

UNIVERSITY OF BRIGHTON

COMPUTER SCIENCE (GAMES) BSc(HONS)

INDIVIDUAL PROJECT - CI301

Final Year Project Report

Author:

Adam WORLEY

Supervisor:

Student Number:

Marcus WINTER

13842206

Hand in Date : 11th of May 2017



Contents

1 Methodology	8
1.1 Overview of types of methodology	8
1.1.1 Rapid Applications Development (RAD)	8
1.1.2 Agile	8
1.1.3 Lean	9
1.1.4 Waterfall	9
1.1.5 Spiral	9
1.1.6 Time Boxing	9
1.2 Choice of methodology	10
1.3 Project Time line	10
2 Stages of Development	12
2.1 Analysis	12
2.1.1 Competitors	12
2.1.2 Problem Analysis	15
2.2 Design	16
2.2.1 Design Considerations	16
2.2.2 Designs	17
2.3 Development	17

2.3.1	Platform	17
2.3.2	Target Market	17
2.4	Testing	18
2.4.1	Tests	18
2.5	Feedback	19
2.5.1	User Trials	19
3	Specification	19
3.1	Deliverables	20
3.1.1	Stages	21
3.1.2	Risk Analysis	21
4	Research	22
4.1	Philips Hue API	22
4.1.1	Influence	23
4.2	Design Considerations	23
4.2.1	Influence	24
4.3	Human Perception of Light	24
4.3.1	Influence	25
5	Assessment of Progress	25
5.1	Early Stages	25
5.2	Development Progress	26

5.2.1	Fragments	26
5.3	Testing and Feedback	27
5.3.1	Debugging	27
5.4	Problems Encountered	28
5.4.1	Alarm creation	28
5.4.2	Time Picker dialog	29
5.4.3	Handling Pending intents	30
5.4.4	Storing Objects	31
5.4.5	Alarm Solution	33
5.5	Lessons Learned	33
6	Accomplishments	34
6.1	The Ability to Control Lighting	34
6.2	The Application UI Design	35
6.2.1	Tab Navigation	35
6.2.2	Snackbars and Toasts	35
6.2.3	Using the Weather Icon Font	36
7	Success and Failure	37
7.1	Successes	37
7.1.1	The Project	38
7.1.2	Using SQLite	38

7.1.3	Pending intents	38
7.2	Failures	39
7.2.1	No Adjustable alarm	39
7.2.2	Not Reaching Stretch Goals	39
8	Original Project Plan	39
8.1	Aims and objectives	39
8.2	Stakeholders	40
8.3	Communications	40
8.4	Installation Process	41
8.5	Quality checks	41
8.6	How will I measure success?	41
8.7	Challenges	42
9	Divergence From Original Plan	43
9.1	User Testing/Evaluation	43
9.1.1	Inclusion of the Light Icon	43
9.1.2	Greater Control of Lights	44
10	Further Areas of Investigation and Enhancements	44
10.1	Further Investigation	45
10.1.1	Android Platform	45
10.1.2	iOS and Other Platforms	45

10.1.3	Functional Programming for Mobile Platforms	46
10.2	Enhancements	47
10.2.1	Integration With Other Smart-devices	47
10.2.2	Improve the Weather functionality	47
10.2.3	Increase Focus of An Intelligent Morning	48
10.2.4	Implement more smart-bulb functionality	49
10.3	Open-Source	49
11	Evaluation	50
11.1	Choice of Project	50
11.2	Course Relation	51
11.2.1	Mobile Application Development	51
11.2.2	Programming languages, concurrency and client server computing	52
11.2.3	Project management	52
11.2.4	User design and product evaluation	53
A	Application Designs	57
A.1	Hub Set-up	57
A.2	Light Tab	59
A.3	Alarm Tab	60
A.4	Weather Tab	61

B	
Application Documentation	61
B.1 Installation	62
B.1.1 Enable Unknown Sources	62
B.1.2 Copy APK to Device and install locally	63
B.1.3 Alternative Installation	63
B.1.4 Enable ADB on Device (Alternative Installation) . . .	63
B.1.5 Install ADB - <i>Windows 10</i> (Alternative Installation)	65
B.1.6 Installation via ADB (Alternative Installation)	67
B.2 Alarms	67
B.2.1 Create	68
B.2.2 Rename	69
B.2.3 Turning Off/On an Alarm	69
B.2.4 Delete	70
B.3 Lights	70
B.3.1 Connecting to the Bridge	71
B.3.2 Toggling the lights	75
B.4 Weather	76
C	
Activity Log	77
C.1 October	77

C.2 November	77
C.3 December	78
C.4 January	79
C.5 February	80
C.6 March	81
C.7 April	82
C.8 May	83

1 Methodology

A methodology is a set of methods, rules and restrictions combined to form a procedure or discipline.

By adhering to a development methodology it is possible to set development goals and a way to identify trouble areas during a project allowing for plans and contingencies to be allocated to improve the likelihood of success.

1.1 Overview of types of methodology

There are many forms of methodologies, below are a few of the most popular and an overview of each.

1.1.1 Rapid Applications Development (RAD)

By producing prototypes of the software quickly customers are able to test and provide feedback as the software is developed. This is useful as often requirements change and it's common for developers to produce software that isn't actually what the customer wanted.

1.1.2 Agile

Originally project management was slow to adapt to changes with user review coming in late stages of development. Agile however aims for incremental development with regular feedback. (Admin 2008)

The most popular form of agile development is the Scrum (Admin 2008) scrum is suited towards small teams and requires close involvement by the product owner to provide regular feedback and review.

1.1.3 Lean

Much like scrum and other agile methodologies, lean aims to produce software quickly and involves close coordination with the product owner, lean differs by reducing waste by selecting the most valuable features required. (*What Is Agile Methodology?*).

1.1.4 Waterfall

Focuses on phases such as; requirement gathering, analyses, development and testing. Each phase is completed entirely before moving onto the next phase and is often depicted by the phases flowing steadily downwards resembling a waterfall.

1.1.5 Spiral

The spiral model is based on the incremental model and consists of four phases; Planning, risk analysis, engineering and evaluation (*What is Spiral model- advantages, disadvantages and when to use it?*). A project will go through each phase multiple times in an iterative process or spirals. This is very well illustrated in figure 1.

1.1.6 Time Boxing

Involves strict deadlines rather than goals. By developing up to the agreed upon time and evaluating progress this can allow for steadier progress and a set time in mind which provides an achievable deadline.

Evaluating at the end of the time frame can show struggles in the development process and provides the ability to address them rather than simply spending more time to complete the goal.

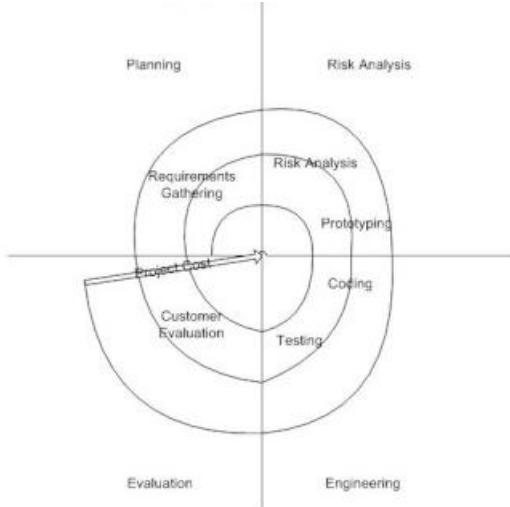


Figure 1: Spiral model diagram (*What is Spiral model- advantages, disadvantages and when to use it?*)

1.2 Choice of methodology

After assessing the various forms of project methodologies I have decided to use an agile methodology most notably the Lean methodology as this will provide me the ability to develop core functionality at a fast pace and add other features time permitting. To assist my development I will also be using time boxing to allocate time for my applications functions and allow me to perform regular performance reviews so I can identify time sinks and other issues to allow me to manage them.

1.3 Project Time line

Below is a Gantt chart of the overall plan for my project. A Gantt chart doesn't suit my development methodology very well and so is fairly high-level overview.

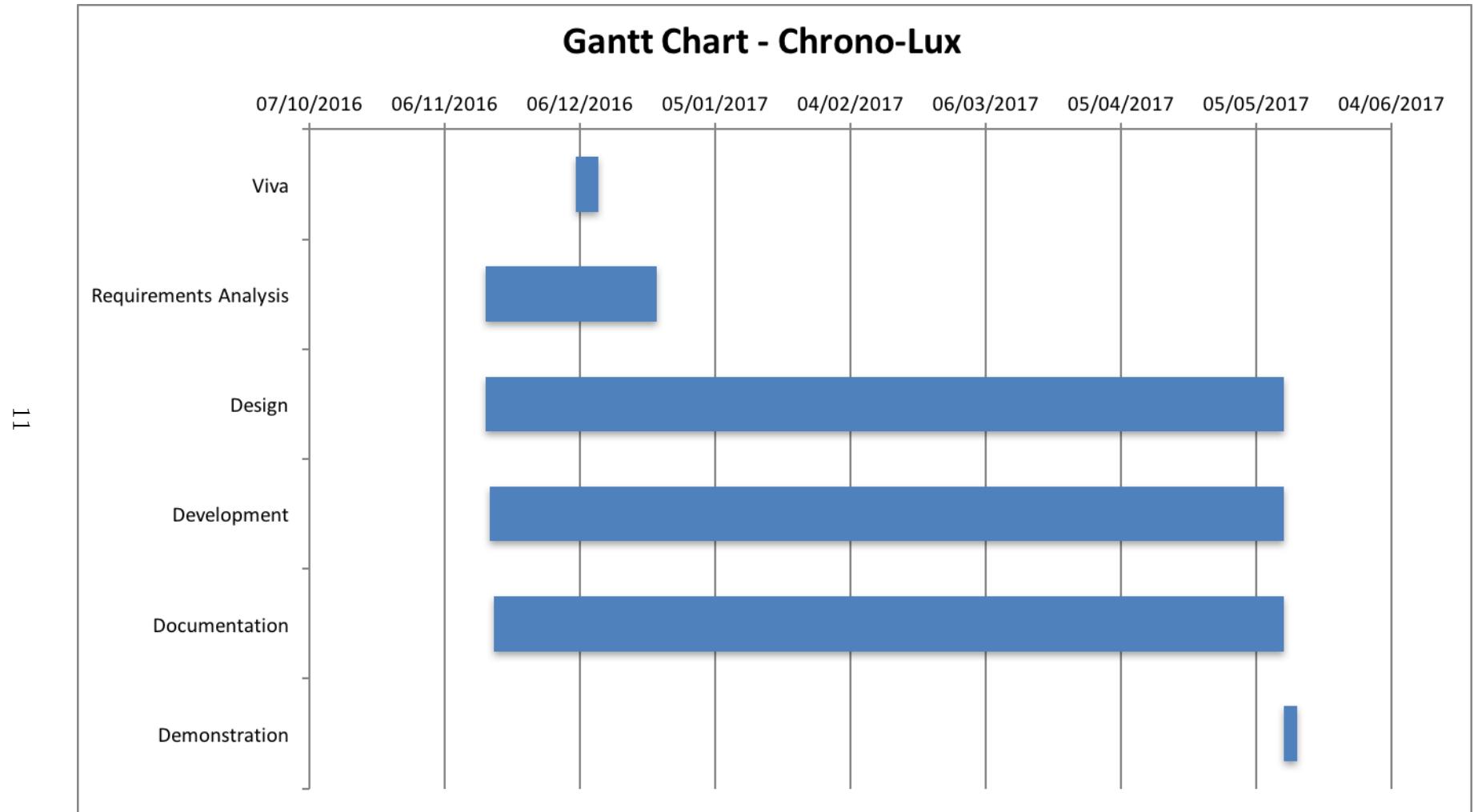


Figure 2: Project Gantt Chart

2 Stages of Development

For a complete project there are fundamental stages that need to be addressed, failure to address these stages can result in project drift resulting in the end result not fulfilling expectations or failing to achieve the requirements of the project.

It must be mentioned that although the term stages has been used, in many methodologies especially those that involve rapid development many of these stages will occur in either parallel or in quick succession; for example testing will often happen throughout development to ensure aspects of the code and design work as expected before extending the software further.

2.1 Analysis

The first stage of development is analysis, failing to analyse the problem or obtain customer requirements renders development practically impossible as the project scope can not be defined. Analysis often requires the assessment of the problem or product to be developed and rendering it down into more fundamental pieces such as how to implement a RESTful API or store any data required. During this stage it is common to perform competitor analysis by finding similar existing solutions to challenges that may be faced and to implement improvements over the competition and produce something objectively better to improve usability, experience, performance and any combination of aspects that can be deemed desirable.

