Smart Revision Calendar and Timetable Android Application

Author
Russell Waterson

Student ID
1330057

Supervisor
Uday Reddy

Degree Course
B.Sc. in Computer Science

Institution
School of Computer Science
University of Birmingham

Date
13th April 2017

1 Table of Contents

2 ACKNOWLEDGMENTS

I would firstly like to thank my supervisor Uday Reddy for supervising my project, and secondly a big thank you to Russell Beale, the project coordinator, for answering and dealing with any queries and problems I faced.

I wish to extend this thanks to my assessment team, Hayo Thielecke and Alan Sexton, as well as all those who took part in my testing and questionnaires, giving me vital feedback and enabling me to improve project.

Also, thanks must go to StepStone Tech and Alam Kanak, the authors of Third Party Libraries used in this project.

3 ABSTRACT

The proposed project is to create a smart revision timetable and calendar Android application. This mobile application is targeted towards students and will allow them to create a work and revision timetable with minimal effort. The premise is, the user enters various inputs, including their class times, extracurricular activities, exams and deadlines, and their preferred style of learning and working. The application will then generate a revision timetable around the classes and activities with the exams as goals to work towards based on their given priority. In addition, the user also has the opportunity to provide feedback on how productive they have been at the end of each day of revision. This feedback, coupled with the user's learning style, means that the more the user uses the system, the more the timetable will learn and be tailored to that individual user.

All code for this project can be found at this **GIT repository**:
https://git-teaching.cs.bham.ac.uk/mod-40cr-proj-2016/rjw357

Keywords: Android Development, Academic Aid, Timetable Generation, Probabilistic Algorithm

4 Introduction

There are various studies that have been conducted proving that using a timetable and having self-imposed deadlines can have a positive impact on self-control and task performance. The aim of the work described in this report was to provide a software tool, in the form of an Android mobile application, in which to assist people, in particular students, with the creation of revision and work timetables in order to increase productivity.

There are a number of other factors that can affect one's peak performance, these include time of day, and length of working hours with consistency of breaks. Further study into various learning methods and techniques, circadian rhythm, and mental stamina, all demonstrate the effects of productivity and performance in work. The proposed system aims to provide additional benefits in the form of smart features, whereby, for example, the system generates revision slots for the user based on what should be prioritised, along with their preferred learning style.

The inspiration of this project comes from the desire to maximise productivity within exam seasons, where timetable creation is a beneficial however time costly process. At the time of writing, there are currently no intelligent revision timetables in existence, and therefore the proposed project aims to add significant value to this field.

A user with say three exams in quick succession one after another, may be struggling to decide what to revise for and when. The system aims to evaluate the work required for each exam, along with the user's style of learning, and determine the perfect revision schedule to maximise productivity for that user, whilst avoiding clashes with their other activities.

The system achieves the proposed goals by using a two-part probabilistic algorithm, taking data from a number of tables within a locally stored SQL database, one of which containing constantly evolving data.

The system adopts a user interface which is developed to meet Ben Shneiderman's "Eight Golden Rules of Interface Design" and Jakob Nielsen's "Usability Heuristics for User Interface Design", ensuring a productive and frustration-free interface. This was attained aided by the use of Google's Material Design Guidelines, an established visual design language created by Google.

The system underwent rigorous and extensive testing and evaluation to ensure that the developed solution was of the highest quality, adopting multiple techniques including black box and white box testing methods, Android specific testing tools, and user investigative evaluation.

The project compromised of four main phases; research, implementation, testing, and evaluation, each of which can be broken down further and are outlined in finer detail within this report.

5 Background Research

5.1 Fixed Scheduling and Learning Styles

A study undertaken by Dan Ariely and Klaus Wertenbroch, examines the effects of self-control by precommitment, and looks to answer a range of questions including whether self-imposed deadlines are effective in improving task performance. The study requires students at MIT to undertake three short papers, half of whom had fixed deadlines at regular intervals, and the other half of whom had one deadline for all three at the end of term, and were asked to work under their own volition and set their own deadlines. It was concluded that in addition to the students with deadlines achieving higher grades, they also attained secondary benefits in other aspects of performance that required the investment of their time as a resource (Ariely and Wertenbroch, 2002).

Cal Newport, a computer science professor at Georgetown University, uses the findings outlined in the above study to create his own method of working that allows for completion of a large amount of work in a small number of work hours; he calls it Fixed Schedule Productivity. In essence, it involves choosing a schedule of work hours that provides an ideal balance of effort and relaxation, and then doing whatever it takes to avoid violating that schedule (Newport, 2008). The aim is that once there is a specific goal that can be focused on, the strategies become easier to deploy (Newport, 2008).

It is evident that the use of timetables is greatly beneficial, however even with a fixed schedule, every individual has their own preferred style of learning. One such method that has been taught to individuals and teams since 1998, is called the Pomodoro Technique. The Pomodoro Technique, developed by Francesco Cirillo in 1992, is a time management technique used to break up working into smaller intervals with frequent, but short, breaks. Usually this would consist of 25 minutes of solid uninterrupted work, followed by a 5 minute break, which is then repeated four times, until a longer break can be had (Cirillo, 2007). It aims to improve productivity by boosting motivation and keeping it constant.

In addition to work and break lengths, individuals may find that the time of day in which work takes place, to have a varying impact on their productivity. A study by Nolan G. Pope, looks to identify how the time of day affects productivity. He deduces that there are definitive time-of-day differences in productivity, where simply rearranging when tasks are performed can allow for greater efficiency gains. Using students as test data, he concludes that despite adolescents having a roughly two hour later circadian rhythm that adults, productivity in students on average is higher in the mornings than the afternoon; believed to mainly be due to the structure of the working day, as well as stamina, the lack thereof resulting in physical fatigue, mental fatigue, and drowsiness (Pope, 2016).

5.2 Current Solutions

Currently, an application does not exist that tackles the exact problem proposed, being a smart, self-generating revision timetable, catering to the needs of an individual's learning style. There does however exist alternative implementations of standard timetables, without the use of smart features.

The most common approach, is to create a handwritten timetable, whereby the user simply writes out by hand what they want to achieve and by when. The advantage of this, is the unlimited freedom, in regards to the design and layout that the user has; conversely, this approach comes with many disadvantages. Plans are constantly changing, especially concerning revision, and the difficulty that comes with the altering and editing of a handwritten approach can be crippling. Further to this, the lack of ubiquitous availability means that the created timetable is not always to hand, meaning that a change of location can lead to not knowing what should be revised when. And finally, hand written timetables are extremely time costly in terms of their creation, time being a precious commodity in the exam season.

Moving to an electronic approach, there are many software packages available to create timetables with, all without any smart features however, for example the commonly used Microsoft Excel. While this program is easy to use and intuitive, it too comes with the burden of a time costly creation of any meaningful timetable.

The proposed implementation aims to alleviate all above drawbacks, thus hopefully encouraging more people to adopt this to aid their revision.

6 Analysis and Specification

6.1 Survey

Once the proposal and overarching theme of the project had been established, the core user requirements had to be identified and documented. During this inception of requirements fabrication, there were a significant number of different directions and priorities in which the project could take in terms of feature set. In order to get a true reflection of what potential users would want from an application such as the one proposed, a data gathering technique had to be adopted. Upon reviewing advantages and disadvantages of a vast range of techniques, including interviews, focus groups, observation, and cultural probes, the decision was made to create and distribute a survey.

A survey is a practical requirements gathering tool that allows for the distinguishing of the attitudes, values, and opinions of the participants, as well as their likes and dislikes. The biggest advantage with using surveys and questionnaires, is that with minimal effort, data can be collected from a large number of participants. Once the set of questions have been created, they can be distributed in a wide variety of ways, including email, via a web-based provider, or simply by paper. The individual data that is retrieved, is then easily compared with other participant's data as it, and the questions, are structured. Further to this, surveys can incorporate a mixture of both quantities and qualitative data. The former being via the use of closed questions, for example check boxes, ranges, and Likert scales. And the latter in the form of open questions.

The survey for this project was created and distributed using the online survey creation software, SurveyMonkey, of which can be viewed via the link provided in the project zip file, as explained in Appendix A. Once the survey had been in circulation of a suitable period of time, the results were collected and with the omission of qualitative data, are as followed.

Q1) What stage of study are you currently at?				
Postgraduate	Undergraduate	Sixth Form / College (A Levels)	Secondary School (GCSEs)	Other (Please specify)
6 – 12.5%	38 – 79.17%	0 – 0%	0 – 0%	4 – 8.33%

Q2) When revising for exams, do you create a revision timetable for yourself?		
Yes	No (Proceed to question 5)	Sometimes
18 – 37.5%	18 – 37.5%	12 – 25%

Q3) Do you feel that having a revision timetable helps you to be more productive when revising?		
Yes	No	Sometimes
26 – 86.67%	2 – 6.67%	2 – 6.67%

Q4) How beneficial would it be to you, in having a computer program or mobile app to automatically and intelligently generate a timetable for you instead of creating your own? (Upon completion, proceed to Q7)				
Very Beneficial	Mildly Beneficial	No Difference	Mild Hindrance	Large Hindrance
12 – 40%	14 – 46.67%	2 – 6.67%	0 – 0%	2 – 6.67%

Q5) Do you feel that having a revision timetable would help you to be more productive when revising?		
Yes	No	Unsure
15 – 83.33%	1 – 5.56%	2 – 11.11%

Q6) How beneficial would it be to you, in having a computer program or mobile app to automatically and intelligently generate a timetable for you?				
Very Beneficial	Mildly Beneficial	No Difference	Mild Hindrance	Large Hindrance
7 – 38.89%	9 – 50%	0 – 0%	0 – 0%	2 – 11.11%

Q7.1) In a timetable or calendar, how important to you are the following features? – Being able to view the timetable on multiple devices			
Very Important	Mildly Important	Mildly Unimportant	Very Unimportant
18 – 42.86%	20 – 47.62%	4 – 9.52%	0 – 0%
Q7.2) In a timetable or calendar, how important to you are the following features? – Having a printable format available			
Very Important	Mildly Important	Mildly Unimportant	Very Unimportant
20 – 47.62%	12 – 28.57%	6 – 14.29%	4 – 9.52%

Q8.1) How much do you feel these extra features can help you to increase productivity of revision? – Gamification and mini incentives				
High increase in productivity	Slight increase in productivity	No change	Slight decrease in productivity	High decrease in productivity
16 – 38.10%	20 – 47.62%	4 – 9.52%	2 – 4.76%	0 – 0%
Q8.2) How much do you feel these extra features can help you to increase productivity of revision? – Social Aspects				
High increase in productivity	Slight increase in productivity	No change	Slight decrease in productivity	High decrease in productivity
4 – 9.52%	18 – 42.86%	6 – 28.57%	4 – 19.05%	0 – 0%
Q8.3) How much do you feel these extra features can help you to increase productivity of revision? – Muting phone during revision periods				
High increase in productivity	Slight increase in productivity	No change	Slight decrease in productivity	High decrease in productivity
24 – 57.14%	10 – 23.81%	8 – 19.05%	0 – 0%	0 – 0%

6.2 Use Cases

Using the response and feedback from the survey outlined in section 6.1, a series of use cases was developed. A use case is a means of capturing requirements of a system by specifying how a user of the system, the Actor, interacts with a detailed instance and its emergent behaviour (Anonymous 2015).

A use case is written as natural language statements that are understandable by non-developers, for example any business-related people involved in the project. Along with the normal case step, an alternative case and an exception case can be specified. The alternative case is when the user may be granted with a choice of possible actions (Robertson and Robertson 2006). The exception case is an undesirable but acceptable variation from the normal case (Robertson and Robertson 2006).

Case	Actor	Basic Flow
1	User	The user opens application for the first time; the system displays initial start-up process welcoming the user and explaining the major features of the system. The user navigates through each step entering data accordingly. **Exception case 1.1:** The user enters invalid data into the learning style, so the system alerts the user with a validation error.
2	User	The user selects the classes screen; a list of all classes is displayed. The user selects to add a new class; the add a new class screen is displayed. The user enters class data and presses save; the class is added to the system. The user is back on the classes screen; the new class is displayed in the list of classes. **Alternative case 2.1:** The user enters class data but presses back; the class is not added to the system. The user is back on the classes screen; the new class is not displayed in the list of classes. **Exception case 2.1:** The user enters invalid class data and pressed save; the system displays an error message informing the user of the invalid data.

3	User	The user selects the classes screen; a list of all classes is displayed. The user selects a displayed class; a screen is displayed showing all the information about the class. The user selects the edit icon; the edit class screen is displayed. The user enters class data and presses edit; the class fields are updated. The user is back on the classes screen; the class is displayed with the updated data. **Alternative case 3.1:** The user enters class data but presses back; the class is not edited. The user is back on the classes screen; the class's data remains the same. **Exception case 3.1:** The user enters invalid class data and pressed edit; the system displays an error message informing the user of the invalid data.
4	User	The user selects the classes screen; a list of all classes is displayed. The user selects a displayed class; a screen is displayed showing all the information about the class. The user selects the delete icon; a message is displayed asking the user if they are sure. The user presses yes; the class is removed from the system. The user is back on the classes screen; the class is no longer present in the displayed list. **Alternative case 4.1:** The user presses no; the class is not deleted. The user is back on the classes screen; the class is still present in the displayed list.
5	User	The user selects the activities screen; a list of all activities is displayed. The user selects to add a new activity; the add a new activity screen is displayed. The user enters activity data and presses save; the activity is added to the system. The user is back on the activities screen; the new activity is displayed in the list of activities. **Alternative case 5.1:** The user enters activity data but presses back; the activity is not added to the system. The user is back on the activities screen; the new activity is not displayed in the list of activities. **Exception case 5.1:** The user enters invalid activity data and pressed save; the system displays an error message informing the user of the invalid data.
6	User	The user selects the activities screen; a list of all activities is displayed. The user selects a displayed activity; a screen is displayed showing all the information about the activity. The user selects the edit icon; the edit activity screen is displayed. The user enters activity data and presses edit; the activity fields are updated. The user is back on the activities screen; the activity is displayed with the updated data. **Alternative case 6.1:** The user enters activity data but presses back; the activity is not edited. The user is back on the activities screen; the activity's data remains the same. **Exception case 6.1:** The user enters invalid activity data and pressed edit; the system displays an error message informing the user of the invalid data.
7	User	The user selects the activities screen; a list of all activities is displayed. The user selects a displayed activity; a screen is displayed showing all the information about the activity. The user selects the delete icon; a message is displayed asking the user if they are sure. The user presses yes; the activity is removed from the system. The user is back on the activities screen; the activity is no longer present in the displayed list. **Alternative case 7.1:** The user presses no; the activity is not deleted. The user is back on the activities screen; the activity is still present in the displayed list.
8	User	The user selects the exams screen; a list of all exams is displayed. The user selects to add a new exam; the add a new exam screen is displayed. The user enters exam data and presses save; the exam is added to the system. The user is back on the exams screen; the new exam is displayed in the list of exams. **Alternative case 8.1:** The user enters exam data but presses back; the exam is not added to the system. The user is back on the exams screen; the new exam is not displayed in the list of exams. **Exception case 8.1:** The user enters invalid exam data and pressed save; the system displays an error message informing the user of the invalid data.

