7.6 – MVP (left) vs final (right)

**The Console**

The console is one of the most useful tool for a developer. It is essential 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.

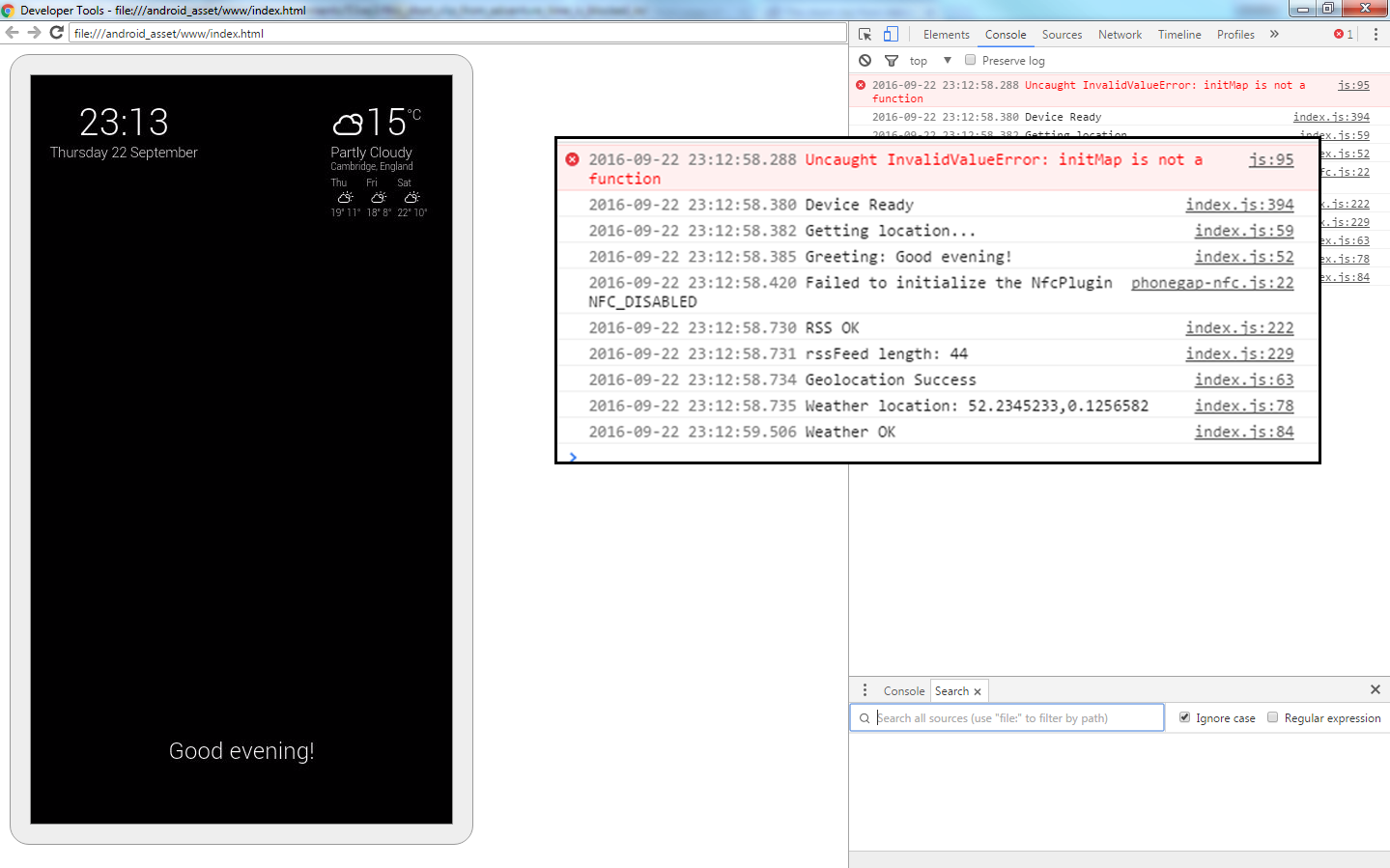


Image of app

Console messages

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.



During this commentary I have not detailed the entire development process as it would go *far* beyond the word limit (which I have already exceeded by quite a bit). 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; please see the Marketplace Evaluation.

[INSERT CODE]

**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. [1]

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 7.8 – NFC tag pictured left, Samsung Wireless Charger pictured right [8]

**Complete List of Features**

* Time & date
* Current weather and forecast
* Weather in other locations
* Greeting
* Speech recognition
* News
* Translation
* Geolocation
* Time to work
* Distance between…
* Music player

Bibliography

|  |  |
| --- | --- |
| [1] | 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. |
| [2] | Oxagile, “Waterfall Software Development Model,” 5 February 2014. [Online]. Available: http://www.oxagile.com/company/blog/the-waterfall-model/. |
| [3] | B. Boehm and R. Turner, Balancing Agility and Discipline: A Guide for the Perplexed, Boston: Addison-Wesley, 2004. |
| [4] | C. Larman, Agile and Iterative Development: A Manager's Guide, Addison-Wesley Professional, 2003, p. 27. |
| [5] | D. W. W. Royce, “Managing the Development of Large Software Systems,” Monterey, 1987. |
| [6] | “cordova-plugin-statusbar,” [Online]. Available: https://cordova.apache.org/docs/en/latest/reference/cordova-plugin-statusbar/. |
| [7] | J. Fleeting, 29 January 2016. [Online]. Available: http://simpleweatherjs.com/. |
| [8] | Samsung, “Wireless Charging Pad,” [Online]. Available: http://images.samsung.com/is/image/samsung/uk\_EP-PG920IBEGWW\_000000001\_Front\_black?$DT-Gallery$. |