2.1.1 Competitors

In this section three competitor apps will be investigated and evaluated by the functionality and appearance.

Sleep as Android This is the most feature packed alarm app available on Android that I could find.

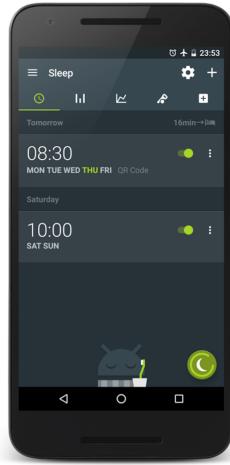


Figure 3: Sleep as Android Screenshot

Appearance The design of the app mostly follows the material guidelines set by the Android development team with small variances in style. Overall the app looks nice however some views can become crowded with information and the number of features does make navigating the app harder than it needs to be.

Features This app contains numerous features from the basics such as recurring alarms to features such as solving a puzzle or scanning a QR code to turn the alarm off. Features such as snore detection, sleep sensors and sonar detection all detect and measure a persons sleep cycle and provide statistics and graphs to assess the nights sleep and help inform the user how to improve their sleep quality, suggesting an earlier bed time or to avoid sleeping aids and alcohol to prevent snoring.

Cuckuu Cuckuu doesn't have the same level of integration with reminders, appointments or weather as what I intend and it doesn't include any smart bulb integration.



Figure 4: Cuckuu App Screenshots

Appearance Cuckuu strikes a balance between Android and iOS allowing for it to be available on both platforms, this is a good choice as it allows a user to move between platforms and remain familiar with the design while maintaining a good level of OS cohesion to keep the experience consistent enough.

Features For Cuckuu social aspects of an alarm are the most important thing as this is the apps unique selling point. Users are able to share alarms (cuckuus) with their friends, possibly to alert them of shared events.

Users can add media or links to alarms both their own and shared alarms, this provides a more visual and engaging alarm over a simple snooze and off button. Another key aspect is a level of gamification towards alarms by obtaining points for shutting off alarms quickly and providing a leader-board among friends to encourage waking quickly and avoiding the snooze button.

Wakie Wakie is very different in how it intends to wake a user up and besides being an alarm has little to what my app will consist of. Wakie does have a very interesting concept though, that a stranger from around the world

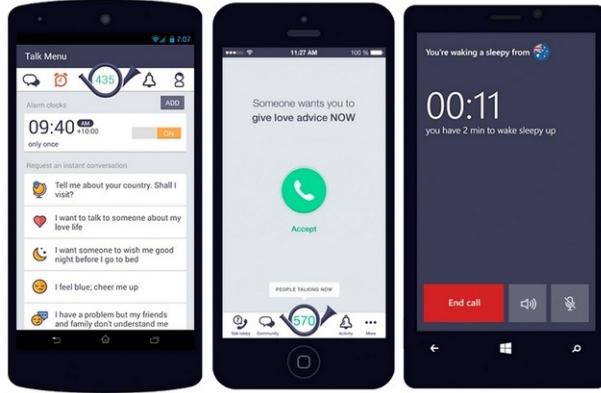


Figure 5: Wakie App

will be able to call you when you would like to be woken and talk about a topic you would like to discuss.

Appearance With a very simple and clean design this app definitely looks nice and allows itself to easily be ported into the varying mobile platforms.

Features Since the initial analysis of this application the alarm aspects have since been removed, as such I will be writing about the app in regards to older versions that still contained the alarm functionality.

Overall the app works simply, you can set an alarm and another user of the app will be connected to you through a phone call at the time the alarm is set for. The concept is to provide a more sociable and personal alarm, knowing that your alarm is a person on the other end possibly encourages the user to wake up more than a simple alarm. The app and it's functionality are free and only users of the app are connected together.

2.1.2 Problem Analysis

Through my competitor analysis it is clear there is little competition for smart-device connected alarm applications and with the increase in smart-device

popularity I feel there is a market for such an application. The problems with existing alarm applications can be attributed to their functionality, in some cases the amount of functionality can be overwhelming in the case of *Sleep as Android* I feel there are just far to many features and this produces a cluttered experience.

By focusing on a small amount of features such as how Wakie focuses on an alarm with a chat, I intend to make a clean and simple app that is familiar to the material design guidelines for the Android platform to make my app blend seamlessly.

2.2 Design

During the design stage the User Interface (UI) and various aspects of the code are planned for development. By critically thinking about certain design aspects with regards to the requirements outlined from the analysis stage, designs of the software can be produced, this includes both the visual designs of the user interface and some of the software solutions that may be required such as certain tooling and frameworks that are available for development to produce more stable software faster by removing the need to produce custom solutions to aspects that have been solved and are found throughout programming.

2.2.1 Design Considerations

The application will often be used within a low-light or dark scenario and as such I will design the application to use a dark colour scheme.

A light colour scheme could be quite energising to the user and could potentially make them more alert which is not ideal before going to sleep and could extend the duration it takes to fall asleep, resulting in restlessness.

It has been shown blue light in particular can affect the circadian rhythm of the body which is used to regulate sleep patterns Oh et al. 2015.

2.2.2 Designs

The pre-productions designs for the application can be found in appendix A

2.3 Development

Throughout development everything that has been produced from both the analysis and design stages are implemented through creation and utilisation of software and assets such as images and sounds to produce the goal of the project, in the instance of a mobile application this would consist of producing an application that can run on the required platforms such as Android, iOS or Windows Phone and for the application to perform the tasks required such as sending and receiving messages and displaying them to the user in the case of a communications application.

2.3.1 Platform

The development platform chosen is for Android devices, this is due to the lead market share of mobile devices and through the relatively simple and free development platform using Android studio and the Android Virtual Device (AVD) tools provided, with the AVD providing the ability to test applications on every version of Android available and with different device specifications such as screen size, screen density, on-board sensors and orientations.

With my personal phone being an Android device it is also possible to install, test and debug my application on a physical device which is often faster than a virtual device and can provide a better experience of what the application will be when deployed.

2.3.2 Target Market

Through various factors including my chosen platform, I will be targeting the latest version of the Android platform 25 also known as Nougat with support

for devices as old as version 21 known as Lollipop. My decision is based on the current version distribution statistics that show over 50% of devices are on versions 21 or higher and as such my target is the majority share.

Version 19 (KitKat) currently holds a 20% share which is a large portion, however these are older devices that are likely to be from poorer markets and due to the cost of smart-devices it is unlikely for there to be a high demand of support for my app from KitKat users. Due to the release and support cycle provided by Google it is also likely for KitKat to be unsupported by the end of the year and as such I feel there are greater benefits to gain from the newer versions in place of using older development features and supporting deprecated code.

Due to time limitations support will only be provided in English, with the app having no alternative language support. From the restrictions and targeted versions I will be focusing on countries that have English as the native speaking language and areas of relative wealth as these are the most likely to have access to the required smart-devices.

2.4 Testing

Often testing is an ongoing aspect of the development life cycle so that as aspects of the application are developed tests and unit testing classes are produced to allow the automation of testing. Performing ongoing testing is useful to indicate that code is working correctly and if during development the automated testing indicates any issues these can be resolved as they occur instead of during feedback or after deployment, both of which increase development cost through having to hunt down bugs and maintain the application during the life time of the application.

2.4.1 Tests

The majority of testing involved will consist of introducing functionality into the app and performing various actions to test that functionality by using it as intended and trying to break it through fringe cases and interactions that

are likely to be less common such as rapidly editing a text input or using input that is likely to be uncommon such as number in place of text.

Further testing in the later stages of development will use others to use the app and to provide feedback for their experience. It is very difficult to test all cases as issues with UI or networking can be intermittent and as such development should involve writing good code and be less reliant on tests to indicate when something isn't working.

2.5 Feedback

Feedback is crucial for assessing the success of a project, by obtaining feedback from testers or trial users changes can be made to improve the experience of the application. Feedback can be obtained during development by having people use certain aspects of the application as it's being developed. If during feedback multiple users raise complaints about the same aspects this is a key indicator that it requires changing before full deployment to ensure a higher level of polish and maintain a respectable and image.

2.5.1 User Trials

As mentioned later stages of testing will involve others to test my app, through testing I can obtain feedback for the design, usability and highlight issues that I may not have accounted for in my development and testing.

3 Specification

Through performing the initial development stages the following project specification has been developed. Due to my chosen methodology various aspects of my specification can be slightly flexible though many are fundamental aspects the project.

3.1 Deliverables

My project has multiple deliverables that I will assign a level of priority to using the must, should, could concept.

The deliverables for my project will be as follows;

Activity	Priority	Deliverable
Writing final report	Must	The final report for the project is crucial and will be undertaken through out the development of my application.
Developing a basic alarm	Must	The application, an alarm app is what my project is based upon and so I will need to develop this before I can develop any further.
Developing a better alarm	Should	The application, a basic alarm will work but I would like this to be an alarm app that has all the features found in any alarm app such as repeat alarms, multiple alarms, etc...
Smart bulb integration	Must	The application, the Smart bulb integration is a main basis of my project and so I will need to include this into my application.
Develop for smart bulbs	Should	The practicality of demonstrating smart bulb integration could be a challenge, by setting up a test platform using an Arduino or raspberry pi would allow me more control than external APIs.
Calendar Integration	Should	The application would be improved with calendar integration providing an agenda for the user in the morning.
Text to speech	Could	The application, text to speech is a part of the Android platform and can be used relatively easy and so doesn't contain much of a challenge to it's implementation.

3.1.1 Stages

It would be naïve for me to provide a detailed schedule of the activities and stages for my project, however I can identify what I will need to do and in which order as well as estimate a time frame for when I intend to begin the activity.

Activity	Time Frame
Writing the report	Writing the report will be on going throughout the development of my application to allow me to assess my work, identify my challenges and to provide the technical research I undertake.
Design	First week will be spent on this. I don't intend to spend much time of the visual design of my application as it should be fairly simple and can be modified easily.
Prototypes	Will occur prior to new visual changes to my application. I will provide a dummy function to the visual elements that involve interactions to test the look and feel before implementing fully.
Basic Alarm	First four weeks, the basic alarm will be the foundation of my application and as such will need to be developed well using the principles I have learnt.
A test platform	Two weeks following the alarm development, this will take some time to develop

3.1.2 Risk Analysis

There are many risks present with any kind of project, I will be identifying the most relevant and predictable of these and assessing the impact that could be caused. By identifying the risks posed I can attempt to avoid and mitigate them and plan for those that I can not control.

Table 3: List of risks.

Risks	Impact level	Reaction
Sickness	Low	Avoid getting ill.
Data loss	Low	Mitigate risk with multiple backups and version control.
Project complexity	Medium	Avoid making it too complex, or too simple.
Scope creep	Low	Avoid implementing features not outlined.
Communication with supervisor	Low	Reduce by keeping in regular contact.

4 Research

Aspects of my project were influenced by the research collected before and during development, resulting in changes from what was originally planned.

4.1 Philips Hue API

My project utilises the Philips Hue connected light bulbs, the Philips 2016 system uses the ZigBee Alliance 2016 standard, though all of this is transparent as the method of interfacing with the devices is to use ‘GET’, ‘PUSH’, ‘POST’ and ‘PUT’ URL requests and provide JSON formatted commands in the body to interact.

The state of a specific light can be received using ‘GET’ and providing the URL /api/devID/lights/1 or all of the lights by not specifying the number.

The state can be changed using ‘PUT’ instead and providing attributes and their values that you would like to change, for example:

```
1 {"on":true, "bri":255}
```

A few useful attributes for my application are: on = true/false bri = Brightness between 0 and 254

Colour settings include: sat = Saturation between 0 and 254 hue = The hue of the light (hue runs from 0 to 65535)

Through researching into the Hue API further there existed a Hue Software Development Kit (SDK), by using implementation of the Hue SDK most interactions with the lighting that would require the use of the RESTful API are now able to be accessed using Java methods. The Hue SDK exists alongside a very basic example Android application that allows the connection to the Hue bridge used for interfacing with the lights and a button to randomly adjust the hue values of the lights connected to the bridge.

4.1.1 Influence

The implementation example of the bridge connection and the storage of the API key required to authenticate a connected device/application included was very useful for the initial set-up for interfacing with the Hue lighting platform and as such a direct use of the RESTful API has not been implemented within my application to interface with the Hue lighting but instead the Hue SDK.