9	User	The user selects the exams screen; a list of all exams is displayed. The user selects a displayed exam; a screen is displayed showing all the information about the exam. The user selects the edit icon; the edit exam screen is displayed. The user enters exam data and presses edit; the exam fields are updated. The user is back on the exams screen; the exam is displayed with the updated data. **Alternative case 9.1:** The user enters exam data but presses back; the exam is not edited. The user is back on the exams screen; the exam's data remains the same. **Exception case 9.1:** The user enters invalid exam data and pressed edit; the system displays an error message informing the user of the invalid data.
10	User	The user selects the exams screen; a list of all exams is displayed. The user selects a displayed exam; a screen is displayed showing all the information about the exam. The user selects the delete icon; a message is displayed asking the user if they are sure. The user presses yes; the exam is removed from the system. The user is back on the exams screen; the exam is no longer present in the displayed list. **Alternative case 10.1:** The user presses no; the exam is not deleted. The user is back on the exams screen; the exam is still present in the displayed list.
11	User	The user selects the timetable screen; the previously viewed timetable view is displayed. The user selects the day view; the day timetable view is displayed. The user swipes left and right; the current day is changed backwards and forwards respectively. The user swipes up and down; the current time is changed earlier and later respectively. **Alternative case 11.1:** The user selects the week view; the week timetable view is displayed. The user swipes left and right; the current week is changed backwards and forwards respectively. The user swipes up and down; the current time is changed earlier and later respectively. **Alternative case 11.2:** The user select the month view; the month timetable view is displayed. The user swipes left and right; the current month is changed backwards and forwards respectively.
12	User	The user selects the smart revision settings; the smart revision settings screen is displayed. The user activates Smart Calendar features and selects to generate revision; the system generates revision based on the learning style and other data, and then gives the user the option to keep the old revision, keep the new revision, or view the difference between them. The user clicks to view the difference; a screen is displayed with a side-by-side comparison of the old and new revision. The user clicks to keep the new revision; the system deletes the old revision. The user returns to the timetable screen; the timetable is displayed populated with all the newly generated revision. **Alternative case 12.1:** The user clicks to keep the old revision; the system deletes the new revision. The user returns to the timetable screen; the timetable is displayed populated with all the previously existing revision. **Alternative case 12.2:** The user activates Smart Calendar features and selects to generate revision for the first time; the system generates revision based on the learning style and other data. The user returns to the timetable screen; the timetable is displayed populated with all the newly generated revision. **Exception case 12.1:** The user selects to generate revision; the system displays an error message warning the user that the smart revision features are not activated. **Exception case 12.2:** The user selects to generate revision; the system displays an error message warning the user that no exams have been entered into the system.
13	User	The user selects on a revision block from a timetable screen; a screen is displayed showing all the information about the revision block. The user selects the edit icon; the edit revision screen is displayed. The user enters revision data and presses edit; the

		revision fields are updated. The user is back on the timetables screen; the revision is displayed with the updated data. **Alternative case 13.1:** The user enters revision data but presses back; the revision block is not edited. The user is back on the timetable screen; the revision's data remains the same. **Exception case 13.1:** The user enters invalid revision data and pressed edit; the system displays an error message informing the user of the invalid data.
14	User	The user selects on a revision block from a timetable screen; a screen is displayed showing all the information about the revision block. The user selects the delete icon; a message is displayed asking the user if they are sure. The user presses yes; the revision block is removed from the system. The user is back on the timetable screen; the revision block is no longer present. **Alternative case 14.1:** The user presses no; the revision block is not deleted. The user is back on the timetable screen; the revision block is still present.
15	User	The user selects the backup option; the backup screen is displayed. The user selects the account they wish to backup to; the system establishes a connection to the selected Google Drive account. The user selects to backup the generated text file; the generated text file is saved to the account. The user selects to backup the database file; the database file is saved to the account. **Alternative case 15.1:** The user selects not to backup the generated text file; the generated text file is not saved to the account. **Alternative case 15.2:** The user selects not to backup the database file; the database file is not saved to the account. **Exception case 15.1:** The user selects the account they wish to backup to; the system cannot establish a connection to the selected Google Drive account, so an error message is displayed giving the reason.
16	User	The user selects the import Google Calendar Events option; the import events screen is displayed. The user selects the account they wish to import from; the system establishes a connection to the selected Google Calendar account. The user enters how many events they wish to search for; the system searches for the entered number of events. The user selects which events to import and presses save; the system imports selected events and adds them to the system. The user is back on the timetables screen; the newly imported events are shown in the timetable. **Exception case 16.1:** The user selects the account they wish to import from; the system cannot establish a connection to the selected Google Calendar account, so an error message is displayed giving the reason. **Exception case 16.2:** The user enters an invalid number for how many events they wish to search for; the system displays an error informing the user the correct range in which to search for events.
17	User	The user selects on an imported event from a timetable screen; a screen is displayed showing all the information about the event. The user selects the delete icon; a message is displayed asking the user if they are sure. The user presses yes; the event is removed from the system. The user is back on the timetable screen; the event is no longer present. **Alternative case 17.1:** The user presses no; the event is not deleted. The user is back on the timetable screen; the event is still present.
18	User	The user selects the settings screen; the settings screen is displayed. The user activates the phone muting during revision hours; the system activates the triggers to turn on and turn off the muting of the phone. **Alternative case 18.1:** The user deactivates the phone muting during revision hours; the system deactivates the triggers to turn on and turn off the muting of the phone.

6.3 Functional Requirements

From the collection of use cases outlined in section 6.2, a set of functional requirements could be established. Functional requirements are a type of specification, independent of any technology used by the product, explaining what the system must do, and the procedures to be carried out (Robertson and Robertson 2006). The functional requirements are derived from the use cases by asking what the system must do in order to complete each step. Below is an extensive list of functional requirements for the proposed system.

Intro screen: so that the user can be introduced to the application

- The product should display information about major features of the app on first time boot
- The product should display instructions on how to operate the app on first time boot

Class: so that the user can manipulate data about their classes and lectures

- The product should allow the user to add a new class into the system
- The product should prevent the user from entering a new class with invalid data
- The product should allow the user to view all the classes in the system
- The product should allow the user to edit previously entered classes in the system
- The product should prevent the user from editing an existing class to have invalid data
- The product should allow the user to delete an existing class from the system

Activity: so that the user can manipulate data about their extracurricular activities

- The product should allow the user to add a new activity into the system
- The product should prevent the user from entering a new activity with invalid data
- The product should allow the user to view all the activities in the system
- The product should allow the user to edit previously entered activities in the system
- The product should prevent the user from editing an existing activity to have invalid data
- The product should allow the user to delete an existing activity from the system

Exam: so that the user can manipulate data about their exams

- The product should allow the user to add a new exam into the system
- The product should prevent the user from entering a new exam with invalid data
- The product should allow the user to view all the exams in the system
- The product should allow the user to edit previously entered exams in the system
- The product should prevent the user from editing an existing exam to have invalid data
- The product should allow the user to delete an existing exam from the system

Timetable: so that the user can view all the events in the system in a presentable manner

- The product should display a day view timetable containing all the events in the system with easy navigation between days
- The product should display a week view timetable containing all the events in the system with easy navigation between weeks
- The product should display a month view timetable

Smart Revision and Data: so that the user can maintain their learning style and generate revision

- The product should allow the user to manually update their smart data and learning style
- The product should prevent the user from entering invalid learning data
- The product should allow the user to provide feedback per revision block via a debrief
- The product should allow the user to provide feedback per day in the form of a debrief
- The product should recommend updates to the learning style data based on the debriefs
- The product should generate revision slots based on learning style and all other inputs
- The product should allow the user to view the differences in data with newly generated revision against existing revision that may be overwritten

Revision: so that the user can manipulate data about the generated revision

- The product should allow the user to view all the revision blocks in the system
- The product should allow the user to edit generated revision blocks
- The product should prevent the user from editing a revision block to have invalid data
- The product should allow the user to delete an existing revision block from the system

Backup: so that the user can backup their data

- The product should allow the user to connect the application to their Google Drive account
- The product should generate a text formatted version of the database to be backed-up
- The product should allow the user to backup the raw database file

Import Events: so that the user can import events from existing calendars

- The product should allow the user to connect the application to their Google Calendar account
- The product should allow the user to import chosen events into the system
- The product should allow the user to view all the events in the system
- The product should allow the user to delete an imported event

Phone muting: so that distractions are minimised during revision

- The product should automatically mute the phone during working hours
- The product should automatically unmute the phone outside of working hours

6.4 Non-Functional Requirements

Non-functional requirements are the set of qualities the system has as well as how well it operates, overseeing that the system performs in a particular way and to a high standard (Robertson and Robertson 2006). These are outlined by set of criteria for which requirements are defined.

Look and Feel: The look and feel is not a detailed design of the interface, but instead looks to identify objects within the appearance (Robertson and Robertson 2006).

- The product should comply with Google's Material Design Guidelines
- The product should be consistent across a large number of display resolutions

Usability and Humanity: The usability requirements look to ensure the system conforms to varying user's levels of ability, expectations, and skill (Robertson and Robertson 2006).

- The product should be easy to use for students with varying levels of application interaction experience
- The product should be easy to learn by a user on their first attempt
- The product should provide the preferred way of timetable creation

Performance Requirements: The performance requirements ensure that the system performs tasks within a specific amount of time or to a given level of accuracy. It also covers capacity concerns, as well as risk of damage to people or property (Robertson and Robertson 2006).

- The product should navigate between screens instantaneously
- The product should generate a set of new revision blocks within 5 seconds
- The product should have a capacity of at least 10,000 items across all tables in the database
- The product should not generate revision where the user's sleep cycle is severely affected

Operational and Environmental: The operational requirements outline the environment in which the system will be used (Robertson and Robertson 2006).

- The product should be used in classroom settings, at home, and at times of potential stress
- The product should be available to use, regardless of an active internet connection

Maintainability and Support: How or what type of maintenance is not always known, but these requirements aim to try and foresee any suspected future patches potentially required (Robertson and Robertson 2006).

- The product should allow for compatibility with future versions of Android
- The product should allow for easy translation into various foreign languages

6.5 Risk Review

"I am used to thinking three or four months in advance about what I must do, and I calculate on the worst. If I take so many precautions it is because it is my custom to leave nothing to chance." – Napoleon I, March 14, 1808

A risk is when a potentially unwanted event has a possibility of occurring, carrying with it some form of negative consequences (Software Testing: An ISEB Intermediate Certificate. 2009). Risk Management is a process undertaken to help identify and mitigate these potential risks. It consists of four stages; these are risk identification, risk analysis, risk mitigation, and risk monitoring (Software Testing: An ISEB Intermediate Certificate. 2009).

Using the steps involved in risk management, as well as calculations for risk exposure, the following table has been devised, where Risk is determined using risk identification, the Strategy to Avoid is determined with risk mitigation, and the Probability and Consequence is determined through risk analysis. The Risk Exposure however, is calculated by multiplying the probability of the risk occurring, with the consequences if it were to occur. Quantifying risks with risk exposure allows for the identification of a priority order with the risks concerned (Hall, 1998).

Risk	**Strategy to Avoid**	**Probability (out of 5)**	**Consequence (out of 5)**	**Risk Exposure**
System Failure	Create regular backups and have an alternative system that can be used to resume work on	2	5	10
Natural Disaster	Create regular offsite backups including use of version control repositories	1	5	5
Git repository compromised	Have multiple instances of local pulls	1	2	2
Malware or ransomware	Use an up-to-date antivirus software, and avoid downloading suspicious software	1	3	3
Underestimated development time	Create a realistic development schedule, ensuring for contingency	3	2	6
Poorly defined requirements	Develop clear and detailed requirements prior to initiating design or development	2	2	4
Poor software quality	Ensure significant testing during development enabling sufficient time to fix bugs	2	2	4
Illness	Maintain a healthy lifestyle and ensure contingency has been factored into scheduling	3	2	6
Unforeseen circumstances	Factor in contingency within the project's development schedule	4	2	8

7 TECHNOLOGICAL REQUIREMENTS

7.1 Proposed solution

7.1.1 Android Mobile Application

A mobile application is chosen as the proposed solution because the product is targeted towards students, and the majority of student's lives revolve predominantly around their smart phone. This ranges from socialising, to media consumption, to navigation, so it is evident that the next evolutionary step is revision guides to go along with the already existing organisational and working tools.

The mobile application is to be programmed to operate on the Google Android operating system. Firstly, this is because the worldwide market share of Android devices far exceeds its closest opposition, with 87.5% in the third quarter of 2016, compared to Apple iOS at 12.1% (Sui, 2016). Secondly, the Android operating system is far more developer friendly than its iOS counterpart, using a ubiquitous programming language, Java, rather than the Apple's own Swift programming language.

7.1.2 Alternative Options

When deciding upon the platform in which the software was to be developed, there were numerous viable options. Firstly, a java based computer program was considered. This method not only reached a large audience, but also allowed for the easiest and most natural I/O when entering and manipulating data in the system. This however, failed to alleviate one of the drawbacks of existing systems, the need for ubiquitous availability. If it were to be a computer software, it would stay on the computer, where if the user did not have their computer with them, they also would not have their timetable with them.

To maintain the advantages of the computer software and to remedy the disadvantages, a web based solution was considered. This would mean that, whenever the user had an active internet connection on any device, they would be able to access their timetable. However, this violates one of the proposed non-functional requirements, "the product should be available to use, regardless of an active internet connection".