4.2 Design Considerations

Through my research I have found that it has been shown bright lighting, blue light in particular can affect the circadian rhythm of the body which is used to regulate sleep patterns Oh et al. 2015. There are applications used to reduce the amount of blue light displayed in the evening and through the night, applications such as flux (*F.lux - software to make your life better*) and features included within mobile operating systems such as iOS (Swider 2016) and Android platforms (Diaz 2016).

4.2.1 Influence

My application will often be used within a low-light or dark scenario and as such I will design the application to use a dark colour scheme.

A light colour scheme could be quite energising to the user and could potentially make them more alert which is not ideal before going to sleep and could extend the duration it takes to fall asleep, resulting in restlessness and a poor nights sleeps.

4.3 Human Perception of Light

Many human senses are based on a logarithmic scale, this is to say we are far more able to distinguish changes in light or sound in the lower band of the senses compared to higher, as such a small increase in volume of a whisper will be a more distinguishable change in volume than two jet engines roaring and increasing by their volume by a small amount.

The same applies to sight, it is more important to distinguish details in low light such as that from the moon compared to the light change of daylight at varying times of the day. We do this to normalise our senses to best suit our environment.

This kind of stimulus perception is defined as the just-noticeable difference (JND). First summarised by Ernst Weber in 1834 his equation was called Weber's Law and simply stated that response intensity increases as stimulus intensity increases (Salkind 2010, p. 1613-1615). Further refined by Gustav Fechner who proposed the use of a constant to provide a curve to the stimulus/perception relationship. Fechners' law was a much better fit, however some stimulus did not fit well, such as that of electric shock.

Most recently in the 60's an American psychologist S. S. Stevens produced a formula that worked for all forms of stimulus, even for electric shocks (Stevens 1957). He proposed an exponential function raising the data to a power rather than using a simple constant. This essentially stated that to get a linear increase in perception of various stimuli, the stimulus would need to increase

in an exponential form.

4.3.1 Influence

Due to this perception of light I feel increasing the brightness of the lighting in the room in a logarithmic fashion will produce a perceptibly more natural and linear increase in brightness over increasing the value directly from 0 to the maximum value.

5 Assessment of Progress

In this section I will be assessing the progress that I made during the development of my application.

5.1 Early Stages

The early stages of my project involved the analysis and design stages of my application. By performing my competitor analysis I was able set myself a guideline for what my app should look like and a concept of how I'd like my users to interact with the app.

I decided that due to developing for Android only I was able to embrace the material guidelines fully and trying to ensure that my app would provide a very native Android experience allowing my app to blend in nicely with the rest of the stock applications on the device included with the OS.

My app consists of a fairly basic layout, following the guidelines it was made clear to group functionality by tabs and to only allow actions associated with the tab; for example if from the lights tab the user were able to manipulate the lighting, this would go against the design guidelines.

Some design guidelines posed a few problems and required work to get them working fully. One such example of this is the coordinator layout

(*CoordinatorLayout*), to be able to utilise androids latest pop-up notification system known as the Snackbar, it is required to use a coordinator layout to wrap the content of the app, by doing this certain views within the app move and react to the Snackbar being shown. Although once configured correctly the coordinator layout is easily added to other views within the application.

5.2 Development Progress

The development of the application was quite a challenge for many reasons, many I had not foreseen in the conception or problem analyses stages of development. As I wanted my app to utilise tabs I had a few means of achieving this though several involved deprecated methods that are officially unsupported and are from older versions of Android, these features would still work as Android still works with these older methods to ensure older apps are supported.

5.2.1 Fragments

As I would like to develop to the latest standards however I used a ViewPager with what is known as *Fragments*. Fragments are like Activities in that they hold behaviour and the user interface, a Fragment however is not a stand-alone aspect of the application and requires an underlying activity to create and display the Fragment. Fragments come with many benefits; they can be re-used throughout the application and multiple fragments can be displayed within a single activity, this is often seen with communications apps where if viewed in portrait a list of messages are displayed and when a message is tapped a new view displays the full message, if both of these views are Fragments both can be displayed when the device is in landscape, displaying the list on one side and the entire message on the other.

Fragments however due to being at a higher level running within an Activity do not have direct access to the application context, with the application context being the current state of the application such as the current activity or Fragment being displayed. This posed less of a challenge and more of a need to learn how to obtain the required context as in some instances simply

using the `getContext()` method worked perfectly, in other instances the need to first get the application activity and to call the `getContext()` method on that instead caused a few issues when testing my application causing it to crash if the incorrect one was used.

5.3 Testing and Feedback

Testing and feedback have been grouped together as although I was testing my application during development key testing came from the users testing my application.

5.3.1 Debugging

The use of the Android Debug Bridge (ADB) (*Android Debug Bridge*) made inspecting my application relatively simple as I was able to monitor for resource usage and any error messages displayed. One issue with the use of ADB is that it is possible for the device to provide log information for a lot more than just the application being developed and this could often cause a lot of information being displayed and obscuring the information most relevant.

It is possible to filter this information to the application name, however even the application could often produce large amounts of log information.

Another issue with the debug information is when an exception error was thrown or the application crashed there would be a very long stack trace displaying messages for core OS methods. It is possible however to produce log information using the provided Log API within Android, this API can be called using `Log.i(tag, message)`, this outputs a log message with a *tag* string that can be provided. The use of the tag allows the developer to filter log message by using the tag allowing for only the messages that developer would like to see be displayed, greatly reducing the clutter. The Log API can be called using different letters as seen in table 4.

Table 4: Log API Calls

Letter	Log Type
v	Verbose
d	Debug
i	Info
w	Warning
e	Error

Verbose should not be used in a deployed application as much like debug, the log is stored, however verbose provides more details about the running application and could present a security risk. ‘Debug logs are compiled in but stripped at run-time. Error, warning and info logs are always kept’ (*Log*).

5.4 Problems Encountered

Throughout the development of my app many challenges where faced and problems occurred that required resolution.

5.4.1 Alarm creation

Before developing the alarm aspect of my application I investigated into how to create alarms within Android, this is were I looked into the `AlarmManager` class. The `AlarmManager` class allows scheduling of an activity of the application be run at some point in the future.

The `AlarmManager` allows the specification of the time the alarm should trigger in either elapsed real-time which is the time from device boot in milliseconds, or by the Real-Time Clock (RTC) based on the Coordinated Universal Time Clock (UTC). For my purposes the RTC clock is the one required as it allows the setting of an alarm to occur at a specific time instead of occurring a set value of milliseconds in the future.

Using the `AlarmManager` it is also possible to set an interval period, for my

application setting the interval to a day would allow the alarm to be triggered daily at the specified time. Within the developer guidelines it is recommended to use an inexact repeating alarm as this allows the device to handle multiple alarms at once when the device wakes; however I required the alarm to go off at an exact time as inexact allows for the alarm to be triggered for almost a full interval of the chosen interval time, in my case a day (*AlarmManager*).

Solution This is the extent of the *AlarmManager* class and as such it doesn't cover any other aspects of what would be regarded as an alarm such as; an alarm tone, a simple ability to toggle an alarm on/off, hold any extra details for reoccurring alarms by various days such as Monday-Friday but not weekends. Due to these limitations I was required to write my own *AlarmManager* class that utilised the *AlarmManager* and stored the extra functionality that I required.

5.4.2 Time Picker dialog

I used the time picker dialog included with Android to allow the users an attractive way to enter a valid time for the alarm creation.

The alarm picker has a method that listens for the confirm activity within the dialog and from here I am able to call `delayNotification(hourOfDay, minute)`; that takes the time selected within the dialog and create the alarm using my alarm manager. Due to the time picker dialog method being of a void return type I was unable to obtain the time chosen from outside of the method, this restricted how I could utilise the time picker as it required the need to create the alarm at the same time making adjusting the time of an existing alarm difficult.

One option for allowing the adjustment of an existing alarm time would be have global variables for the hour and minute values set within the time picker listener; I did not like this option however due to the less predictable nature of my application caused through using global variables.

An alternative to this approach would be to create a class that would be able to handle picking the time and be able to return the hour and minute values.

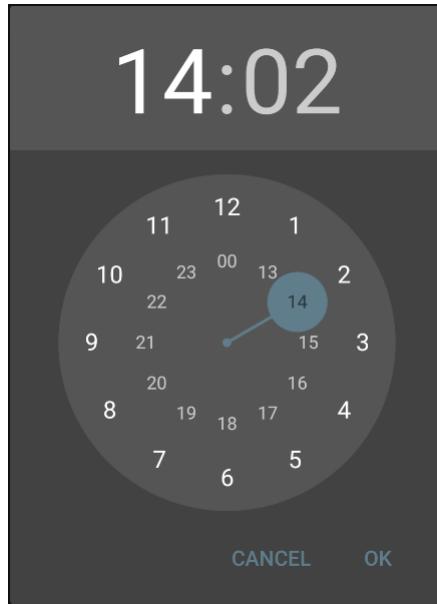


Figure 6: Time Picker Dialog

This would be less than ideal as the time picker dialog would be created in a different class than the alarm fragment but requires the need to be able to displayed on the UI thread and this posed certain problems.

Solution I feel there is still a better way to handle the time picker, due to working on other aspects of my application however I ran out of time to be able to implement a better method for selecting the time for the alarm. The user is able to pick their alarm time as required though by having the alarm creation called from within the dialog listener there is no way to edit an existing alarm, this means the user would need to create a new alarm and delete the old one if they would like to set an alarm for a different time.

5.4.3 Handling Pending intents

Within Android there is a need to specify an Intent to start a new activity or fragment. The Intent is a fairly simple class that allows for activities to be called/started and provides a data structure that holds the activities and

tasks in a back stack, this allows for the use of the back button by closing the current activity/task the app can pop the activity from the back stack and present the previous element in the stack.

A new intent is created as follows:

```
1 Intent intentName = new Intent(activity.class, context)  
2  
3 startActivityForResult(intentName, result)
```

It is also simple to pass data between intents by the use of `addExtra` which provides a means to add a key-value pair that can be accessed. The use of the result in the above Intent example will request that the intent return a response, this can be used to handle various exit values of the intent to be able handle errors or developer specified return values.

The use of a Pending Intent allows for a foreign application to launch an Intent with the permissions of the application the Pending Intent was specified from. This is useful as it allows for an Intent to be launched by the NotificationManager and the AlarmManager, allowing for an activity to be launched even when the application is not open.

Cancelling pending intents is not as simple as creating one, cancelling a pending intent requires for a new pending intent to be specified just as the original was and to pass `.cancel()` as the last method, if the pending intent exists it will be cancelled and if it did not nothing else will happen.

Solution By storing a reference to any created pending intents, it becomes trivial to be able to cancel a pending intent, by simply obtaining a stored reference and calling `.cancel()`.

5.4.4 Storing Objects

For my alarms to be persistent I was required to store the alarm objects by some means so that they could be created, updated or deleted.

Within Android there are several methods for storing data persistently these

include using; Shared Preferences, internal/external storage or an SQL database. Each of these storage solutions come with their benefits and restrictions.

Shared Preferences Shared Preferences allows the storage of key-value pairs within storage, by storing the key-value pair data can be stored and accessed by the usage of the key. Shared preferences although simple to use has the restriction of only being able to store string values within the value of the key, as such this greatly restricts what can be stored.

It is possible to store an object within the shared preferences by converting the data into a JSON object by use of an interface library. This is mostly suited towards small and simple objects as changing a stored value would involve retrieving the entire object from memory, altering the JSON associated with the change and then storing the data back as a JSON object.

Due to these limitations it would be a poor choice for storing my Alarm objects.

Internal/External storage Storing data to the internal and external storage is about as challenging as storing data within the shared preferences. All data stored in the internal storage can only be accessed by the application, preventing other applications or direct user access to retrieve the data. External storage allows for the data to be accessed freely by other applications or directly by the user by browsing through the device file system.

Both forms of storage however are better suited for media and files such as images, music or text files. Again it would be possible to store the object as a JSON object and access it much like how shared preferences would provide access and due to this comes with similar restrictions.

Database By using a database this is the most involved method for storing object data with each row of the database table able to store an object variable and provide direct access to each variable through interfacing with the database.

The database can be interacted with directly by executing raw SQL statements

and queries or by the use of a database helper which allows a method means of interacting with the database.

A database is stored with the internal storage and as such only allows access to the application and prevents access from external use, this is ideal as there is no need for another application or for the user to have direct access to the database.

5.4.5 Alarm Solution

I used an SQLite database as it provides all the functionality I required and kept my project simple. One restriction of using SQLite is that there is no dedicated Boolean type and instead a Boolean value can be stored as an integer value with 0 representing *false* and 1 representing *true*, this was a simple enough restriction to work around.

Within a day I was able to get my alarms stored persistently and another day to be able to completely implement all required Create, read, update and delete (CRUD) operations.

5.5 Lessons Learned

Knowing target market is important due to the variance of deprecated methods and functions. It is possible to develop for only the latest version of Android and this comes with the benefits of being able to use the latest version of Java, as of writing this the latest is version 8; this version comes with many useful features from the use of lambda functions, to the ability to use mapping functions and allow for the simple parallelisation over a group of elements, of which can lead to improvements in both speed of the action called and in the way the function is called by the developer leading to more readable code and a better ability to reason about what is happening.

However due to the current market share of the latest version of Android and the relatively slow adoption rate of newer version by phone manufacturers when newer versions are released, a majority of the market would not be

catered for, thus limiting the reach of the application.

The separation between UI and the rest of the application can cause certain issues; for example running code from within a fragment that is not the current running activity will cause a `runOnUiThread undefined` error in most cases, this is because a fragment does not have a `runOnUiThread` method as it runs as the fragment may have become detached from the activity and so the fragment may have been destroyed by the time the method comes to run. Due to the unknown nature of the fragment it is required to get the current activity when running and to check that the activity is not null before running.

Application security is good in Android, through the use of sand-boxing and defaulting to keeping data private, requiring the developer to specify actions that are *unsafe* it is possible to prevent app hijacking or having data leaked from within the app. Devices running the latest versions of Android are encrypted by default, this provides an extra level of protection on a device as when the device is off it is harder to retrieve data from the device where it used to be an attack vector by exploiting the ADB to gain access to the internal storage.

This extra level of security is not of much importance to my application, it is something learned through the investigation of the different means of storing data on the device however and could be useful for future developments.

6 Accomplishments

The following are some of the accomplishments within developing my application of which I am most proud.

6.1 The Ability to Control Lighting

There are few applications available that allow for the connection to and manipulation of smart-lights from within the application. Because of my app

being one of these I feel is an accomplishment.

6.2 The Application UI Design

Overall I feel the look and feel of my application works very well; it adheres to the Android design guidelines (Android 2016) and feels like a native application by maintaining a cohesive user experience.

6.2.1 Tab Navigation

The tab navigation works well, providing the user with a simple and clean way for using my application. By providing only the relevant functionality for each tab the user experience is simple and intuitive. Through avoiding too much being displayed on the screen at once, relevant information can be seen instantly ensuring ease of use and allowing the user to only spend as long as necessary within the application.

Through including tab navigation it was required to use fragments within my application, fragments are not as simple to use as a regular activity and pose multiple design challenges over an application activity. The extra difficulty associated with fragments did slow the initial development of my application and my initial tab view needed to be replaced due to the use of several deprecated styles. Despite this challenge I feel the fragments work well within my application and the functionality I outlined within my project has been accomplished.

6.2.2 Snackbars and Toasts

By developing for newer versions of Android I utilised the snackbar notification system introduced in API level 23. The snackbar in most instances replaces the original Toast notification system. Snackbars improve upon toast in many ways as they; provide information to the user and allow the user to interact with the notification by performing an action if provided by the snackbar, such as undoing an action that triggered the snackbar to be displayed. The

Snackbar can also be dismissed by swiping, while the toast is displayed until the specified display time has passed.

Toasts are still used within the newer versions of Android, however they are now often used for system notification as they can be displayed without being associated with an activity and as they can't be dismissed are good for showing warnings or important information. Snackbars also blend better within an application, showing up above or moving elements such as a floating action button, ensuring that the notification can be seen and that aspects of the view are not obstructed.

It is specified within the guidelines that a Snackbar should be displayed as long as necessary, as such short notifications should not be displayed for a long time, while notifications that provide an action or consist of a lot of text should be displayed for longer to allow the user to fully read the text or provide adequate time for interaction.

6.2.3 Using the Weather Icon Font

The usage of the weather icon font provides multiple advantages over that of PNG, JPG or SVG graphics. Of these graphics types JPG doesn't allow for transparency and so would require a background colour to be saved, this would not only increase the storage space required to store the image, but would require the image to be edited if the background colour of the view were to be changed, making the use of a stored background difficult to manage. PNG does allow for transparency, however PNG images requires a relatively larger amount of storage space when compared with JPG.

PNG and JPG are bitmap image types, meaning they store all the pixel data such as colour values and luminosity among other values as essentially a three dimensional array over the colour space. Due to the means of storage the graphics are not suitable for scaling, especially to increase the size of the graphic as this would require the need to generate data that does not exist; for example in an image that is 800×800 pixels, there would be 640,000 pixels, if the image were scaled to 1000×1000 pixels there would need to be 1,000,000 pixels and as a result 360,000 pixels need to be generated from the existing image, resulting in processing artefacts, banding or other graphical

errors.

SVG is a vector graphic and consists of instructions on how to draw a graphic and this is handled when the image is to be displayed. Due to this flexible drawing ability, the graphic can be generated to be as large as required with a trade off of computational time for storage, compared to bitmap images. Android is now capable of handling both types, however this has not always been the case. SVG support has been officially supported for versions above 4.4 (KitKat) with many adopting workarounds using the webview library prior to support (*Add Multi-Density Vector Graphics*). As a result there are many poor implementations of using SVG and the current advice provided in the Android guidelines is to generate PNG graphics from SVG files as to support older devices. When the PNG files are produced multiple sizes can be generated to support devices of varying screen sizes and in doing so requires multiple sizes of the same file to be saved, taking up more storage space.

By using a TrueType Font (TTF) it is possible to gain the benefits of an SVG without the need to generate resource files or use unsupported implementations to display the graphics. Other benefits include the ability to easily colour the icons by changing the font colour, the font appears as a single file within the Android project resulting in less clutter in the project structure. Lastly the size required to store the font consisting of hundreds of icons is relatively small as it requires 98kb of storage while a single large PNG around 30kb.

7 Success and Failure

In this section is an assessment of some of the successes or failures of the project as a whole.

7.1 Successes

Within the project there were some highlights of success that are mentioned as follows.

7.1.1 The Project

As outlined within the project specification several goals and stretch goals were set and if the goals were meant this would regard the project as being a success. The application provides; alarm functionality, smart bulb integration and weather functionality each of which are the main goals set and as such the project was a success.

7.1.2 Using SQLite

I had not foreseen the need to use a database within my application as the storage of alarms had not seemed to be a complex enough scenario to need a database. As the other storage methods available were not suited for storing an object with multiple variables however it was necessary to utilise the SQLite database for storage.

Though more complex than other storage methods, the use of database helpers reduced the need to write raw SQL queries and requests and allowed for a more programmatic style to interface with the database. Overall the database worked well.

7.1.3 Pending intents

When developing the alarm and using the alarm manager it was necessary to use pending intents. Pending intents are simple enough to understand however their implementation is a little odd, as cancelling a pending intent requires another matching pending intent being produced, however the use of the `cancel()` method at the end cancels the pending intent.

Due to this and the need to allow the user to enable and disable an alarm, it was necessary to leave the pending intent and instead to check the alarm that was being triggered was enabled. In doing so there is no longer a need to create and delete pending intents for alarms that had been disabled, as the alarm object would still exist within the database and could cause errors if there was an ID mismatch.

7.2 Failures

Although the project was successful, there are several aspects that failed to be implemented or did not make it into the final application.

7.2.1 No Adjustable alarm

Due to the time picker not returning values, the alarms are created from within the listener method, because of this it made it difficult to be able to allow for alarms to be changed once set. The lack of ability to pass values into the time picker means that there is no way to pass an alarm id into the method either, something that could allow for a check to see if an alarm by that ID already exists, if the alarm does exist adjust the time and if not create a new alarm.

7.2.2 Not Reaching Stretch Goals

This could be classed as not being a failure, as the main goals outlined have been met. However I feel that the stretch goals would have worked well within my application and helped it to stand out more through being more than a regular alarm application, more so than the inclusion of controlling smart-lights.

8 Original Project Plan

The following is the original project plan for my application.

8.1 Aims and objectives

What I will be developing over the upcoming months is an app to help users get out of bed easier in the morning in a useful and information rich way.

My aims are:

- Produce an alarm app with all the functionality users are used to.
- Integrate Smart bulb functionality into the app to turn the light on in the morning with the alarm.
- To turn off the lights at night without having to get out of bed.
- Provide weather information for the day.
- Inform the user of their schedule for the day and upcoming events.
- Publish the application to the play store for download and use by others.

8.2 Stakeholders

I have identified the following stakeholders:

- Myself - Not only am I developing the application making me a stakeholder, I am also very interested in home automation and waking up happy.
- My supervisor, Marcus Winter - By accepting to be my supervisor Marcus is also a stakeholder for my application.
- The second reader - Will also be involved and will be grading my project.
- An expanding user base of smart bulbs - Although the market currently is small the cost of smart bulbs is decreasing making them more available to users.
- Anyone that uses an alarm - The largest stakeholder I have is anyone that uses an alarm on the Android platform. More specifically those who own smart bulbs or other connected devices.

8.3 Communications

I will maintain contact with my project supervisor with monthly meetings where I intend to measure my progress against deadlines and goals, reflect

on what progress has been made and address issues, challenges and for development advice to assist me successfully complete my project as planned.

Regular emails will also be used between meetings to keep in contact and keep my supervisor informed of what I intend to talk about and on my progress made.

8.4 Installation Process

I would like to publish my application to the Play store once developed to a satisfactory level. To install an app from the Play store you can simply select the option to install the app and it will be downloaded and installed seamlessly.

I will be ensuring to develop to a high standard and ensure there are no issues, bugs or flaws with my application.

8.5 Quality checks

During development I will ensure to maintain my code and follow the principles that have been taught to me and that I have learned and will learn, in doing so my code should be easily maintainable, readable and extendable for possible extensions and stretch goals.

I will develop a test plan as I continue to develop my application to allow me to note issues and ensure previous functionality has not been effected by further developments.

8.6 How will I measure success?

My key performance indicators are outlined below:

- Alarm functionality

- Smart bulb integration
- Weather functionality
- Calendar Integration (Stretch)
- Text to speech (Stretch)

If I am unable to produce a working alarm app with smart bulb functionality I will have failed to achieve what I intended to develop and so these are my highest priority.

8.7 Challenges

There are many challenges I will face during my project these include the following:

- Making my application extensible to other home automation systems such as the *Belkin Wemo*. This would allow for greater flexibility and a larger target audience.
- Handling various devices, there are now many various appliances that are connected to home automation, from washing machines and fridges to door locks and CCTV. Many devices could be useful for morning and night automation, such as closing the curtains at night or turning the kettle on in the morning. Each of these devices will have different functions and are very different from one another. To be able to add several and include them within a scene/scenario to perform multiple actions would be a very powerful inclusion within the app.
- Providing a good level of home automation without making the app cumbersome or difficult to use. My target audience is average users who would like to wake up easier and more refreshed and have their days information available straight away. They don't want to spend ages setting up the light bulbs or struggling to find settings as they will get fed up and stop using and potentially uninstall the app.

9 Divergence From Original Plan

No project goes through all the stages of development and remains exactly as outlined during the first stages of development. This is attributed to many factors that may not have been apparent at the onset of development, many changes are small involving adjustments to the UI to improve user experience; these changes can be seen in completed products also, with many applications receiving design overhauls such as the introduction of the ribbon design to the Microsoft office suite for office 2007 (*Microsoft Professional Developers Conference 2005 VPR: Image Gallery 2005*) and the regularly changing design philosophy of Android.

9.1 User Testing/Evaluation

Through getting others to use my application I was able to gain useful feedback to alter the design and functionality of my application to improve upon the designs.

Due to the need for a hue lighting bridge to be available, it was not always possible to obtain hands on feedback from users and instead required the need to obtain feedback through the use of screenshots and describing the actions available to the user; through this limitation it was not possible to gain the most feedback possible, but should provide a good insight into the design and enhancements for the application.

9.1.1 Inclusion of the Light Icon

Several users suggested the inclusion of a lighting icon to indicate the type of light being listed, this is because of the ability for lights to be renamed and could lead to confusion in the future.

The icon provides at a glance a hint to the type of light being shown and can assist the user. Within the application the icons used are freely available and provided by Philips under the development program to promote development

and use of the Hue lighting system (*Icon Pack* 2016).