Once a mobile application approach was decided upon, the obvious Android verses iOS deliberation took place. As explained previously in section 7.1.1, iOS uses their own proprietary programming language called Swift written using Apple's IDE, Xcode, of which the required hardware to run was not available – a Mac running macOS ("Apple Developer Program", 2017). In addition, to develop and publish applications for iOS, a $99 a year fee is required, whereas a one-off payment of $25 is required for Android development and publication ("Google Play Developer Console", 2017).

7.1.3 Overview of Android Programming

The Android developer API guide states, "Android is an open source, Linux-based software stack created for a wide array of devices and form factors" ("Platform Architecture", 2017). Of the major components of the Android Platform, the Java API framework is the entire feature set of Android in APIs written in the Java language, which aim to simplify the reuse of core, module system components and services ("Application Fundamentals", 2017). Android applications are therefore written using the Java programming language utilising these APIs. The created Java code, along with the resource files, are compiled into an Android Package called an APK, which is then installable on an Android device.

There are four main app components of an Android app, each with their individual purpose. These are Activities, Services, Content Providers, and Broadcast Receivers ("Application Fundamentals", 2017).

An activity is represented by a single screen with a user interface. Each activity is independent of each other; however, they can work together and be accessed from one another to form a unified experience ("Application Fundamentals", 2017).

A service has no user interface, instead it is a process that runs in the background to carry out some sort of operation ("Application Fundamentals", 2017).

A broadcast receiver also has no user interface, instead it is a component that allows the system to deliver events to the app away from the usual operations, including when the app is not running. This means the app does not need to be constantly running if a notification is scheduled to be pushed at a certain time ("Application Fundamentals", 2017).

A content provider manages persistent and shared storage locations, such as an SQLite Database. This data can be queried and modified providing the content provider grants the necessary permissions ("Application Fundamentals", 2017).

The manifest file is a compulsory file, stored at the root of the project directory, that carries out several roles. Most importantly, it is used to declare all the components in the file; the system reads the manifest to find their existence. Other features include declaring user permissions, minimum API levels required, and additional API libraries that the app needs ("Application Fundamentals", 2017).

7.2 Integrated Development Environment (IDE)

An IDE, or integrated development environment, is a software package that amalgamates all the tools often required when writing and testing software. They utilise a graphical user interface to incorporate their core features, often including a code editor, a compiler, a debugger, and a virtual machine (VM) platform or emulation suite. They regularly have some form of integration with version control libraries, such as Git and SVN (Rouse, 2016).

7.2.1 Android Studio IDE

Android Studio is the official IDE for Android app development. It is based on the IntelliJ IDEA IDE, with added tools specific for Android development, including a Gradle-based build system, an Android emulator, and extensive testing tools and frameworks ("Meet Android Studio", 2017). Being the official IDE for Android comes with many benefits and so resulted in Android Studio being the IDE of choice. These benefits range from always being up-to-date with the latest features and firmware, the most recent update being from March 2017, to having an extensive range of guides and tutorials readily available online.

7.2.2 Alternative Options

When it came to Android development, there are only a small number of viable alternatives in regards to the IDE. One such substitute is the Eclipse IDE with the Android Developer Tools (ADT) plugin. The ADT plugin offers GUI-based access to a large proportion of the command-line Android SDK tools ("ADT Plugin", 2015). However, at the end of 2015, development and official support for ADT terminated (Eason, 2015). Therefore, for obvious reasons, Eclipse with ADT was not selected as the IDE of choice.

A second option is Visual Studio. Visual Studio takes a different approach, whereby it aims to target multiple platforms in the same solution, sharing the code from UI elements. This is done through C# and the .NET Framework, amongst other languages such as HTML and JavaScript, and C++ ("Cross-Platform Mobile", 2017). As this project aims to target solely the Android platform, this approach is not necessary as it undertakes goals outside the scope of the objectives.

7.3 Programming Language

7.3.1 Java

As explained in section 7.1.3, Android apps are predominately written in Java, enabling interaction with the Java API framework. Therefore, it is the obvious decision to use Java as the programming language of choice for this project.

Java is a high-level computer programming language that is concurrent, class based, and object-oriented (Gosling et al., 2015). It is designed so that once the Java code is compiled; it can be run on any platform that supports Java without the need for recompilation ("Java Language Environment", 2017).

7.3.2 Alternative Options

As of the release of Android Studio 2.2, released in September 2016, the Native Development Kit (NDK) could be used to compile C and C++ code into a native library and packages it into the APK ("Getting Started with", 2017).

While Java can generate code quickly using the JVM-optimised byte-code, native machine code is faster, especially with intensive operations such as gaming, and physics simulations. In addition, the lack of garbage collection means that the memory footprint is far smaller than Java's (Bolton, 2016).

While all of this makes C and C++ an attractive and viable option, these advantages are not necessary for the proposed solution due to the lack of any demanding processes taking place.

7.3.3 XML

A layout in Android, is a definition of the visual structure for the user interface within an application. These UI elements are declared using an XML vocabulary provided by Android which corresponds to the View classes ("Layouts", 2017).

Extensible Markup Language (XML) is a human and machine readable language designed to store and transport data ("XML Tutorial", 2017).

Whilst there is an alternative for declaring layouts, being programmatically instantiated at runtime, it is recommended that they are declared using XML to better separate the presentation of code ("Layouts", 2017).

7.4 Hardware

7.4.1 Development Device

As discussed in section 7.2, Android Studio is the chosen IDE for development. A machine meeting the minimum specification is therefore required to run the program in one of the supported operated systems; Windows, Mac, and Linux. The machine to be used for development will be a Windows computer, with specs against the minimum requirements below.

Android studio requirements ("Android Studio System", 2017):

- Microsoft® Windows® 7/8/10 (32- or 64-bit)
- 3 GB RAM minimum, 8 GB RAM recommended; plus 1 GB for the Android Emulator
- 2 GB of available disk space minimum,
- 4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
- 1280 x 800 minimum screen resolution
- For accelerated emulator: Intel® processor with support for Intel® VT-x, Intel® EM64T (Intel® 64), and Execute Disable (XD) Bit functionality

PC specification:

- Microsoft® Windows® 10 64-bit
- 16 GB RAM
- 128 GB SSD with 1TB SSHD Hybrid Drive
- 2560 x 1440 monitor
- Intel® Core i5 4690K CPU – supports VT-x, Intel® 64, and Execute Disable Bit ("Intel® Core™ i5-4690K", 2017)

7.4.2 Testing Device

As development progresses, a testing device will be required to run the application to test for expected behaviour. In Android Studio, this can be done in two ways. First is to use real hardware in the form of an Android device, and second is to use the Android Emulator built into Android Studio. The Android Emulator simulates any device, and then displays it straight from the development computer. It supports Android phones and tablets, as well as Android Wear and TV devices, ideal as additional or specific hardware devices are not required ("Run Apps on the", 2017).

Both the emulator and a hardware device will be used for testing and development. The hardware device to be used is a Huawei Nexus 6P running unrooted stock Android, version 7.1.2 Nougat. It has a screen with a resolution of 1440 x 2560 with a xxhdpi (extra-extra-high dots-per-inch) pixel density at ~518ppi ("Nexus 6P Specifications", 2017).

7.5 Version Control

Version control is a system which records and stores changes made to files over time so that specific versions can easily be recalled at some point in the future (Chacon and Straub 2014). This is especially important in software development, as it allows for code to be reverted to an earlier revision if a bug is discovered in a new code commit.

7.5.1 Git

The version control software (VCS) to be used for this project is called Git. Git differs from other version control software as, instead of storing a set of file based changes, it takes a snapshot of the current state of the files for each commit. And where a file has not changed, a link to the previously identical file is made, making for greater efficiency. In addition, the majority of Git's operation is local, meaning speeds are greatly increased due to the omission of network latency overheads (Chacon and Straub 2014).

Git was the chosen VCS for a variety of reasons. Firstly, was the previous experience held with handling Git, most notably in a professional industry setting within a fast-moving agile team of multiple programmers. Secondly, Git has seamless integration within Android Studio, so not only mitigating the likelihood of mistakes occurring through improper use of a command line, but also improving efficiency by being able to review the differences in all the files being committed in a clearly laid out GUI before any push is made.

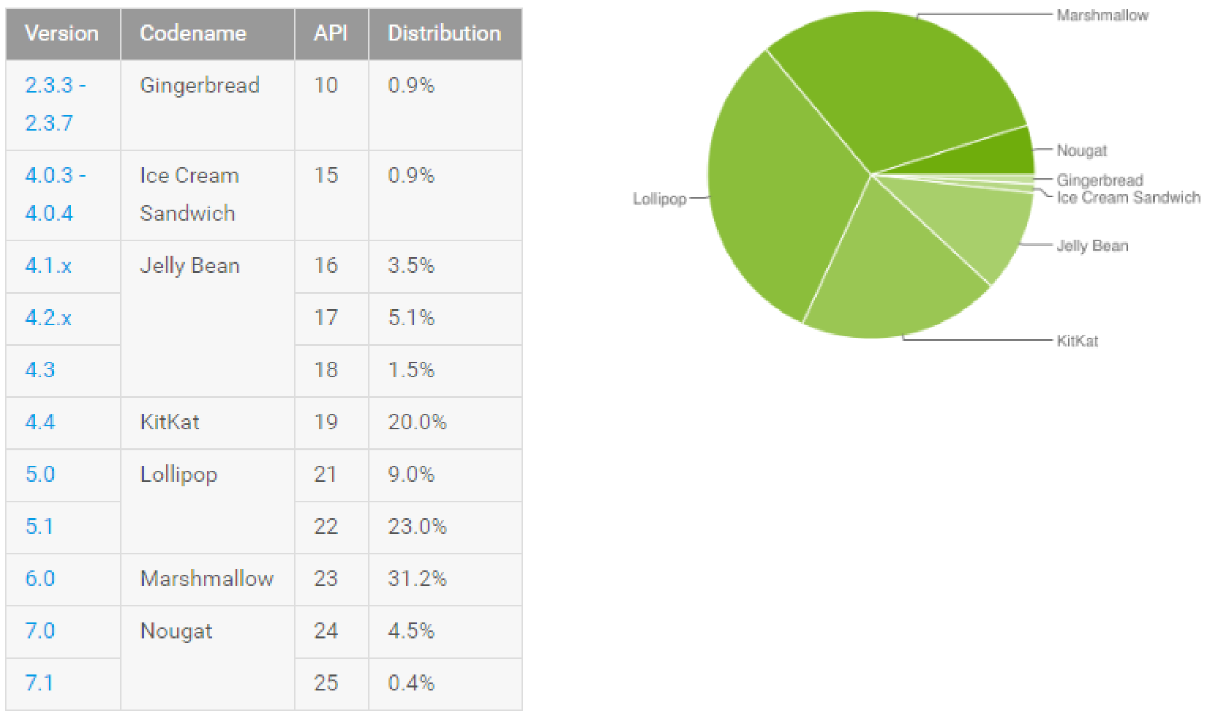
7.5.2 Alternative Options

Whilst there are a large number of alternative VCS, the principal alternative considered was Apache Subversion, or SVN for short. Primarily as there was a remote SVN repository available to use, but additionally because of SVN’s ability to check out a single subdirectory of a repository; Git is unable to carry out this action (Pearce, 2013). Whilst a significant advantage, it applies less with this project as it will not be an industry scale endeavour, meaning the time costs of a whole repository download is not as impactful.

7.6 Target Platform

Roughly every year, the Android operating system receives a software version update. This update usually brings with it a host of new features for both the end user and the developers, the latter of which is through a new API level. Therefore, new features developed for a new API will not be available in previous Android versions where this API was yet to exist. When developing an Android application, a minimum and target API level must be stated, known as minSdkVersion and targetSdkVersion. The aim is to target the latest version of Android, while maintaining backwards compatibility with older versions.

When deciding the min and target SDK version for this application, the Android Developer Platform Version Dashboard was consulted. This dashboard provides data about the relative number of devices running a given version of Android ("Android Dashboards", 2017). The targetSdkVersion will be 25 (7.1 Nougat), as it is currently the latest version available, and the minSdkVersion will be 14 (4.0 Ice Cream Sandwich), as it covers 99.1% of active devices *(as of 3rd April 2017)*.



Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.9%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.9%
4.1.x	Jelly Bean	16	3.5%
4.2.x		17	5.1%
4.3		18	1.5%
4.4	KitKat	19	20.0%
5.0	Lollipop	21	9.0%
5.1		22	23.0%
6.0	Marshmallow	23	31.2%
7.0	Nougat	24	4.5%
7.1		25	0.4%

Figure 7-1 Android Platform Distribution

8 Design and Implementation

8.1 Database Management System

Data storage is a key component of any system and so careful consideration is required when determining which method of storage is to be implemented. Given that the proposed solution is an Android application, this limits the number of storage options available. These options include Shared Preferences, Internal Storage, External Storage, SQLite Database, and via a Network Connection ("Data Storage Options", 2017). As Shared Preferences is used for primitive data types in key-value pairs, and the internal and external storage are for saving files, none of these implementations were suitable for the given use case.

A database was required to store the expected data which would be saved onto the system. This meant the choice of a local database, through SQLite, or a remote database, through a network connection, had to be decided upon. Whilst a remote database meant that the user's data would be readily available across multiple devices, it also meant that the user needed an internet connection to use and update data in the system. Given that a non-functional operational requirement states that the product should always be available to use, regardless of an active internet connection, it meant that the local SQLite Database would have to be used. SQLite is the most used and most widely deployed database engine available, claiming to be self-contained, server-less, and of the most reliable ("About SQLite", 2017).

An entity-relationship diagram (ERD) is a data modelling technique in which the relationships between objects in a system, is represented in a graphical layout (Rouse, 2014). Figure 8-1 shows the ERD for this system.