9.1.2 Greater Control of Lights

Many users suggested a greater level of lighting control within the application, features such as; the ability to connect with multiple bridges, add new lights to the system, adjust brightness and colour. Each of these features could make for a more enjoyable user experience and allow the users to perform many function from within the application, however due to complexity in certain aspects such as allowing for multiple bridges, these were not possible to be included within the application.

Features such as the brightness and hue controls were not included as a design limitation, with the belief that adding these features would add an extra level of complexity of use that was not intended for the application. During the conceptual phase of development a goal for a clean and minimal design was set, resulting in simplicity of use.

I would like to include these controls if possible in the future but I would like to find a user friendly means of providing the controls. Hue adjustment would be more complex than the inclusion of a brightness adjustment as brightness can be linearly controlled while the hue involves; hue, saturation, colour temperature and the use of the CIE colour space (Wright 1929). Due to this complexity it is unlikely to be included within my application in the near future.

10 Further Areas of Investigation and Enhancements

During this project there are areas that given time and resources would like to have investigated or developed further.

10.1 Further Investigation

The following are areas that provided more time, would have been investigated into further.

10.1.1 Android Platform

While developing my application for the Android platform it became clear there are plenty of libraries available for use. This can be attributed to not only being based upon Java, with Java having a large code base and many libraries included and even more available by importing external libraries such as; Retrofit, that provides a type-safe REST client for Android providing mappings into a client interface using annotations, dialogplus provides the ability to display dialog items in a pleasant Lollipop styles animation, with many more available (Esquenazi 2017).

As the space of development is so large I would often conceive of a plan to resolve issues or implement functions within my application that may not have been the optimum solution. Through further experience with the platform and further time for development several solutions I implemented could have been implemented more efficiently, safer and more stable through the use of Android libraries or external libraries known to be improve development speed and code readability by addressing common Android development paradigms and styles.

10.1.2 iOS and Other Platforms

Though Android makes the majority of the current mobile platform, iOS is second taking around a 12.5% market share (*Smartphone OS Market Share, 2016 Q3* 2016) and so should be considered when developing applications. It has been found that iOS users often will spend more on and within applications making the platform potentially more lucrative over others according to details from 2013 (Louis 2013).

iOS applications can be written in several languages such as; Swift, C++, C

Objective-C and more (*Xcode IDE*). Much like the Android studio integrated development environment (IDE), iOS applications can be written within Xcode; Apple's own IDE. The benefits of using the IDEs is a tool-chain useful for developing, debugging and designing applications.

It would clearly be beneficial to be able to develop an application once and be able to deploy it across multiple platforms and systems. By leveraging systems such as *Apache Cordova* 2017 it is possible to develop an application once and have it wrapped in a native container, this is then able to be distributed on respective platforms application store.

Cordova is able to use many native APIs available on the platforms being targeted and in doing so is able to blend with the native platform and appear like a regular app. There are APIs that are unavailable through Cordova however and so there may be a need to target a platform directly. Other benefits of building natively is the ability to write code at a lower level, for instance Android applications can be written in C and C++, allowing for lower level development which can be beneficial if applications are being written for lower powered devices or further performance is to be desired from the application.

10.1.3 Functional Programming for Mobile Platforms

Being able to leverage features available within functional programming would allow for the ability to produce safe, concise and more reasonable code. The benefits of which are numerous, allowing for code to become more modular and increasing re-use and prevent potential issues when running multi-threaded functions through the inclusion of thread-safety.

Java 8 includes several features that are provided to developers to enable the use of such functional and so being able to include these within my application would be beneficial. Unfortunately Java 8 is limited to the latest version of Android and so would reduce the target market for my application. In the future however as the market moves towards the newer versions it would be possible to re-work aspects of my application to a more functional style.

10.2 Enhancements

Due to the limited timeshare to develop my application and the other projects and work undertaken throughout the year there are many aspects of my application that given the time, I would like to have implemented or improved.

10.2.1 Integration With Other Smart-devices

One key area of enhancement would be to allow for my application to interact with other smart-devices that may be available around the home, anything from heating and kettles to a smart-toaster. By integrating more devices into my application this could allow for a greater level of home automation and make the morning work around the user and reduce the amount of interaction the user would be required to make in the morning.

10.2.2 Improve the Weather functionality

The weather functionality within my app is limited to presenting the users with the weather of their specified city. Preferably the ability to obtain the users current location to pull the relevant weather would have worked better, though this would have required requesting the user to allow permissions for location tracking, something the user may not want to provide to an alarm application.

Other aspects of the weather that could use improvement would be to allow users to pick the source of their weather data as there multiple sources of weather data and each provide varying weather accuracy. This was not implemented due to API restrictions presented by many of the sources, limiting API usage for free accounts or requiring the need to pay for use. As this application is not currently intended for public use I opted for the OpenWeatherMaps (OWM) API (*OpenWeatherMap API 2017*).

The weather icon font used within my application provides a neat way of displaying scalable graphics that can easily be displayed in different colours and reduce the number of graphics imported within my application, reducing

the size of the application and clutter within the project structure during development. The icons fit well within the material design outlined by the Android design guidelines making them ideal for use within my application.

As each weather icon is presented using different character codes and these codes are stored within the strings.xml file within my application and the reference for each string includes the weather code returned by OWM, this is used to display the appropriate weather icon. Each weather API may return different weather codes however and due to this it is not possible to request the icon be displayed with a weather code provided by a different API. This could be implemented by providing an interface that can obtain the weather from various weather APIs and have a standardised weather code returned, though this would have taken quite some time as OWM returns over 90 different weather codes with more possible through the use of modifiers, for example; sunny with cloud or cloudy at night. As there are so many different weather codes to be handled I felt the time would be better spent on other aspects of my application.

10.2.3 Increase Focus of An Intelligent Morning

Initially the concept for the application came from a desire for a greater level of home automation and a more useful alarm.

Currently there is a strong focus for contextual and relevant information being presented to the user as and when they need it. An example of this is if a user had a flight booked, through the use of Google Now/Assistant Google maps could recommend the time you would need to leave to get to the airport and the best route to take, once the user is at the airport their boarding pass can be easily accessed and any delays can be pushed to the user via notifications so the information is presented to the user and doesn't require any effort on behalf of the user.

This focus towards improving the user experience by reducing the effort required by the user is of interest and I feel there is a lot of work that can be done to improve such functionality. Currently these artificial assistants require some user interaction to see the information being presented; integrating my alarm application with the user schedule however it would be possible

to present useful information after the alarm has been turned off. This information could be presented using text to speech (TTS) or by the use of a bold and brief notification being displayed on the screen.

10.2.4 Implement more smart-bulb functionality

Currently my application allows for lights to be turned off and on however I would like to increase the functionality to match that provided by the manufacturer, this could include grouping lights, changing their names for easier identification but most essential would be the ability to adjust the brightness and hue of the lighting. I was unable to implement this before the deadline as there were other key aspects of my application that required work and refinement. Currently my application supports only Philips hue lighting systems, however the current product range for the Hue system alone covers 33 products of varying functionality and handling each in a user friendly way posed a challenge on presenting only the relevant options. By providing a simple switch to turn the light on or off however accounted for all of the products and in a way that fits within the Android platform well.

10.3 Open-Source

In the future it would be beneficial to make my application open-source, by allowing others access to the source code of my application and allow for others to pull and edit my application; either to create a new fork and work on a project based upon my application or to continue developing Chrono-lux directly and improve upon the work.

The alarm application itself can be improved upon, however the key interest would be in adoption of more connected devices, allowing others to adapt the application to include their own devices not yet supported and allowing improvements to the application for everyone is possibly the most desired outcome of open-source software.

11 Evaluation

The following is a critical evaluation of every significant area my project, including my choice of project and how it fits in with the modules that I have studied over the course of my degree.

11.1 Choice of Project

Deciding upon a project that was challenging and feasible for the amount of time allocated for completion was quite a difficult task. Many ideas and concepts that I thought of were either already well investigated or posed the risk of being incomplete by the final deadline. I proposed several ideas for a final year project to Marcus Winter, my project supervisor and he helped me decide upon my home automation alarm as the project that I should undertake.

The project I decided upon however addressed less of a computer science research project and instead is more focused on user design and interaction.

My reasons for the choice of my project are due to my interest in mobile applications development and in the concept of home automation. I'm particularly interested in a future in which many common actions are handled transparently by automated systems, such as opening and closing curtains and adjusting lighting as required by the user without the need for direct user interaction and many other situations that can provide benefits by saving time each day, to assisting those who are unable to live alone and require regular assistance and house calls.

Overall I feel the project I chose has been challenging and is suited towards my degree and the modules that I have undertaken.

11.2 Course Relation

Over the course of my degree I have undertaken multiple modules many of which are relevant to my final project.

11.2.1 Mobile Application Development

Mobile app development is one module that clearly relates to my final year project having worked on a mobile application for my project. Throughout the module issues such as platform independent issues and general principles for mobile user interaction design and the constraints of developing for a mobile platform.

The issues surrounding mobile web vs native applications were also addressed, deciding on leveraging native tools by developing directly for a platform or for the development of a cross platform application.

There are also many platform specific issues from the development environment and tool-chain, to the architecture and APIs provided that need to be addressed throughout the development and maintaining of the application.

I have worked with both physical and virtual devices and have been able to see the benefits and restrictions of both, such as leveraging the speed of a physical device both in installation and application usage, over the ability to emulate numerous devices and test the application in ways that could be very costly and time consuming.

Many requirements of most applications today are the need for; persistence and storage, multimedia, interacting with hardware sensors and web services. Arguably most important of all the deployment to the relevant platforms application store was also addressed, allowing an application to go from a development build to a signed and globally installable application.

Each of the mentioned aspects of the module influenced my design and development choices.

11.2.2 Programming languages, concurrency and client server computing

As Android is based upon the Java development language, a language that has been taught and worked with throughout the course, much of what has been learned has been put into practice within my project. Many programming principles from the SOLID principles influencing my development choices to ensure the code being written wasn't fragile and could be easily extended in the future.

UI thread and concurrency are both key when developing for Android as there is a focus around ensuring only UI based activities take place within the UI thread, this is to prevent lock-ups and ensure that the user experience remains fluid and feels fast. Through the use of asynchronous tasks and other features available within Java, it is possible to run activities over multiple threads or cores, something that is becoming increasingly more important as the speed of cores remains the same or slower each year, with an increased focus on lower energy usage cores that can be activated as needed.

Java 8 provides many useful features to take advantage of aspects of functional programming, this can allow for easier implementation of concurrency within the application with the use of lambda expressions. Unfortunately the use of Java 8 is only available in the latest version of Android and as such, reduces the target market when developing. It is possible to write for both Java 8 and the same code for older versions of Android, however this would add unneeded complexity to my application.

RESTful APIs were covered within the module and proved useful for the development of my application with many APIs available today using RESTful interfaces including the Philips hue and weather API.

11.2.3 Project management

Throughout the development of my application project management has been a useful skill to have, allowing for a balance between multiple projects and the insight to set goals and milestones to be able to evaluate the project during

development.

By addressing issues outlined through missed deadlines, it is possible to reduce the impact of certain aspects of development, such as scope creep, or too much time being spent on one aspect while others go without being addressed.

11.2.4 User design and product evaluation

My project focused mainly on the user design and so this module was helpful in how to perform user evaluation and to address not only the aspects that could do with improvement but also those that worked well, providing a useful feedback loop during development, allowing me to work towards the strengths of the application design and re-work aspects that could be improved.

References

- [1] *Add Multi-Density Vector Graphics.* Google. URL: <https://developer.android.com/studio/write/vector-asset-studio.html>.
- [2] Admin. *The Agile Movement.* English. Used for information on the agile movement and scrum methodology. 2008. URL: <http://agilemethodology.org/>.
- [3] *AlarmManager.* Android. URL: <https://developer.android.com/reference/android/app/AlarmManager.html>.
- [4] Android. *Design.* English. Used for design guidlines for the UI development for the Android ecosystem. Oct. 2016. URL: <https://developer.android.com/design/index.html> (visited on 28/10/2016).
- [5] *Android Debug Bridge.* Android. URL: <https://developer.android.com/studio/command-line/adb.html>.
- [6] *Apache Cordova.* 2017. URL: <https://cordova.apache.org>.
- [7] Belkin. *Belkin Wemo.* English. Used to reference other competing home automation technologies. Belkin. URL: <https://www.wemo.com/>.
- [8] *CoordinatorLayout.* Android. URL: <https://developer.android.com/reference/android/support/design/widget/CoordinatorLayout.html>.
- [9] Justin Diaz. *Samsung Adds Blue Light Filter Setting On Galaxy Note 7.* 2016. URL: <https://www.androidheadlines.com/2016/08/samsung-adds-blue-light-filter-setting-galaxy-note-7.html>.
- [10] Nathan Esquenazi. *Must Have Libraries.* 2017. URL: https://github.com/codepath/android_guides/wiki/Must-Have-Libraries.
- [11] *Flux - software to make your life better.* URL: <https://justgetflux.com/research.html>.
- [12] *Fragments.* Android. URL: <https://developer.android.com/guide/components/fragments.html>.
- [13] *Icon Pack.* Koninklijke Philips Electronics N.V. 2016. URL: <https://developers.meethue.com/documentation/icon-pack>.