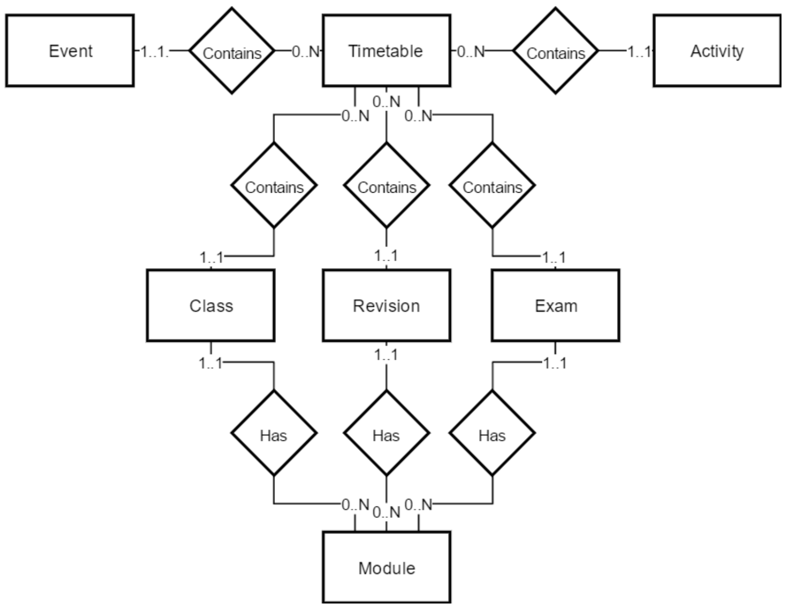


Figure 8-1 Entity Relationship Diagram for Smart Revision Calendar Application

The following tables show the individual tables within the database, delineating the attribute name, its data type, whether it is required in each record, and whether it has a special key type.

Attributes	Data Type	Key Type	Required
Class Table			
Class ID	Integer	Primary Key	Yes
Module	Text		Yes
Title	Text		Yes
Day	Text		Yes
Start Time	Text		Yes
End Time	Text		Yes
Repeat	Integer		Yes
Teacher	Text		
Room	Text		
Colour	Text		Yes
Activity Table			
Activity ID	Integer	Primary Key	Yes
Title	Text		Yes
Day	Text		Yes
Start Time	Text		Yes
End Time	Text		Yes
Repeat	Integer		Yes
Colour	Text		Yes
Exam Table			
Exam ID	Integer	Primary Key	Yes
Module	Text		Yes
Title	Text		Yes
Date	Text		Yes
Start Time	Text		Yes
Duration	Integer		Yes
More Info	Text		
Colour	Text		Yes
Content Size	Integer		
Priority	Integer		

Attributes	Data Type	Key Type	Required
Revision Table			
Revision ID	Integer	Primary Key	Yes
Module	Text		Yes
Title	Text		Yes
Date	Text		Yes
Start Time	Text		Yes
End Time	Text		Yes
Notes	Text		
Colour	Text		Yes
Event Table			
Event ID	Integer	Primary Key	Yes
Title	Text		Yes
Date	Text		Yes
Start Time	Text		Yes
End Time	Text		Yes
Colour	Text		Yes
Debrief Table			
Debrief ID	Integer	Primary Key	Yes
Productivity	Integer		Yes
Revision Length	Integer		Yes
Revision Rating	Integer		Yes
Break Length	Integer		Yes
Break Rating	Integer		Yes
Start Time	Text		Yes
Start Rating	Integer		Yes
End Time	Text		Yes
End Rating	Integer		Yes
Variety	Integer		Yes
Variety Rating	Integer		Yes

8.2 Javadoc

Javadoc is a tool used for generating documentation in a HTML format from comments made in Java source code ("Javadoc Tool", 2004). This is done through the use of Documentation Comments, in which they are placed before the declaration of a class, interface, method, constructor, or field ("javadoc", 2010). Writing Javadoc comments carry many benefits, however most notably being that it massively increases the readability of code, resulting in easier testing and maintenance long after the code's creation.

Javadoc has been used within this project, where every class, and every method has been commented upon. The format of these comments stay consistent throughout, where for each class the fundamental information is incorporated, including: class name, class description, author, version in which it was added, and creation date.

```
/**
 * <h1>Title</h1>
 * Description
 *
 * @author  Russell Waterson
 * @version 1.0, 22-03-2017
 * @since   dd-mm-yyyy
 */
```

The fully generated Javadoc in HTML format can be found within the project zip, as described in Appendix A.

8.3 Package Management

As explained in section 7.1.3, the Android build system compiles source code and resources into APK files. To automate and manage the build process, whilst still allowing for flexibility within configurations, Android studio uses Gradle ("Configure Your Build", 2017). Gradle is an open source build tool that replaces XML based build scripts for declaring project configuration, previously used by Apache Maven, with domain-specific language (DSL) based on the Groovy programming language (Kainulainen, 2004). If the project were to be built outside of Android Studio using the command line, a separate set of build tools would be required. Apache Ant is an example of such, used mainly for building Java applications ("Ant Apache", 2017); however, such tools were not used in this project, only Gradle.

8.4 Key Components

This sections aims to outline core components of the system, how they work, and how they are related. The class diagram created, seen in Appendix B, demonstrates the organisation of the classes used. A class diagram is used to document the structure of a system by showing the classes and their relation to each other (Stevens and Pooley 2000). While the class diagram shows the classes and their relation, the following sections outline how each class or component achieves their defined behaviour, each of which relates to its applicable user interface component in section 9.4, where 8.4.x equates to 9.4.x.

8.4.1 Initial Start-up

Upon the very first launch of the application, the user will be presented with a series of screens, welcoming the user, explaining elements of the system, and allowing them to enter some initial data. This process was implemented using the Steppers component from the Material Design Guidelines, of which is yet to be implemented into the first party libraries, and as such, a third party library, Material Steppers by StepStone Tech, was used (StepstoneTech , 2017). In order of their appearance, the screens consist of, Welcome, Add Class, Add Activity, Google Calendar Sync, Cloud Backup, Explanation of Smart Calendar, Add Exams, Learning Style, and Debriefs. Each section tapping into relevant components of the system, reusing the components of the main system as to ensure consistency.

This start-up section relates back to the Intro Screen functional requirements, whereby the application should inform the user about major features and how the app operates. This is an important process to have within the application as, as well as meeting a non-functional requirement of Usability and Humanity, it also meets one of Jakob Nielsen’s 10 Heuristics, providing “help and documentation”. In 1994, Jakob Nielsen, derived a set of 10 heuristics, of which are a broad set of ideologies for creating an interactive design (Nielsen, 1994).

8.4.2 Main Screens

The main screens of the application are the top layer in the navigational tree, and from here the user can access every part of the application. Upon its first boot, the user is presented with a dialog explaining how to navigate the app. The top-level views in the application, navigated via the Bottom Navigation component, are Timetable, Exams, Classes, and Activities, all of which are separate fragments. Within the exams, classes, and activities fragment, is a Floating Action Button to take the user down a level into the system, where they can add a new instance of that respective item. A side navigation drawer is used for selecting options taking the user down a level, for options such as settings, smart data, and import external events, each of which using a new activity.

8.4.3 Timetable Views

The timetable views display the calendar layouts, including a day, week, and month view. The day and week views show all the events in the system, and is where the user can see their generated timetable. They are implemented using the Android Week View third party library (Kanak , 2016). The month view however, simply shows a month calendar view the number of days until the next exam. This is implemented using the built in Android component called Calendar View. The day and week view timetables are generated using the third party library, but only after the data has been parsed into it. This is done by querying and returning all of the records from the Class, Activity, Exam, and Event tables in the database. All of this ensures that all the Timetable functional requirements have been met.

Whilst using a built-in component is a favoured approach, the Calendar View class lacks in some functionality, and so an additional third party library was considered. The One Calendar View library (Morocho , 2017) seemed to be a promising addition, however, this implementation was low in priority as a month view was already in place, and in the interest of time, was not incorporated into the application.

8.4.4 Database Queries

As explained in section 8.1, the application uses SQLite to manage its local database; this is done through the created DatabaseHelper class which extends SQLite Open Helper. All database queries are made through this class via public methods; this includes the initial creation of the 6 tables, and adding, deleting, editing, and viewing records from each table. These public methods are accessed throughout the application.

The insertClassData, insertActivityData, and insertExamData methods are called when adding a new item to the database, where all the user inputs have been pre-validated, providing “error prevention”, one of Nielsen’s Heuristics. When adding a class or exam, the Module field is an entity which can be accessed across different components in the system. Each table is selected and each unique module is fed back to the user to choose from an existing one, or to add a new one. The insertRevisionData method is called when the smart revision has been generated and all fields are entered by the system. The insertEventData method is called when adding an event imported from Google Calendar and therefore the fields are generated. The insertDebriefData method is called upon after completion of daily debrief where the fields are also generated based on a limited set of user interactions.

The updateClassData, updateActivityData, updateExamData, and updateRevisionData methods are called when the user opts to edit an individual item. The user's inputs are also pre-validated before the record update takes place.

The deleteClassData, deleteActivityData, deleteExamData, deleteRevisionData, and deleteEventData methods are all called from when the user selects to delete that item from its respective individual screen.

The getAllClassData, getAllActivityData, and getAllExamData methods are called when the data in each table is to be viewed. These methods are called predominately in their respective fragments, displayed each record in a custom-built component, as well as in the timetable view. The getAllEventData is also called mainly from the timetable view. In addition to the timetable view, the getAllRevisionData is also called when generating new smart revision so that it can be compared with the newly generated data. The getAllDebriefData method is called when the user wants to update their learning style with the user predicted values.

All these methods aim to satisfy a large number of functional requirements, including ones from class, activity, exams, revision, and import events sub-headers.

8.4.5 The Smart System

The smart system is what allows the application to generate revision slots for the user. It is compromised of the smart algorithm, containing over 620 lines of code, and the smart data, of which is fed into the algorithm. This algorithm can be found from within the com.russell.smartrevisioncalendar.smartdata.CreateRevisionBlocks class. The system is built so that every user will get a difference experience, tailored to their exact specification, and exact individuality and habitual nature.

The smart data is initially input into the system by the user, either during the initial start-up, or if subsequently through the settings. It consists of the user's preferred start and end time for revision, their preferred revision length and break length, and the amount of variety between tasks when revising.

The smart algorithm consists of two core steps; the creation of the revision blocks, and the allocation of revision to the created blocks; both of which are run as an intent service and therefore in the background on a different thread.

To complete the first step, the system takes a wide range of data from throughout the system, this includes, preferred start and end time, revision and break length, last exam in the database, and other items in the database in order to avoid clashes. With this data, the system populates the revision database with revision slots, with length of the revision plus breaks in-between, up to the date of the final exam in the system. These blocks are generated within the start and end times, and make sure that they avoid all other events in all other databases (classes, activities, exams, and Google Calendar Events). The following flowchart in Figure 8-2**,** explains how this works in a relatively straightforward depiction.

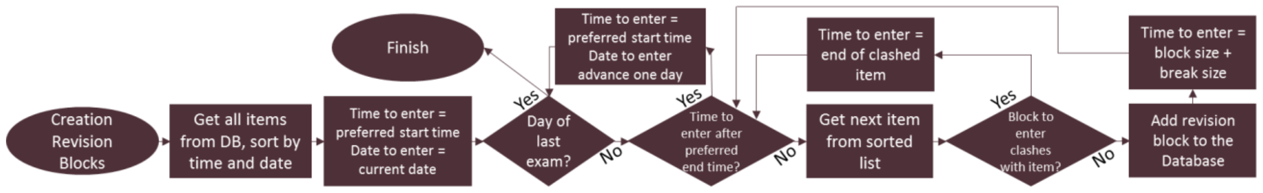

Finish
Time to enter = preferred start time Date to enter advance one day
Time to enter = end of clashed item
Time to enter = block size + break size
Creation Revision Blocks
Get all items from DB, sort by time and date
Time to enter = preferred start time Date to enter = current date
Yes
Day of last exam?
No
Yes
Time to enter after preferred end time?
No
Get next item from sorted list
Yes
Block to enter clashes with item?
No
Add revision block to the Database

Figure 8-2 Flowchart for the creation of revision blocks

To complete the second step, a custom-built probabilistic algorithm was created. This works out which revision should be done when using the following inputs: variety, exam content size and priority, and the days until that exam. The larger the content size and larger the priority relative to other exams in the system, the higher probability it has of being selected. Likewise, the lower the number of days till that exam relative to others, the higher the probability. The variety comes in when a subsequent revision slots are to be generated. The higher the variety, the higher the likelihood of it being a different exam to revise for. The following flowchart in Figure 8-3**,** explains the steps taken in this probabilistic algorithm to achieve allocation of revision.

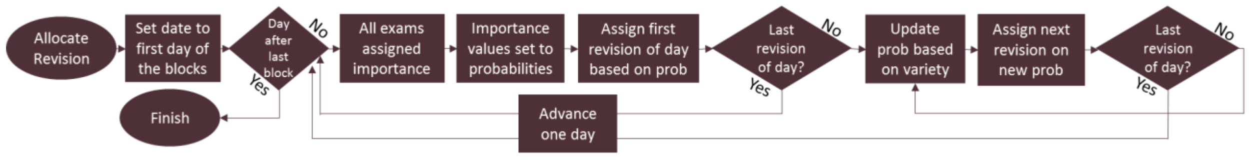

Allocate Revision
Set date to first day of the blocks
Day after last block
No
Yes
Finish
All exams assigned importance
Importance values set to probabilities
Assign first revision of day based on prob
Last revision of day?
No
Yes
Advance one day
Update prob based on variety
Assign next revision on new prob
Last revision of day?
No
Yes

Figure 8-3 Flowchart for the allocation of revision using the probabilistic algorithm

After the revision has been generated, the user is given the opportunity to make a comparison between the old revision and the new revision. This is in the form of a side by side exposition of the revision generated by day, as well as a module count for how many revision slots have been created for that module. This is done by doing a series of database queries to the revision database and working out which records have newly been generated and which were old. Then making the necessary calculations for the revision count, and then adding each record to a custom-built UI component on a per day basis. However, if it is the first time revision has been generated, the user will not get this option, and instead will be informed that it will be available on subsequent generations.

As the smart algorithm is the heart of the system, many alternative approaches were considered when tackling this problem. The main alternate method considered, was the use of machine learning. Machine learning was defined by Arthur Samuel in 1959 as giving "computers the ability to learn without being explicitly programmed" (Munoz, 2014). In machine learning, there are many approaches and algorithms suited for a wide array of applications, ranging from Artificial Neural Networks and Deep Learning, to Clustering and Bayesian Networks. Originally the use of Google's TensorFlow library was evaluated, but upon deeper research, it and machine learning as a whole was rejected. TensorFlow is an open source library using data flow graphs for numerical computations, where a node is a mathematical operation and an edge is a multidimensional data array ("About TensorFlow", 2017). The reason for machine learning being rejected as an approach for the smart learning system within the application, was because machine learning requires significant amounts of data, and imposed additional computational stress. As the application relies on offline computations only, to meet the operational non-functional requirement, the scalability that comes with machine learning allows for a distributed training infrastructure, of which will require access to a remote learning engine. Following on from this, machine learning processes usually involve intensive computations, inflicting heavy stress on CPUs and GPUs. The proposed system aims to be available on a wide range of devices, including those of older hardware. Incorporating machine learning may inhibit this, as well as restrict offline only computations, therefore its use has been dismissed.