- [14] ISTQB Exam Certification. *What is Spiral model- advantages, disadvantages and when to use it?* Used to learn about the spiral methodology and contains a useful diagram that I have used within my report. ISTQBExamCertification. URL: <http://istqbexamcertification.com/what-is-spiral-model-advantages-disadvantages-and-when-to-use-it/>.
- [15] Log. Android. URL: <https://developer.android.com/reference/android/util/Log.html>.
- [16] Tristan Louis. *How Much Do Average Apps Make?* 2013. URL: <https://www.forbes.com/sites/tristanlouis/2013/08/10/how-much-do-average-apps-make/#45bdd7e746c4>.
- [17] Mike McLaughlin. *What Is Agile Methodology?* Details different forms of agile methodologies. VERSIONONE. URL: <https://www.versionone.com/agile-101/agile-methodologies/>.
- [18] Microsoft Professional Developers Conference 2005 VPR: Image Gallery. Sept. 2005. URL: <https://web.archive.org/web/20051125235511/http://www.microsoft.com/presspass/events/pdc/gallery.mspx>.
- [19] Ji Hye Oh et al. ‘Analysis of circadian properties and healthy levels of blue light from smartphones at night’. In: *Scientific reports* 5 (2015).
- [20] OpenWeatherMap API. 2017. URL: <http://openweathermap.org/current>.
- [21] Philips. *Hue*. Referencing the philips hue ligthbulbs, part of the home automation section and closely linked with my application. Philips. 2016. URL: <http://www.meethue.com>.
- [22] N.J. Salkind. *Encyclopedia of Research Design*. v. 1. SAGE Publications, 2010. ISBN: 9781412961271. URL: <https://books.google.co.uk/books?id=pvo1SauGirsC>.
- [23] *Smartphone OS Market Share, 2016 Q3*. IDC. 2016. URL: <http://www.idc.com/promo/smartphone-market-share/os>.
- [24] Snoop05. *15 seconds ADB Installer v1.4.3*. English. Dec. 2013. URL: <https://forum.xda-developers.com/showthread.php?t=2588979>.
- [25] Stanley S Stevens. ‘On the psychophysical law.’ In: *Psychological review* 64.3 (1957), p. 153.

- [26] Matt Swider. *Night Shift: How this iOS 9.3 feature helped me sleep better.* techradar. 2016. URL: <http://www.techradar.com/news/phone-and-communications/mobile-phones/night-shift-how-this-ios-9-3-feature-helped-me-sleep-better-1317815>.
- [27] W D Wright. ‘A re-determination of the trichromatic coefficients of the spectral colours’. In: *Transactions of the Optical Society* 30.4 (1929), p. 141. URL: <http://stacks.iop.org/1475-4878/30/i=4/a=301>.
- [28] *Xcode IDE*. Apple. URL: <https://developer.apple.com/xcode/features/>.
- [29] ZigBee Alliance. *What is ZigBee?* Used to reference emerging standards in IoT devices and home automation. ZigBee Alliance. 2016. URL: <http://www.zigbee.org/what-is-zigbee/>.

All links were last followed on the 9th of May, 2017

A

Application Designs

A.1 Hub Set-up

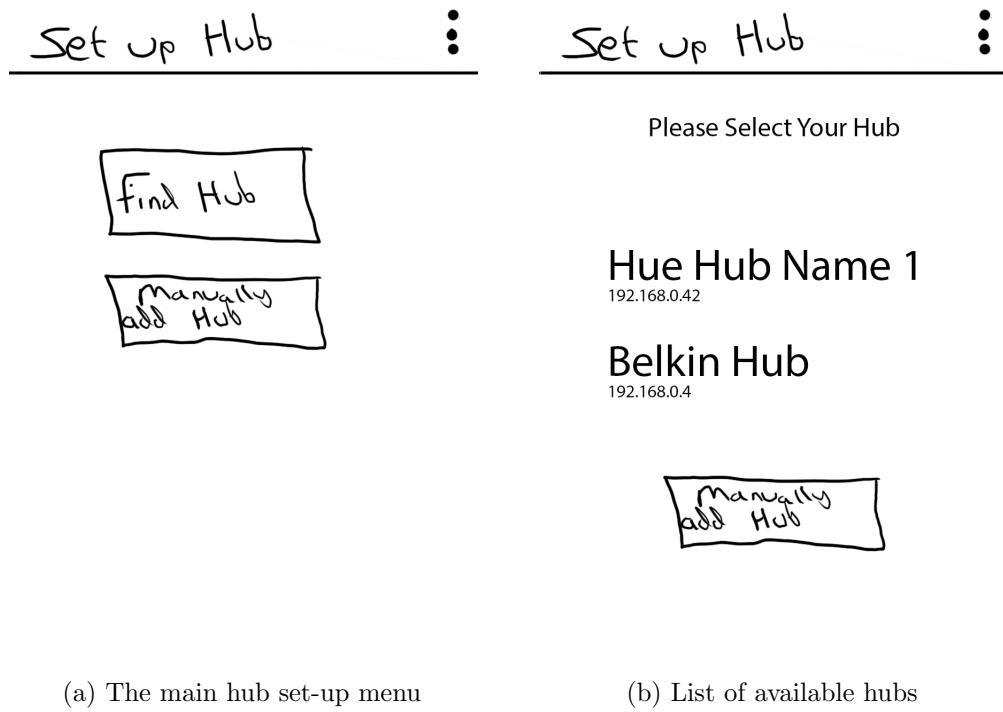


Figure 7: The views for configuring the hub.

Set up Hub

Press the touch button
to connect to hub

30 Second(s) Remaining



Figure 8: Push-link prompt

A.2 Light Tab

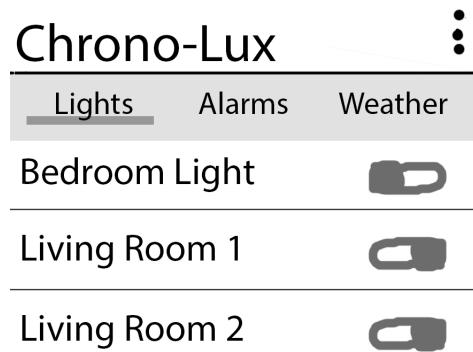


Figure 9: The design of the light tab

A.3 Alarm Tab

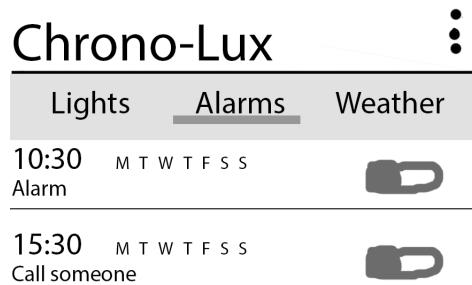


Figure 10: The design of the alarm tab

A.4 Weather Tab

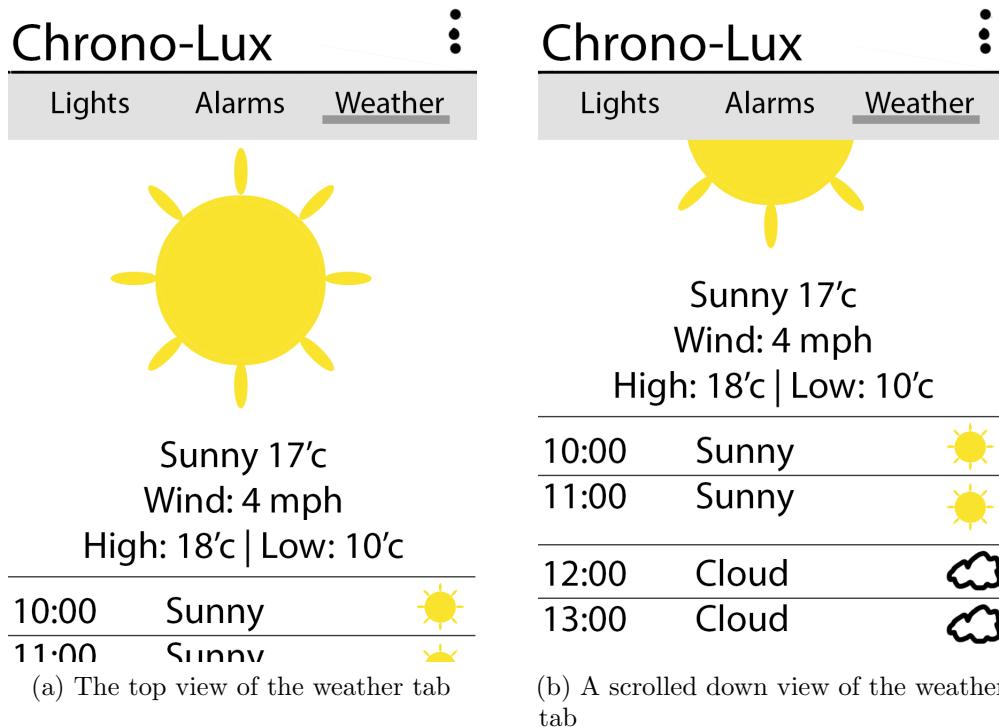


Figure 11: A view of the weather tabs.

B Application Documentation

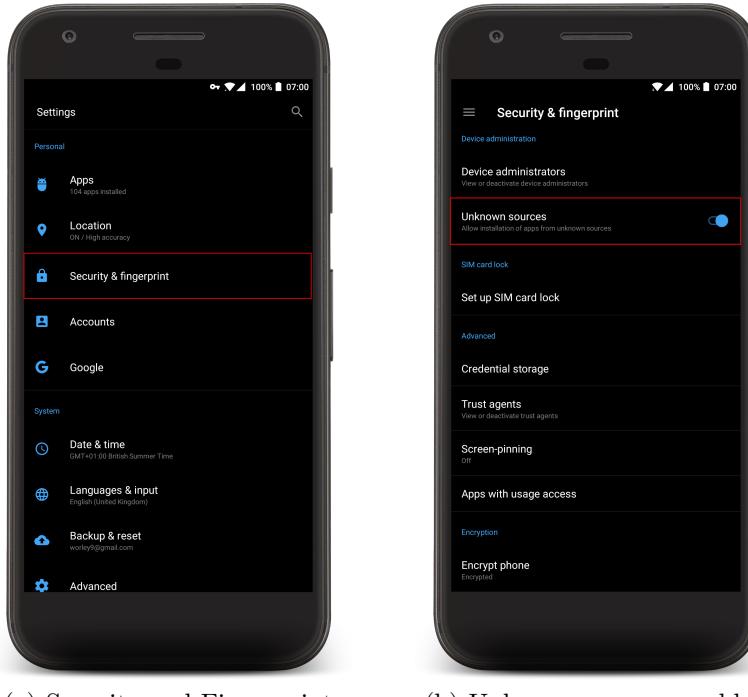
The following documentation provides instructions on how to install and use the application.

B.1 Installation

To install an Android application not available from the play store, there are several steps that are required to install the app.

B.1.1 Enable Unknown Sources

To install the application to a physical device it is necessary to enable the setting to allow for installation of apps from unknown sources; this is required as the application is not available from the play store. The setting to allow unknown sources can be found in: Settings → Security → Unknown sources:



(a) Security and Fingerprint (b) Unknown sources enabled

Figure 12: Enabling unknown sources

B.1.2 Copy APK to Device and install locally

Once installation from unknown sources has been enabled, copy the application APK to the device. On the device navigate to the location that APK was copied to within the file browser on the device. Tap on the APK and an installation prompt will be presented; accept the installation and wait for the app to be installed.

When installation has been completed the app will now be available from within the application draw or on the home-screen.

B.1.3 Alternative Installation

It is possible to install the application using the command line; in this example it shows doing so using Windows 10 and the command line.

Before being able to install the application it is not only necessary to enable unknown sources as above, but also to enable ADB debugging.

B.1.4 Enable ADB on Device (Alternative Installation)

Enabling ADB can be done by first enabling developer options, go to Settings → About phone. From here you will need to tap the *Build number* around 7 times in succession, after which a success message will show you have enabled developer settings.

Kernel version
3.4.113-cyanogenmod-g82e38c7
blinky@cyanogenmod #1
Sat Dec 24 20:03:39 PST 2016
Build date
Sat Dec 24 19:58:33 PST 2016
Build number
You are now 3 steps away from enabling cm_find7-user.debug.7.1.1.NME260.S12b2fd1a6 test-keys
SELinux status
Enforcing

(a) Enabling developer options

Kernel version
3.4.113-cyanogenmod-g82e38c7
blinky@cyanogenmod #1
Sat Dec 24 20:03:39 PST 2016
Build date
Sat Dec 24 19:58:33 PST 2016
Build number
cm_find7-user.debug.7.1.1.NME260.S12b2fd1a6 test-keys
You have enabled development settings!
SELinux status
Enforcing

(b) Developer options enabled message

Figure 13: Enabling developer options

Developer settings are now available from within the settings menu as shown in figure 14a. Enter this option and scroll to the option *Android debugging* as shown in figure 15, when toggled a warning message will appear (figure 14b); read and accept the warning to enable ADB.

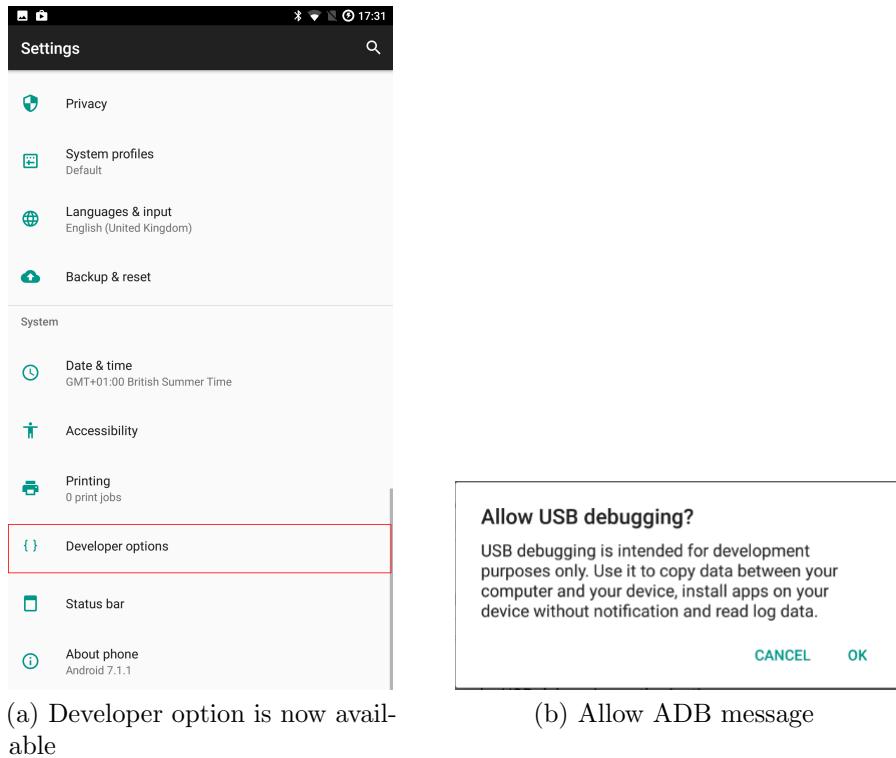


Figure 14: Enabling ADB on device

B.1.5 Install ADB - *Windows 10* (Alternative Installation)

The installation of ADB will often be included with Android studio, however to install ADB alone the easiest method would be to download and use the ADB installer tool found on XDA-Developers Snoop05 2013.

Once installed running `adb devices` will show all devices connected to the computer with ADB enabled as shown in figure 16.

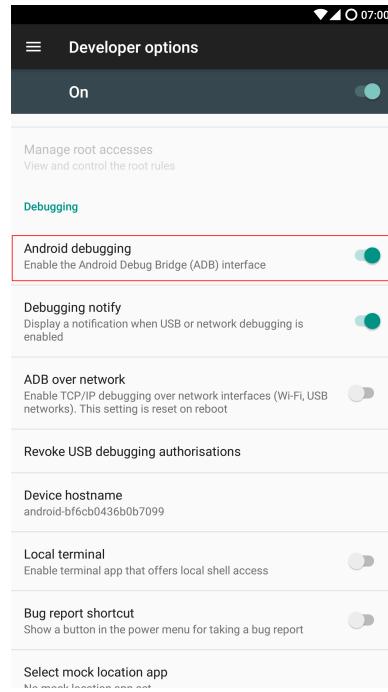


Figure 15: ADB option in developer menu

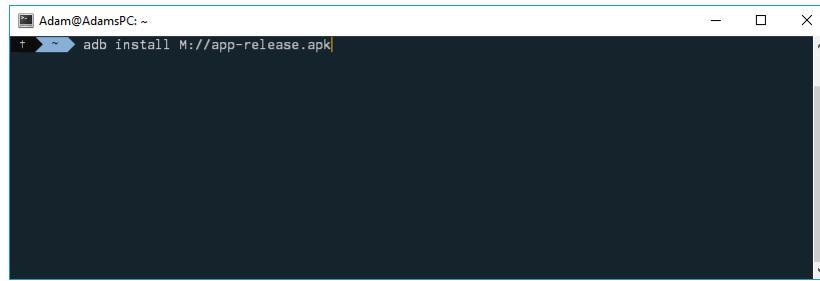
```
Adam@AdamsPC: ~
+ ➔ adb devices
List of devices attached
4a9252ee    device
```

A screenshot of a terminal window titled 'Adam@AdamsPC: ~'. The window displays the output of the 'adb devices' command, which shows a single device connected with the identifier '4a9252ee' and labeled as 'device'. The terminal has a dark background with white text.

Figure 16: Results of running ADB devices command

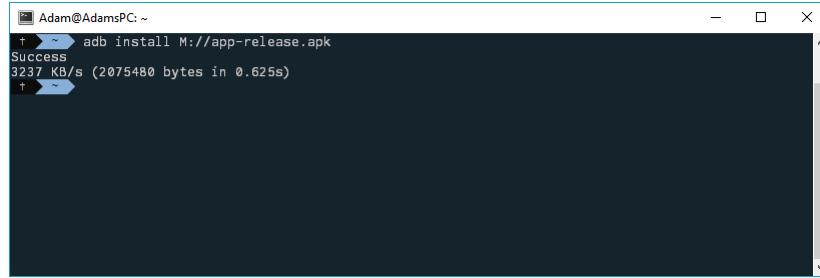
B.1.6 Installation via ADB (Alternative Installation)

Now that ADB is enabled on the device and installed on the computer it is possible to run the command `adb install *apk file location*` passing in the location to the apk file. If successful an installation success message will be displayed as shown in figure 17.



```
Adam@AdamsPC: ~
+ ➤ ~ adb install M://app-release.apk
```

(a) ADB install command



```
Adam@AdamsPC: ~
+ ➤ ~ adb install M://app-release.apk
Success
3237 KB/s (2075480 bytes in 0.625s)
+ ➤ ~
```

(b) Results of application installation success

Figure 17: Installing APK using ADB

B.2 Alarms

This section will provide instructions for the functionality of the alarm functionality.

B.2.1 Create

By clicking on the alarm create button at the bottom of the screen (as seen in figure 18) a time picker dialog will appear.

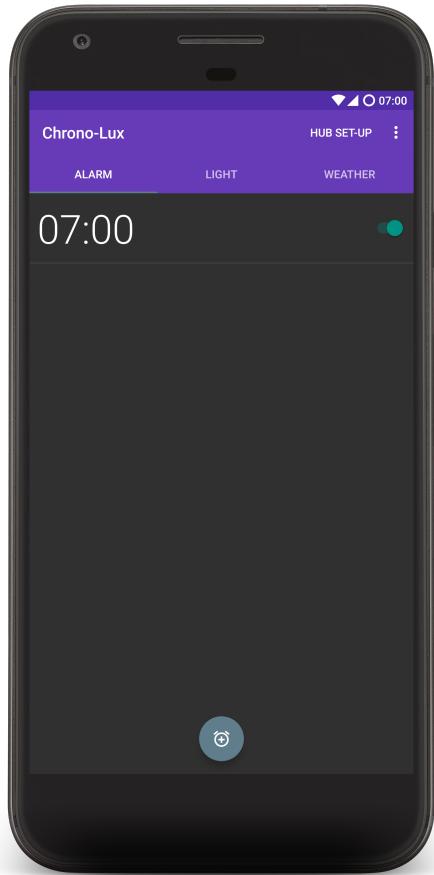


Figure 18: Create alarm button

First the hour will be selected, to select the hour desired for the alarm either tap on time or for more precision, press and drag the *hand* of the clock to the desired time. When released the picker will now allow for the selection of the minutes; repeat the same action as for the hour to select the chosen time.

B.2.2 Rename

Renaming an alarm is simple, by tapping the alarm that needs renaming an edit text pop-up will appear (as seen in figure 19); enter the new label desired and confirm the change, the alarm is now called something else.

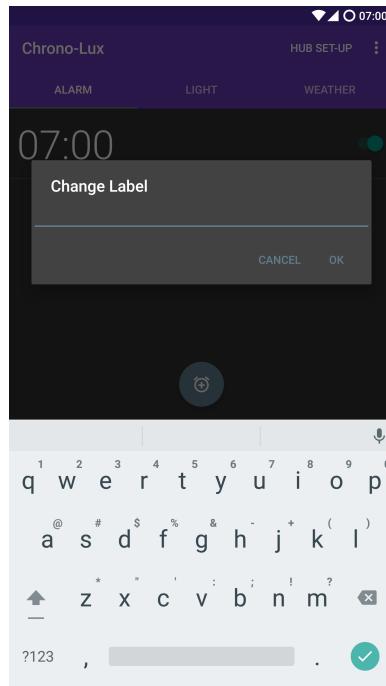


Figure 19: Change label edit text

The label provided to the alarm will appear when the alarm goes off and be displayed in the alarm notification.

B.2.3 Turning Off/On an Alarm

By pressing the switch on the right hand side of the screen, it is possible to toggle the alarm on or off.

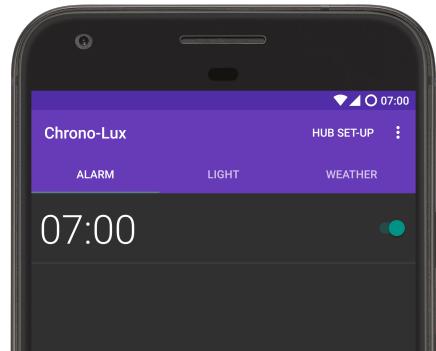


Figure 20: Toggle alarm switch

B.2.4 Delete

To delete an existing alarm, simply press and hold the alarm to be deleted, a prompt will appear asking to confirm the action.

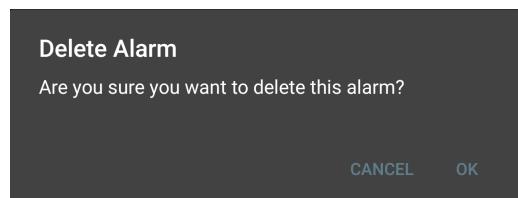


Figure 21: Delete alarm dialog box

B.3 Lights

It is required to connect the application to a lighting bridge to be able to enable the lighting functionality within the application. The alarms and weather will work without the lighting being configured.

B.3.1 Connecting to the Bridge

To be able to use the smart-light functionality it is necessary to pair the application with the bridge prior to use, this allows the application to be *white-listed* and provide access to the lighting interface.

When navigating to the lighting tab, if no bridge has been configured a text prompt will appear and at the press of the button will take you to the set-up activity. If the activity does not automatically begin searching for a bridge, pressing the search for bridge button will begin a search.