8.4.6 Debriefs

The debriefs allow the application to fine tune the learning style of the user based on their continued feedback, aiming to meet a number of the Smart Revision and Data functional requirements previously summarised. At the end of each day containing revision, an internal alarm is triggered so that a notification on the mobile is displayed informing the user to complete the debrief, regardless of whether they are currently using the app or not. The alarm is also updated when the phone is rebooted, there is change in time zone, or the internal clock has been changed.

Once each debrief is completed the input data is stored as a record in the Debrief Table in the database, with fields of the record as described in section 8.1. All of this data will eventually be used to calculate and recommend to the user how they should update their learning style data to help maximise their learning output. This can be done via the Smart Calendar Data settings screen and choosing the “Recommend Smart Data Updates”. A complex, custom-made formula is used to calculate the updated values, which works roughly as follows:

First, the productivity of the day is taken from the number of stars selected by the user, it is then used to calculate the severity of change for each of the components, where the more productive the day, the less each value will change. The discrete slider position for each of the respective values represents how the value will change, where in the centre represents no change, and the further left and further right it goes represents a more significant change. This is done for every single debrief record in the database, and the average of all the values for each field is taken, resulting in the new recommended smart data. Therefore, the more the user feeds back data to the system, the more accurate the predictions will become. The user is then told the recommended values, and has a choice of whether they would like to accept them or not.

8.4.7 Cloud Backups

The cloud backups allow the user to not only backup all their records into the cloud, but to also be able to view these records from another device, thus meeting the Backups functional requirements. Upon surveying a number of potential users, there was 90.48% interest, being very or mildly important, in having their calendar viewable on another device, and so, this feature was included in the functional requirements.

For this feature, Google Drive was the cloud storage option to be implemented. The reason Google Drive was implemented as opposed to alternatives such as Dropbox or OneDrive, was because Google Drive’s readily available use, as well as online support, for their APIs is inimitable. In addition, it has over 240 million active users as of October 2014, and over one million paying users as of September 2015 (Darrow, 2015), and has been consistently on the rise.

To use a Google API, a number of steps are required before they able to operate within the application. First a Google API Manger account had to be setup where access to Google’s APIs were requested. Then the application must be authorised against OAuth 2.0. OAuth 2.0 is the industry-standard protocol for authorisation, which focuses on client developer simplicity ("OAuth 2.0", 2017). This was achieved by registering the application with its certificate, obtained by running a Keytool utility to get the SHA1 fingerprint of the digitally signed apk file ("Getting Started", 2017). Once complete, the code could then be written to access Google Drive.

Once the connection to the user’s Google Drive is established, two files will be offered to be backed-up, these are a text file, and the raw database file. As the name suggests, the raw database file is taken straight from the phone’s storage without any alterations. The text file however, is generated by the system; it parses all the tables in the database, and writes out each record to a text file, under the heading represented by each table name.

8.4.8 Import Google Calendar

The user has the option to connect their Google Calendar to the application so that they are able to import chosen events, which can then be considered during the timetable generation. For this feature, the Google Calendar API was to be utilised. The reason for Google Calendar’s use, as opposed to alternatives such as iCal or Outlook Calendar, was main due to the fact that Google Calendar is the default calendar application and provider built into all Android phones. Google Calendar API also requires the use of OAuth 2.0, which required the following of the same steps outlined in section 8.4.7.

After the user has chosen the number of records to search for, and the connection to their Google Calendar has been established, a list of these events will be displayed within custom-built UI components per event. The API however, returned a long and convoluted string containing the event's time and date details. To convert this to a user readable format, regular expressions had to be utilised. A regular expression is a special text string for depicting a pattern to be used in a search query (Goyvaerts, 2016). After selecting the events to add, scrolling to the bottom of the page and clicking the "Save" button will add the selected items to the events database, thus meeting the Import Events functional requirements.

8.4.9 Mute Phone

The mute phone functionality is a user activated setting whereby the user can choose to have the application automatically set the phone to silent during the hours of revision. This was an additional feature outside of the main aims of which had 80.95% interest, being high or slight increase in productivity, where surveyed potential users felt it would be a beneficial addition, potentially minimising distractions. And so, it was introduced into the functional requirements as Phone Muting.

In a similar way as the debriefs, an internal alarm is set at the preferred start time and preferred end time, however to mute and unmute the phone respectively, instead of showing a notification.

8.4.10 Settings

The settings screen and the smart calendar data screen are both settings activities where the user is able to edit data and preferences that are persistent throughout the application. These activities use the PreferenceFragment which provides a platform in which to access the shared preferences, the persisting data within a system ("PreferenceFragment", 2017).

From the settings screen, the user can:

- Activate and deactivate phone muting, as described in 8.4.9
- Rerun the initial setup walkthrough, as described in 8.4.1
- Write an email to the developer, whereby the user's email client is opened with a draft email containing the dev's email address, and the subject "Smart Revision Calendar".
- View changelog
- View a dialog explaining more about the application and its inception

8.4.11 String Resource File

As defined in the non-functional requirements in section 6.4, the ability to allow for easy translation into various foreign languages in future maintenance and support is a desired requisite. The implemented string resource file aims to satisfy this requirement. The string resource file contains all the text strings within the application, so that the translation of the written text can be done using just the single file without the need for going through the application's code.

9 User Interface

The user interface (UI) and user experience (UX) of a piece of software are paramount for its success; if a user does not like the way an application looks or operates, they will be less inclined to continue using it. To ensure the UX of this project was of the highest quality, a lengthy multi-stage process was undertaken, using both low fidelity and high fidelity techniques.

To aid with this, a set of commonly used principles and rules were considered and intertwined within each design decision. The most prominent being Shneiderman's 8 Golden Rules. These rules consist of: 1) Strive for consistency, 2) Seek universal usability, 3) Offer informative feedback, 4) Design dialogs to yield closure, 5) Offer error prevention and simple error handling, 6) Permit easy reversal of actions, 7) Keep users in control, 8) Reduce short-term memory load (Shneiderman et al., 2016).

9.1 Material Design

A common approach when designing a user interface, is to create an evolution of an existing product. For this application, the Material Design guidelines were utilised as the core of the interface design.

Material design is a visual design language created by Google which aims to deliver a "single underlying system that allows for a unified experience across platforms and device sizes" ("Introduction", 2017). By using a print-based design – typography, grids, space, scale, colour, and use of imagery – Material Design's aim is to put an emphasis on the user's actions thus making the core functionality of a system immediately apparent.

The utilisation of Material Design was decided upon partly due to its massive span of current implementation. In Google made apps alone, there are 16 Android applications that implement Material Design with over one billion downloads ("Google Inc.", 2017); thus, the intention is to create a means of uniformity and familiarity for the user. In addition, its implementation meets the looks and feel non-functional requirement set out in section 6.4.

9.2 Sketches

"The best way to get a good idea, is to get lots of ideas" – Linus Pauling. Sketches are a low-fidelity form of prototyping that allows for a vast range of alternatives to be explored quickly and cheaply, where poor or inferior ideas can be thrown away with minimal cost. A wide array of sketches were constructed in order to achieve exactly that. Figure 9-1 shows a small subset of sketches, drawn long before a computer was even involved, demonstrating an array of different approaches for the user interface.

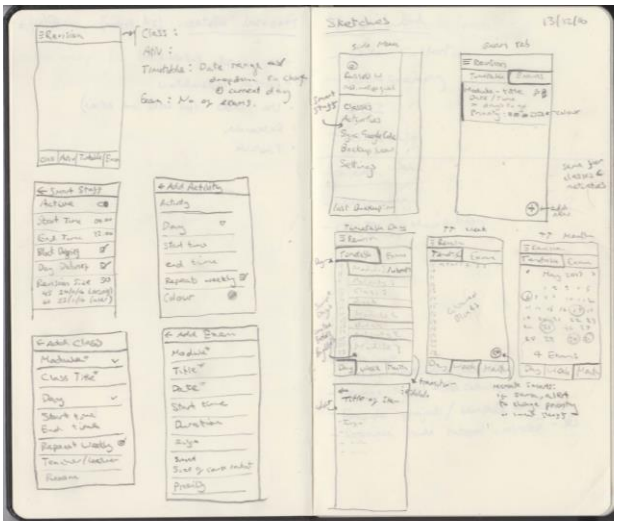
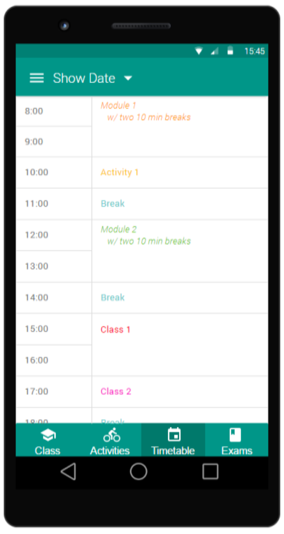


Figure 9-1 Example of UI Sketches

9.3 Wireframes

Once the sketches had been evaluated, a prototype was created in the form of a wireframe, representing how the application and its interactions will operate. A wireframe is a layout of an application demonstrating key interface elements, providing a visual understanding, enabling evaluation, reflection and feedback, as well as being able to test out ideas without its full implementation ("What is Wireframing", 2017).

The software that was used in order to create the wireframes, was called JustInMind. A program specialising in the prototyping of websites and apps for Web, iOS, and Android. An interactive demo for this wireframe can be found in the supplied zip file, as explained in Appendix A. Figures 9-2 through to 9-9 show screenshots from this constructed wireframe.



8:00 Module 1
w/ two 10 min breaks
9:00
10:00 Activity 1
11:00 Break
12:00 Module 2
w/ two 10 min breaks
13:00
14:00 Break
15:00 Class 1
16:00
17:00 Class 2

Figure 9-2 Day Timetable View

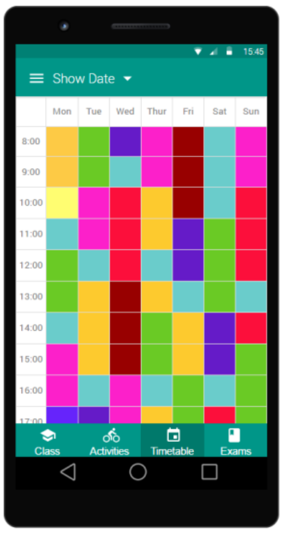
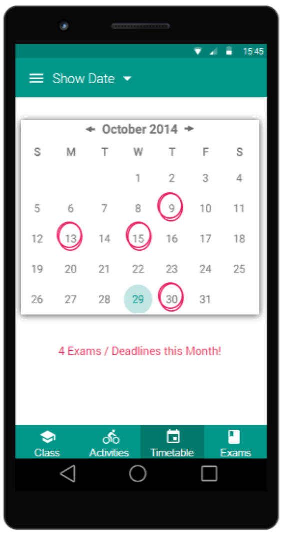


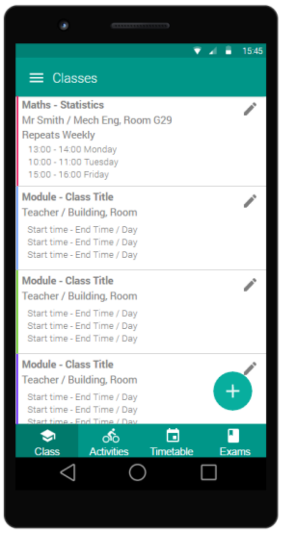
Figure 9-3 Week Timetable View



S	M	T	W	T	F	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

4 Exams / Deadlines this Month!

Figure 9-4 Day Month View



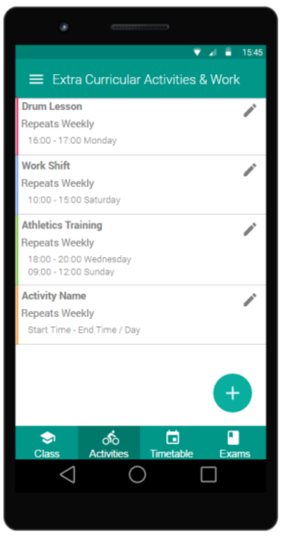
Maths - Statistics
Mr Smith / Mech Eng, Room G29
Repeats Weekly
13:00 - 14:00 Monday
10:00 - 11:00 Tuesday
15:00 - 16:00 Friday

Module - Class Title
Teacher / Building, Room
Start time - End Time / Day
Start time - End Time / Day
Start time - End Time / Day

Module - Class Title
Teacher / Building, Room
Start time - End Time / Day
Start time - End Time / Day
Start time - End Time / Day

Module - Class Title
Teacher / Building, Room
Start time - End Time / Day
Start time - End Time / Day

Figure 9-5 Class List View



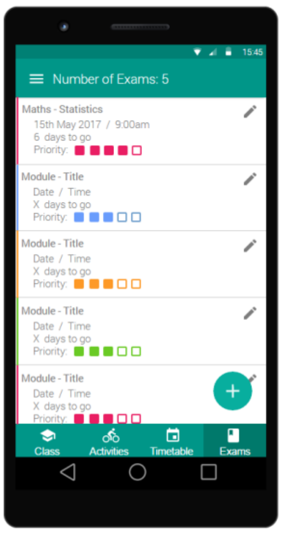
Drum Lesson
Repeats Weekly
16:00 - 17:00 Monday

Work Shift
Repeats Weekly
10:00 - 15:00 Saturday

Athletics Training
Repeats Weekly
18:00 - 20:00 Wednesday
09:00 - 12:00 Sunday

Activity Name
Repeats Weekly
Start Time - End Time / Day

Figure 9-6 Activity List View



Maths - Statistics
15th May 2017 / 9:00am
6 days to go
Priority: ■■■■□

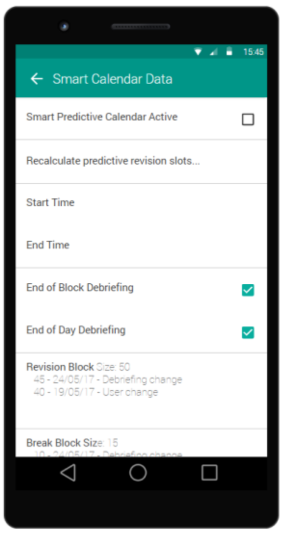
Module - Title
Date / Time
X days to go
Priority: ■■■□□

Module - Title
Date / Time
X days to go
Priority: ■■■□□

Module - Title
Date / Time
X days to go
Priority: ■■■□□

Module - Title
Date / Time
X days to go
Priority: ■■■□□

Figure 9-7 Exam List View



Smart Predictive Calendar Active ☐

Recalculate predictive revision slots...

Start Time

End Time

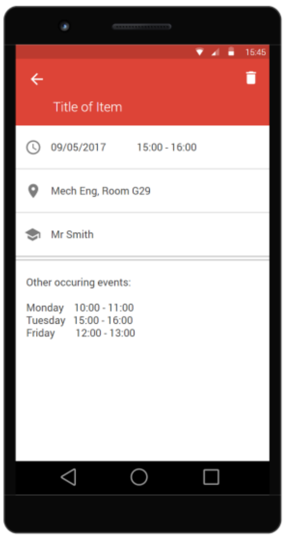
End of Block Debriefing ☑

End of Day Debriefing ☑

Revision Block Size: 50
45 - 24/05/17 - Debriefing change
40 - 19/05/17 - User change

Break Block Size: 15

Figure 9-8 Smart Data Settings



Title of Item

09/05/2017 15:00 - 16:00

Mech Eng, Room G29

Mr Smith

Other occuring events:

Monday 10:00 - 11:00
Tuesday 15:00 - 16:00
Friday 12:00 - 13:00

Figure 9-9 Individual Item Screen

9.4 Final GUI

Once a true to life wireframe had been created, the next step was to find the necessary layout components in the Android library, as well as choose a colour scheme for the application. Of the list of Material Design colours ("Color", 2017), a colour was chosen to stand out from the plethora of existing apps. Brown #795548 is a colour that does not appear often in other apps, and teal #009688 is a good complimentary colour to brown. A study carried out by Stuart Hall in 2015, plotted the top 200 apps on the Google Play Store on a colour wheel, and concluded that majority fall under blue, red, or green.

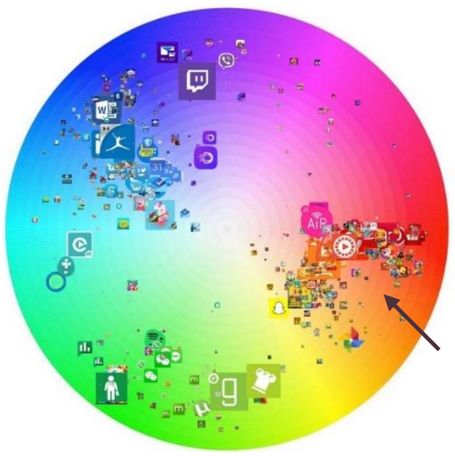


Figure 9-10 Colour wheel plotting app icon colours (Hall, 2015), arrow indicating brown

As discussed in section 6.3.3, the layouts for the user interface are predominately established within XML files in the resource directory. The following sections go into future detail about how the significant UI elements of each section are achieved, each of which relate back to its key component in section 8.4, where 9.4.x equates to 8.4.x. All screenshots are taken from the application running on the Nexus 6P testing device stated in section 7.4.2, however, the UI has been designed so that it can scale to any screen size and resolution, as seen in figures 9-11 through 9-13, and therefore meeting the "looks and feel" non-functional requirement for consistency set out in section 6.4.

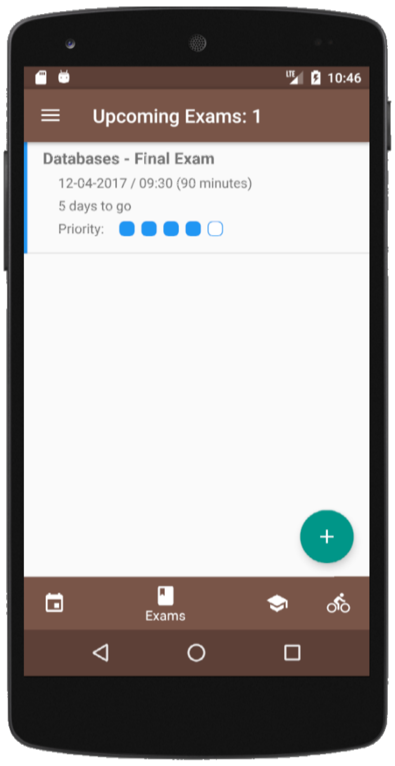


Figure 9-11 Neuxs 5 1080x1920: 445ppi

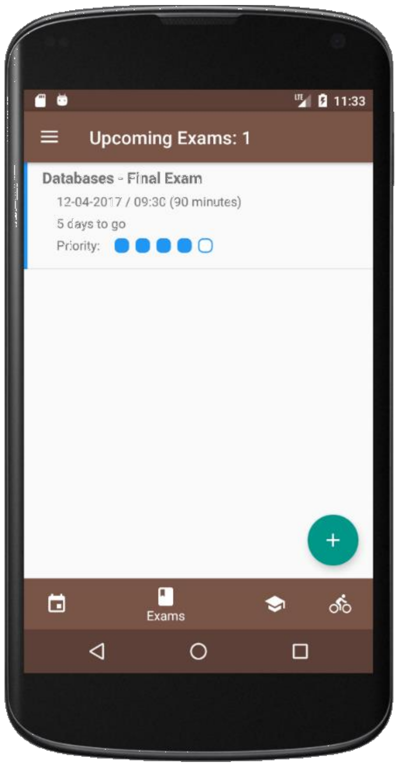


Figure 9-12 Neuxs 4 768x1280: 318ppi

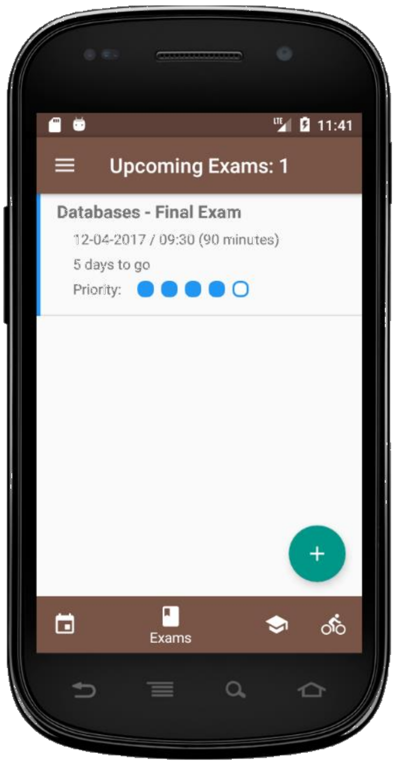


Figure 9-13 Neuxs S 480x800: 233ppi

9.4.1 Initial Start-up

To implement this section of the application, the Steppers component from the Material Design Guidelines was utilised ("Steppers", 2017). As this design component was not implemented into first party libraries, a third party library had to be used. The chosen library, Material Stepper by StepStone Tech, was the closest in likeness to the design spec, so made a suitable fit.

Of the large number of stepper types included within the design spec, and subsequently in the library, the chosen implementation was the "Dots" stepper type. This was because it clearly shows the user's current step as well as the total number of steps in the process, thus conforming to one of Nielsen's Heuristics, offering "visibility of system status" (Nielsen, 1994).

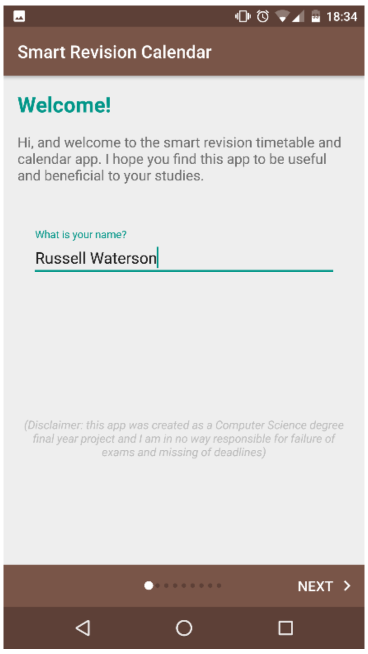
When researching for third party libraries to implement the steppers component, there were a number of others, including Material Stepper by Francesco Cannizzaro (Cannizzaro , 2016), which were to be considered. Whilst this implementation was arguably simpler, it did not have the full feature-set that the chosen library had.

An alternative material component that was considered when implementing the initial start-up, was expansion panels ("Expansion panels", 2017). Expansion panels are often used in the lightweight editing of elements. Whilst this component can be connected to larger surfaces, including cards, it did not have the flexibility that comes with steppers and having separate fragments per step.

Step 1 (Figure 9-14): shows a welcome screen giving the user a chance to enter their name, easing them into the system with a personal touch. Clicking Next here, or on any of the following screens, will take the user to the next step, ensuring a consistent experience; meeting Shneiderman's first rule, "strive for consistency" (Shneiderman et al., 2016).

Step 2 (Figure 9-15): shows classes in the system, clicking "Add Class" button takes the user to the same screen used for adding a class during the app's normal operation, i.e. post start-up, see section 9.4.4. Clicking back here, or on any of the following screens, will take the user to the previous step.

Step 3 (Figure 9-16): shows activities in the system, clicking "Add Activity" button takes the user to the same screen used for adding an activity during the app's normal operation, i.e. post start-up, see section 9.4.4.


18:34
Smart Revision Calendar

Welcome!

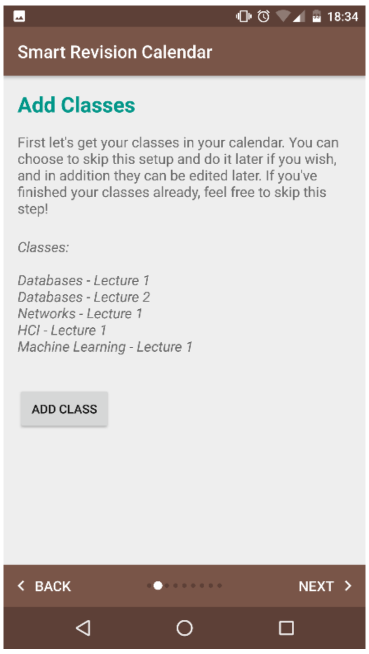
Hi, and welcome to the smart revision timetable and calendar app. I hope you find this app to be useful and beneficial to your studies.

What is your name?
Russell Waterson

(Disclaimer: this app was created as a Computer Science degree final year project and I am in no way responsible for failure of exams and missing of deadlines)

NEXT >

Figure 9-14 Welcome


18:34
Smart Revision Calendar

Add Classes

First let's get your classes in your calendar. You can choose to skip this setup and do it later if you wish, and in addition they can be edited later. If you've finished your classes already, feel free to skip this step!

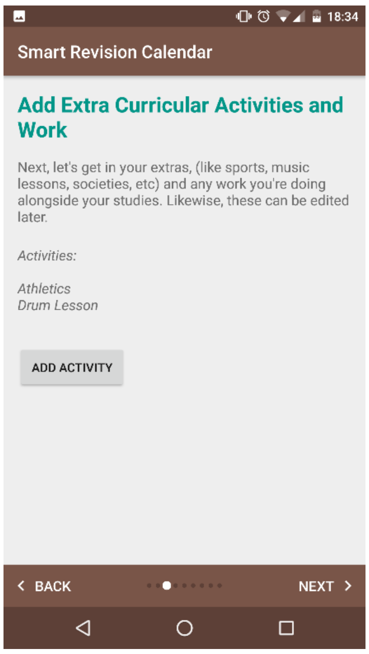
Classes:

Databases - Lecture 1
Databases - Lecture 2
Networks - Lecture 1
HCI - Lecture 1
Machine Learning - Lecture 1

ADD CLASS

< BACK NEXT >

Figure 9-15 Add Class


18:34
Smart Revision Calendar

Add Extra Curricular Activities and Work

Next, let's get in your extras, (like sports, music lessons, societies, etc) and any work you're doing alongside your studies. Likewise, these can be edited later.

Activities:

Athletics
Drum Lesson

ADD ACTIVITY

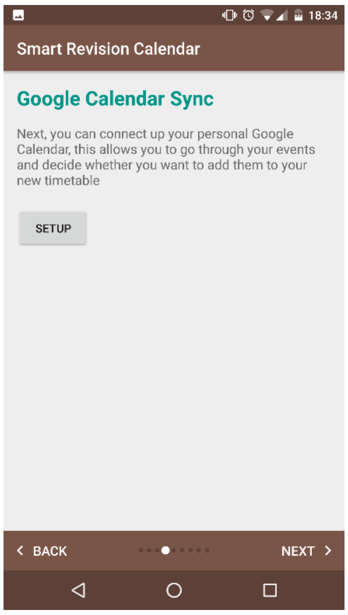
< BACK NEXT >

Figure 9-16 Add Activity

Step 4 (Figure 9-17): clicking "Setup" button takes the user to the same screen used for syncing Google calendar during the app's normal operation, i.e. post start-up, see section 9.4.8.

Step 5 (Figure 9-18): explaining the cloud backup process.

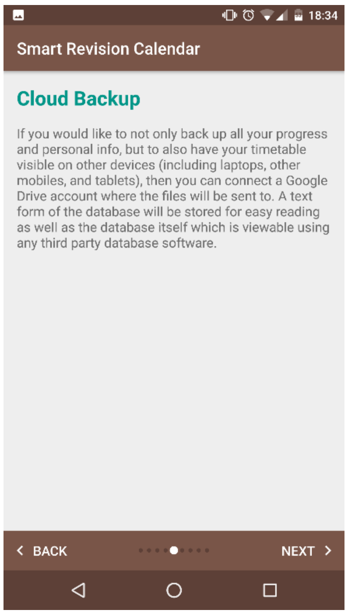
Step 6 (Figure 9-19): explaining features of the smart calendar and revision generation.



Google Calendar Sync

Next, you can connect up your personal Google Calendar, this allows you to go through your events and decide whether you want to add them to your new timetable

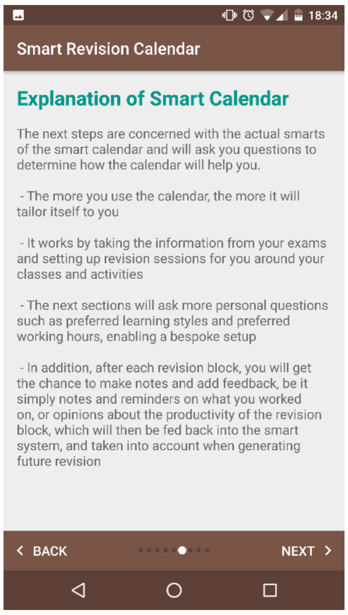
Figure 9-17 Google Calendar Sync



Cloud Backup

If you would like to not only back up all your progress and personal info, but to also have your timetable visible on other devices (including laptops, other mobiles, and tablets), then you can connect a Google Drive account where the files will be sent to. A text form of the database will be stored for easy reading as well as the database itself which is viewable using any third party database software.