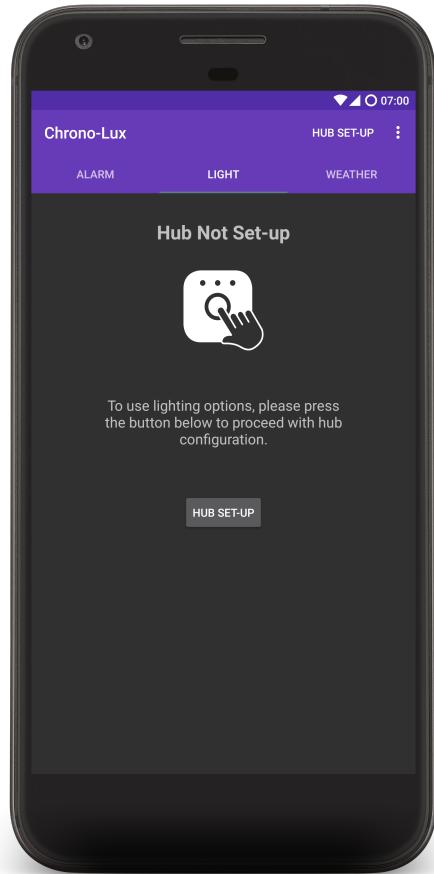


Figure 22: Bridge set-up prompt screen

If no bridges are returned please ensure you are on the same local network as the bridge and there are no firewalls, VPNs or other potential network configuration that could be blocking communications. If no bridges are found or you would like to search again, simply press the search button again.

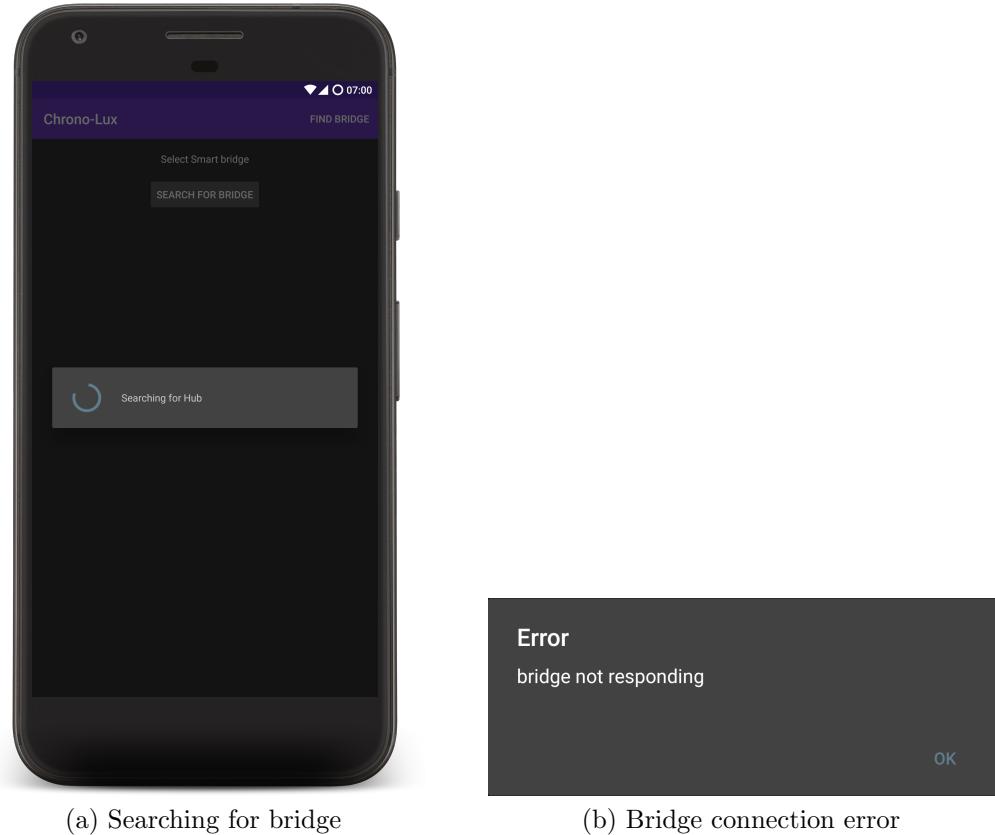
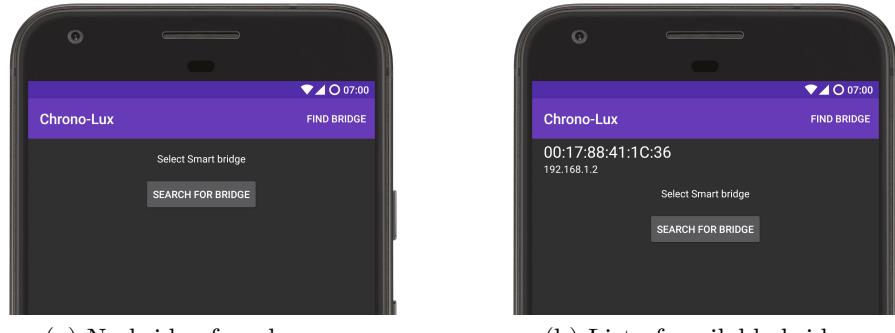


Figure 23: Finding available bridges

When the search is completed a list of all found bridges will be displayed, to select one tap on it to begin the authentication process.



(a) No bridge found screen

(b) List of available bridges

Figure 24: Finding available bridges

On selecting a bridge the push-button authentication will occur, please press the large push-link button on the front/top of the bridge. This action is to ensure that there is physical access to the bridge as a security measure to prevent un-authorised applications or intruders on the network gaining access to the lighting interface.



Figure 25: The push-link prompt and countdown

On successful authentication you shall be returned back to the main application, now when navigating to the lighting tab any connected lights shall be displayed similar to that shown in figure 26.

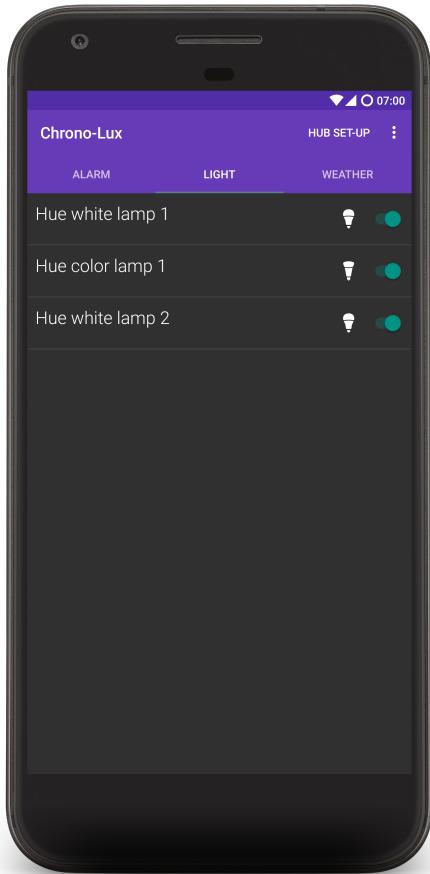


Figure 26: List of available lights

B.3.2 Toggling the lights

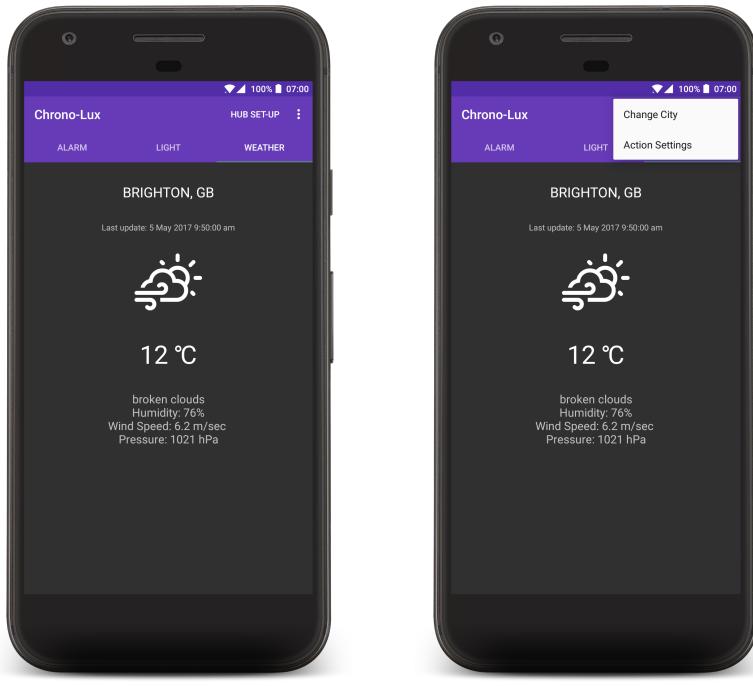
Any lights assigned to the bridge will be displayed in a list, displaying their name and an icon indicating the light type.

Each alarm can be turned on/off easily by tapping the switch on the right hand side of the screen.

It should be noted that if a light is turned off at the wall/switch the last known light state will be displayed within the application when displayed.

B.4 Weather

Weather will be displayed on the weather tab. To change the location of the weather being displayed, press on the menu expansion button from anywhere within the application as seen in figure 27, this will display the option to change the city stored.



(a) The options menu is in the top right
(b) Change city option

Figure 27: The city selection option

when the change city option is selected a text dialog will appear allowing for the entry of a different city, simply type in the name of the city desired and accept the change, the weather will be updated when the weather tab is loaded next.

C

Activity Log

The following is a log/overview of the work done each week.

C.1 October

First Week

- Researching into competitors and working on the interim report

Second Week

- Working on the interim report

Third - Fourth weeks

- A majority of time spent working on other projects/modules.

C.2 November

There was a lot of time spent working on other modules due to deadlines.

First week

- Working on the interim report and other modules
- 03/11/2016 - Exceeded word count and so need to reduce it to below 2000

- Reading into the Philips Hue API and trying out the app. The API seems fairly straight forward, though authenticating with the hub could be a challenge.
- There is no need to handle the hub connection through JSON, there is instead a Philips hue library that can be imported for use. It provides extra functionality to handle all API calls, though I think I will opt for directly using the JSON API.

Second Week

- Worked only on other modules
- Met with Marcus on the 9th

Third Week

- Worked only on other modules
- Ordered a smart-bulb

Fourth Week

- Worked only on other modules

C.3 December

Spent a lot of time working on other projects and being engaged in family/Xmas events, less work completed than I had hoped.

First week

- Working on app, struggled with creating the designs and views. Generally getting to grips with the Android SDK

- Met with Marcus on the 1st

Second Week

- working on other projects and on app

Third Week

- No work done

Fourth Week

- Little work done towards other modules

C.4 January

First week

- Working on finishing another piece of work due to deadline.

Second Week

- Worked on application

Third Week

- Worked on application, a lot of the work towards the

Fourth Week

- 02/01/2017 - Attempting to develop the layouts for my app. I'm finding a lot of the Android library to be deprecated such as snackbars replacing the toast messages; to handle these and correctly show them as per the design guidelines is not as straight forward.
- 03/01/2017 - Using the container layout with a nested linear/relative layout is needed to handle the Snackbar and it is now rendering correctly. More issues have occurred as I would like a tabbed toolbar for navigating through the different aspects of the app such as alarm/light control/weather. The tabs require fragments and a pageviewer to handle displaying the correct layout and I'm having issues with the app closing on start. I have also made an icon for my app to replace the default green android.
- 04/01/2017 - I am still facing issues with the TabLayout as it doesn't appear to render correctly and trying to access certain fragments closes the app. I'm finding a lot of resources are quite outdated and many show incorrect or incomplete implementations.
- 05/01/2017 - The tab pages are now working correctly. I have a spinner that can display multiple String elements, I am now trying to display alarm objects in the list instead but am having difficulties.

C.5 February

First week

- Worked on other projects

Second Week

- Worked on other projects

Third Week

- Lots of development on the app happened:
- Re-done the tab layout using fragments to avoid using deprecated code.

Fourth Week

- Worked on other projects

C.6 March

This was a heavy month for developing my application

First week

- Worked on getting alarms working
- Met with Marcus on the 3rd

Second Week

- Still working on alarms, using pending intents and broadcast receiver is more complicated than expected.

Third Week

- Include lights, had a bit of trouble getting the bridge connected properly

Fourth Week

- Bridge not connecting/showing, there was an error in the logic detecting if bridge had been connected.
- Improving lights and alarms.
- Got weather working

C.7 April

This month involved a lot of report writing

First week

- Every day of the week spent writing the report

Second Week

- Every day of the week spent writing the report

Third Week

- Working on reports for other modules.

Fourth Week

- Less work was done this week, worked on developing app for a different module and writing the report for that work instead.

C.8 May

First week

- Writing more for the report
- Wrote application documentation

Second Week

- Proof-read, bound and submitted work.