Figure 9-18 Cloud Backup



Explanation of Smart Calendar

The next steps are concerned with the actual smarts of the smart calendar and will ask you questions to determine how the calendar will help you.

- The more you use the calendar, the more it will tailor itself to you

- It works by taking the information from your exams and setting up revision sessions for you around your classes and activities

- The next sections will ask more personal questions such as preferred learning styles and preferred working hours, enabling a bespoke setup

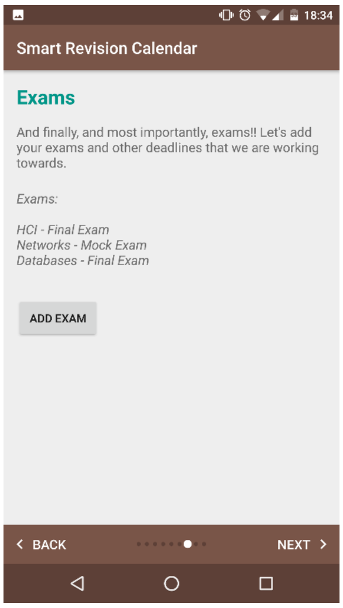
- In addition, after each revision block, you will get the chance to make notes and add feedback, be it simply notes and reminders on what you worked on, or opinions about the productivity of the revision block, which will then be fed back into the smart system, and taken into account when generating future revision

Figure 9-19 Smart Calendar Features

Step 7 (Figure 9-20): shows exams in the system, clicking "Add Exam" button takes the user to the same screen used for adding an exam during the app's normal operation, i.e. post start-up, see section 9.4.4.

Step 8 (Figure 9-21): all steps are scrolling layouts for when there is more content than the devices screen can fit, this learning style screen being an example of that. Clicking either times, brings up a TimePicker Dialog, of which is a built-in Android component, so should be familiar to the user. Clicking on either block length number, brings up a dialog with an EditText field and the numerical only keyboard. This numerical only keyboard only permits the user to enter numbers, thus satisfying one of Shneiderman's rules, "Offer error prevention and simple error handling" (Shneiderman et al., 2016). The variety is a slider where the value beneath is updated as the slider changes.

Step 9 (Figure 9-22): tick boxes to activate or deactivate debriefs. Pressing complete will finish this activity and return to the previous one, or start the main activity if on first boot. A toast message will be displayed congratulating the user that they have completed the start-up, once again fulfiling another golden rule by Shneiderman, "design dialogs to yield closure" (Shneiderman et al., 2016). A toast message is a small popup intended to provide feedback, which disappears after a timeout ("Toasts", 2017).



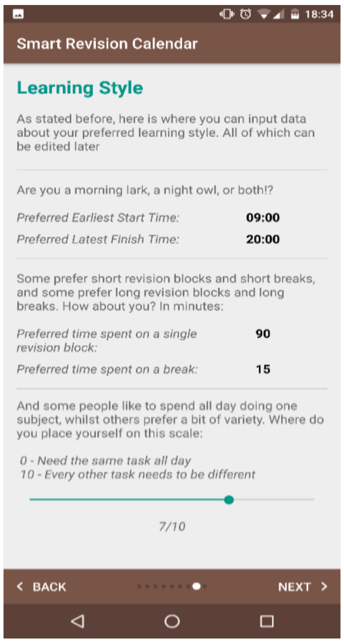
Exams

And finally, and most importantly, exams!! Let's add your exams and other deadlines that we are working towards.

Exams:

HCI - Final Exam
Networks - Mock Exam
Databases - Final Exam

Figure 9-20 Add Exam



Learning Style

As stated before, here is where you can input data about your preferred learning style. All of which can be edited later

Are you a morning lark, a night owl, or both!?

Preferred Earliest Start Time:	09:00
Preferred Latest Finish Time:	20:00

Some prefer short revision blocks and short breaks, and some prefer long revision blocks and long breaks. How about you? In minutes:

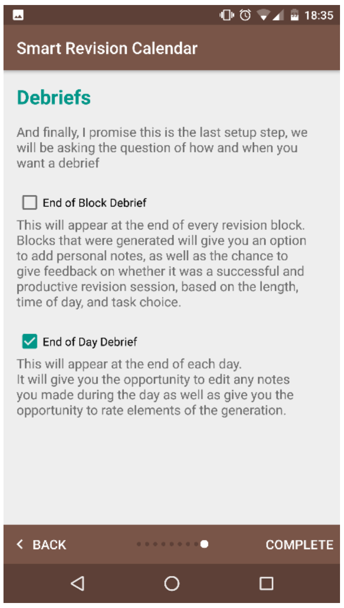
Preferred time spent on a single revision block:	90
Preferred time spent on a break:	15

And some people like to spend all day doing one subject, whilst others prefer a bit of variety. Where do you place yourself on this scale:

0 - Need the same task all day
10 - Every other task needs to be different

7/10

Figure 9-21 Learning Style



Debriefs

And finally, I promise this is the last setup step, we will be asking the question of how and when you want a debrief

☐ End of Block Debrief

This will appear at the end of every revision block. Blocks that were generated will give you an option to add personal notes, as well as the chance to give feedback on whether it was a successful and productive revision session, based on the length, time of day, and task choice.

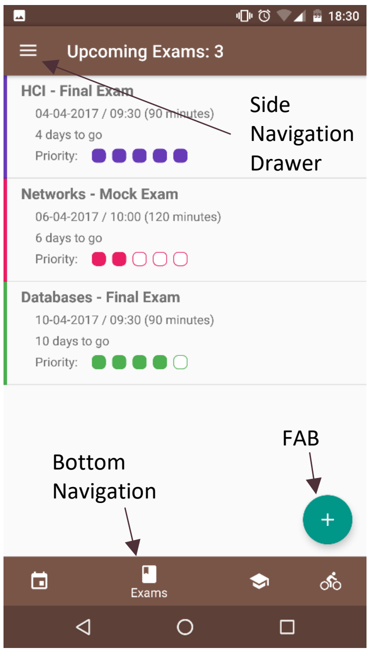
☑ End of Day Debrief

This will appear at the end of each day.
It will give you the opportunity to edit any notes you made during the day as well as give you the opportunity to rate elements of the generation.

Figure 9-22 Debriefs

9.4.2 Main Screens

The navigation is split into two core components, Bottom Navigation ("Bottom navigation", 2017) and Drawer Layout ("DrawerLayout", 2017). The bottom navigation bar allows the user to switch between top-level views in a single tap. Selecting a bottom navigation icon will take the user directly to the associated view or refreshes the currently active view. The side drawer layout is also a top-level navigation tool, however an interactive "drawer" is pulled out from the left side of the screen. In addition to these two navigational techniques, a Floating Action Button (FAB) is used. This is a special button that represents a primary or promoted action within an application ("Floating Action Button", 2017). Within this application, it is used to add a new class, activity, or exam, depending on which screen is currently being viewed. The icons for the timetable, exams, classes, activities, and add new, are a calendar, book, graduation hat, cyclists, and a plus respectively. These icons are similar to their operation, and hence provide a "match between the system and real world", of which satisfies one of Nielsen's Heuristics.



HCI - Final Exam
04-04-2017 / 09:30 (90 minutes)
4 days to go
Priority: ●●●●●

Networks - Mock Exam
06-04-2017 / 10:00 (120 minutes)
6 days to go
Priority: ●●○○○

Databases - Final Exam
10-04-2017 / 09:30 (90 minutes)
10 days to go
Priority: ●●●●○

Figure 9-23 Main Screen

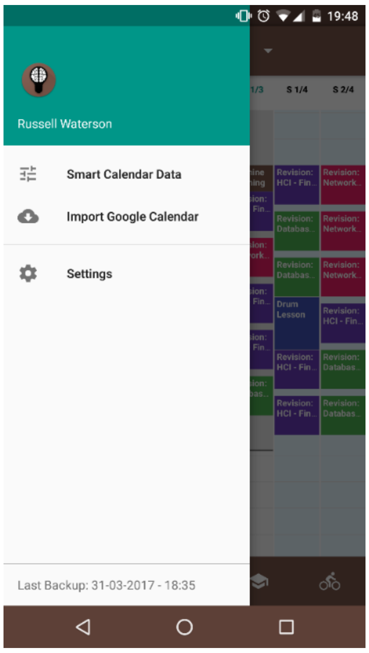


Figure 9-24 Side Navigation Drawer

9.4.3 Timetable Views

When adding a new class, activity, exam, or event, the user has the choice to select a colour for this item. This is so each item can better be distinguished from one another within the timetable views, as seen clearly Figures 9-25 to 9-27. Individual research conducted both by Pearson, and Halverson, conclude that colour can greatly increase effectiveness of visual search, especially in content rich pages (Halverson and Hornof, 2004; Pearson and van Schaik, 2003). In addition to using user defined colours, the familiar day, week, month view has been adopted, of which is used in the Google Calendar application. With between 100 and 500 million users on Android alone ("Google Inc.", 2017), the aim with implementation within this application, was to not only "seek universal usability", one of Shneiderman's rules (Shneiderman et al., 2016), by introducing transferable skills, but to also introduce "recognition rather than recall", one of Nielsen's heuristics (Nielsen, 1994), by minimising the user's memory load.

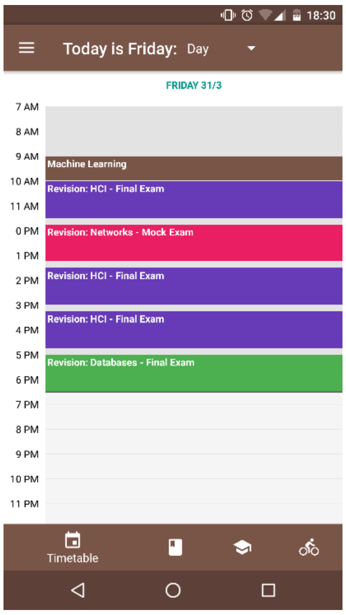

18:30
Today is Friday: Day
FRIDAY 31/3
Machine Learning
Revision: HCI - Final Exam
Revision: Networks - Mock Exam
Revision: HCI - Final Exam
Revision: HCI - Final Exam
Revision: Databases - Final Exam
Timetable

Figure 9-25 Day Timetable View

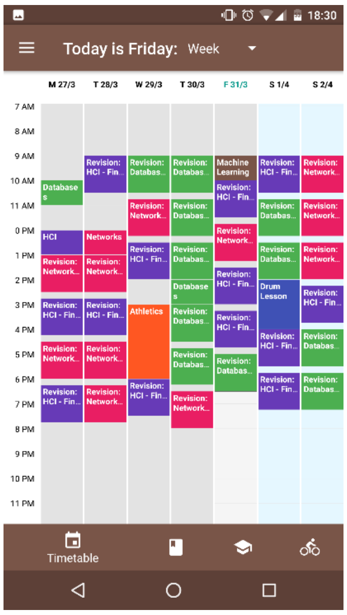


Figure 9-26 Week Timetable View

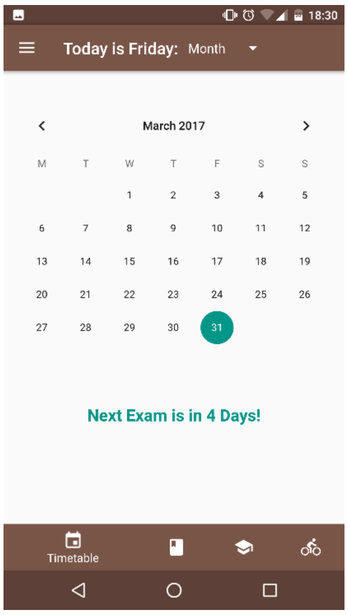


Figure 9-27 Month Timetable View

9.4.4 Database Queries

Adding class, activity, and exam data is done through their respective add screens, accessed through the floating action button from each fragment. Each field to be entered uses an appropriate Android component for inputting this data. For example, text fields will use a TextInputEditText component with appropriate capitalisation, numbers will also use a TextInputEditText component however with a numerical keyboard displayed, dates will use a DatePicker Dialog, times will use a TimePicker Dialog, days will use a Spinner dropdown list containing all the days, repeats will use a switch, and the colour will use a custom-built colour picker. When any of these inputs are invalid, a dialog will be displayed informing the user exactly which inputs need correcting.

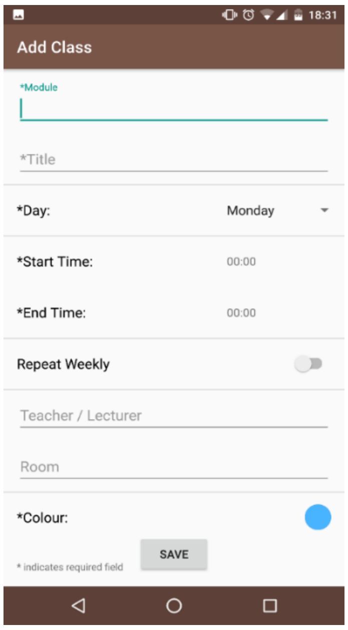


Figure 9-28 Add Class

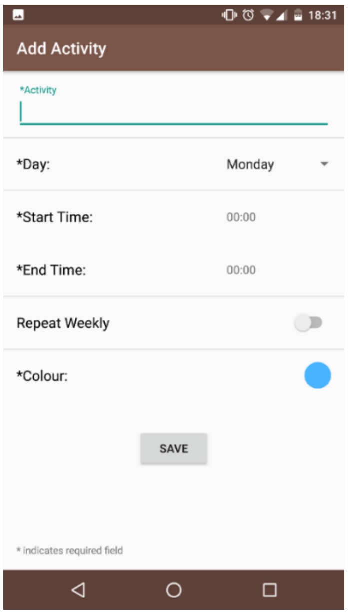


Figure 9-29 Add Activity

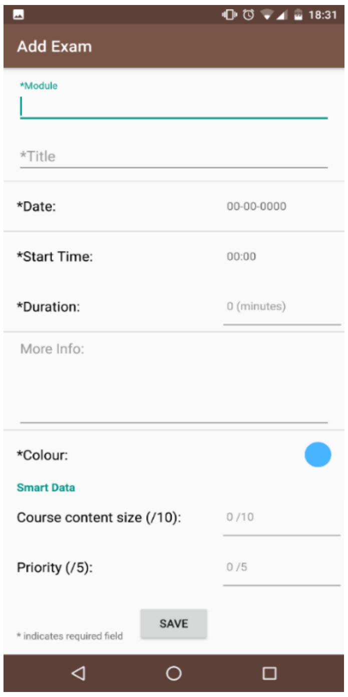
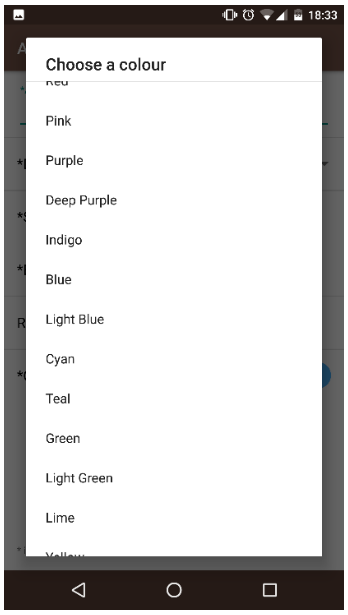


Figure 9-30 Add Exam

The custom-built colour picker is a list of all the colours in the system by name, as seen in Figure 9-31. Selecting one of these colours will change the colour view to the one selected showing how it will appear throughout the system. All colours consist only of level 500 Material Colours ("Color", 2017).

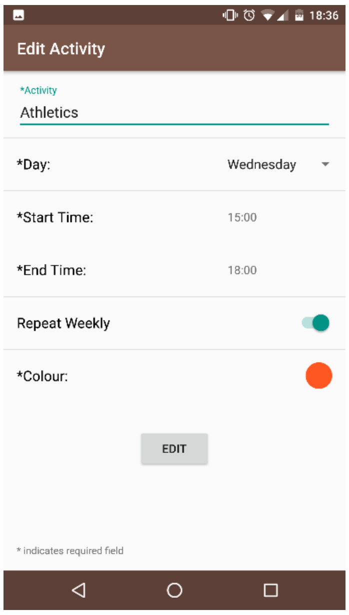
Editing class, activity, exam, or revision data is done through the same add screens from before. However, the title instead reads “Edit [item]” and all the data fields are populated with the previously existing data, as seen in Figure 9-32.


18:33

Choose a colour

Pink
Purple
Deep Purple
Indigo
Blue
Light Blue
Cyan
Teal
Green
Light Green
Lime

Figure 9-31 Custom Colour Picker


18:36

Edit Activity

*Activity
Athletics

*Day: Wednesday
*Start Time: 15:00
*End Time: 18:00
Repeat Weekly
*Colour:

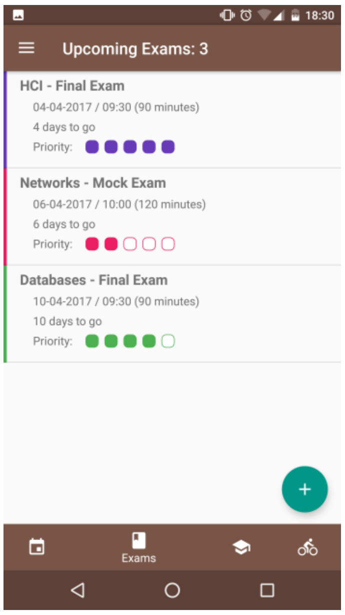
EDIT

* indicates required field

Figure 9-32 Edit Item Screen

Records can be viewed in a variety of ways in each database. Outside of the timetable view, the main two methods for viewing data, are the main fragments for class, activity, and exam, as seen in Figures 9-33, 34, and 35 respectively, and the individual item screen, as seen in Figure 9-36.

Each of the fragments uses a custom-built UI component for displaying each individual record, showing the majority of its data fields for each. This is done through text, a coloured line on the left representing the record’s colour, and a repeat icon if applicable. The exam component has a priority view whereby its value is represented as filled in counters in the same colour of that exam. This is as the priority of each exam, is a key component of the system, so this view stands out significantly more than a simple number as part of a text view. As the number of components for each screen is determined by the number of items in that particular table in the database, the UI is rendered programmatically rather than through XML.


18:30

Upcoming Exams: 3

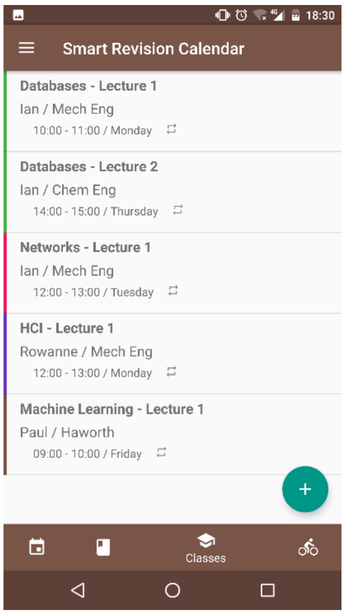
HCI - Final Exam
04-04-2017 / 09:30 (90 minutes)
4 days to go
Priority: ●●●●●

Networks - Mock Exam
06-04-2017 / 10:00 (120 minutes)
6 days to go
Priority: ●●○○○

Databases - Final Exam
10-04-2017 / 09:30 (90 minutes)
10 days to go
Priority: ●●●●○

Exams

Figure 9-33 Exams Screen


18:30

Smart Revision Calendar

Databases - Lecture 1
Ian / Mech Eng
10:00 - 11:00 / Monday

Databases - Lecture 2
Ian / Chem Eng
14:00 - 15:00 / Thursday

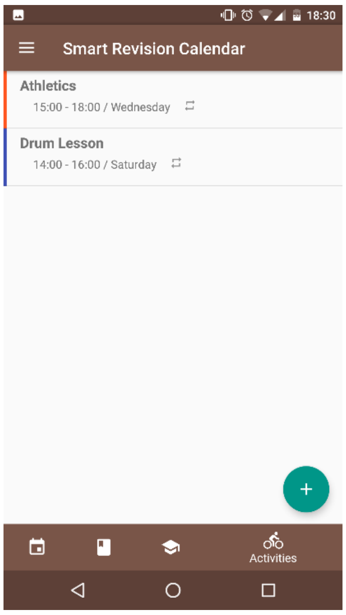
Networks - Lecture 1
Ian / Mech Eng
12:00 - 13:00 / Tuesday

HCI - Lecture 1
Rowanne / Mech Eng
12:00 - 13:00 / Monday

Machine Learning - Lecture 1
Paul / Haworth
09:00 - 10:00 / Friday

Classes

Figure 9-34 Classes Screen


18:30

Smart Revision Calendar

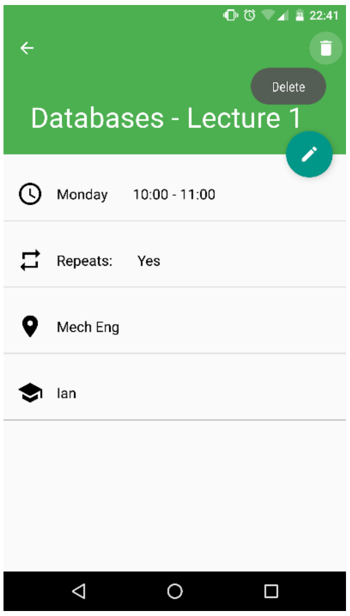
Athletics
15:00 - 18:00 / Wednesday

Drum Lesson
14:00 - 16:00 / Saturday

Activities

Figure 9-35 Activities Screen

The individual item screen uses the same template through displayed classes, activities, exams, revision, and events, however the data and icons displayed is different for each. It uses an expanding and collapsing toolbar, through the CoordinatorLayout which reacts to scrolling, of which is in the colour of that item. There is a floating action button for editing the item, which also reacts to scrolling, as well as the delete icon in the toolbar. This delete icon uses a tooltip, a text label containing brief helper text about its function, which in this case is activated by long pressing the icon ("Tooltips", 2017). This provides "help and documentation", one of Nielsen's Heuristics (Nielsen, 1994). The icons for the delete and edit options, are a bin and a pencil respectively, providing a "match between the system and real world", satisfying another one of Nielsen's Heuristics (Nielsen, 1994). The navigation bar is changed to black on this screen as to not overload the user with colour.


22:41

Delete

Databases - Lecture 1

Monday 10:00 - 11:00

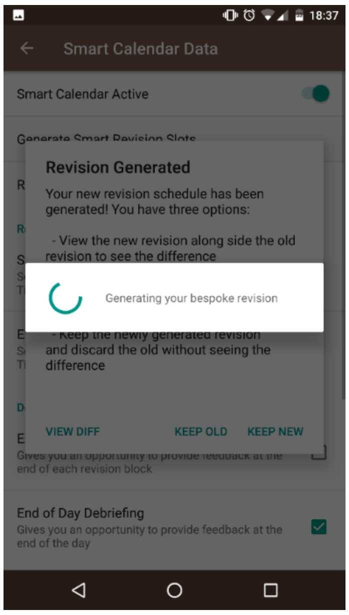
Repeats: Yes

Mech Eng

Ian

Figure 9-36 Individual Item Screen

When deleting classes, activities, exams, revision, or events, the system first asks the user whether they are sure they wish to complete this action, followed by a toast message explaining which course of action has taken place and its effect on the system. This implementation satisfies three of Shneiderman's rules, being "offer informative feedback," "design dialogs to yield closure," and "offer error prevention and simple error handling" (Shneiderman et al., 2016).


18:37

Smart Calendar Data

Smart Calendar Active

Generate Smart Revision Slots

Revision Generated

Your new revision schedule has been generated! You have three options:

- View the new revision along side the old revision to see the difference

Generating your bespoke revision

- Keep the newly generated revision and discard the old without seeing the difference

VIEW DIFF KEEP OLD KEEP NEW

Gives you an opportunity to provide feedback at the end of each revision block

End of Day Debriefing

Gives you an opportunity to provide feedback at the end of the day

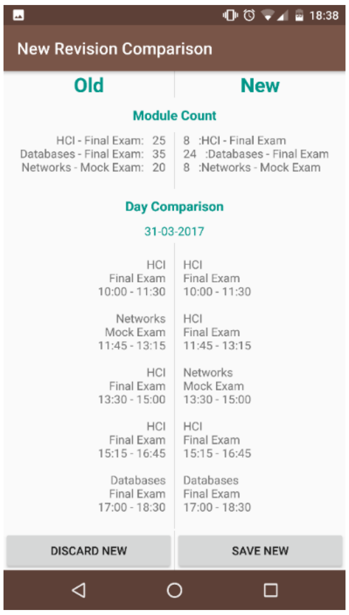
Figure 9-37 Revision Generation

9.4.5 Smart Data

The smart data is editable using two separate methods, one is through the initial start-up process as explained in section 9.4.1, and the other is through the smart calendar data settings explained in section 9.4.10.

As the smart algorithm is an intent service and is run in the background without a GUI, a decision was made to stop the user from making any actions on the main thread. This is done by displaying an uncancellable Progress Dialog informing the user of the actions taking place. This is ensuring "visibility of the system status," one of Nielsen's Heuristics (Nielsen, 1994).

The screen shown subsequent to revision being generated showing the difference between old and new revision, is an activity containing custom-built UI components in a scroll view. A custom component is used for each day of revision, where the date is displayed followed old revision on the left and new revision on the right, each containing what it is for and its time. Outside of these components and module counter, are static components that do not react with the scroll view. These are the column headers, and the buttons to save the chosen revision. Having them constantly in view keeps the user in control, thus "supporting an internal locus of control", one of Shneiderman's rules (Shneiderman et al., 2016).


18:38

New Revision Comparison

Old	New
Module Count	
HCI - Final Exam: 25	8 :HCI - Final Exam
Databases - Final Exam: 35	24 :Databases - Final Exam
Networks - Mock Exam: 20	8 :Networks - Mock Exam
Day Comparison	
31-03-2017	
HCI Final Exam 10:00 - 11:30	HCI Final Exam 10:00 - 11:30
Networks Mock Exam 11:45 - 13:15	HCI Final Exam 11:45 - 13:15
HCI Final Exam 13:30 - 15:00	Networks Mock Exam 13:30 - 15:00
HCI Final Exam 15:15 - 16:45	HCI Final Exam 15:15 - 16:45
Databases Final Exam 17:00 - 18:30	Databases Final Exam 17:00 - 18:30
DISCARD NEW	SAVE NEW

Figure 9-38 Revision Difference Screen

9.4.6 Debriefs

After the debrief notification has been clicked, the user is presented with an activity containing a RatingBar, a built-in Android component for selecting ratings in stars, and a series of Discrete SeekBars. The rating bar is used to rate the productivity of that day in a clear way, through the use of 5 stars. The sliders are used to rate individual components where the middle is perfect, the left is a less than desired action, and the right is a more than desired action, all of which is described at each step. Clicking the "Done" button adds the entered data to the Debriefs table in the database.

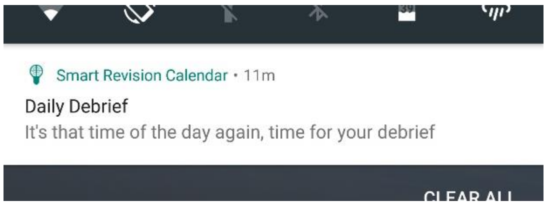
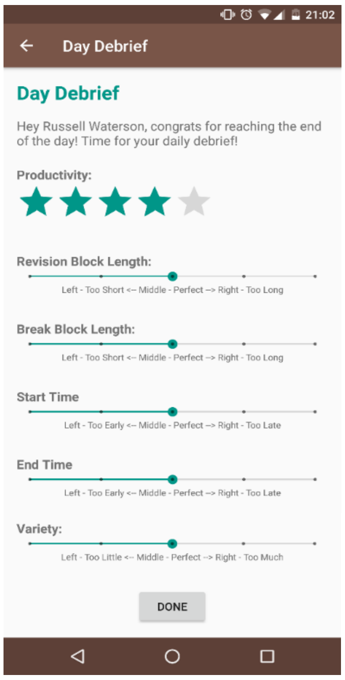

Smart Revision Calendar • 11m
Daily Debrief
It's that time of the day again, time for your debrief

Figure 9-39 Debrief Notification



Day Debrief
Hey Russell Waterson, congrats for reaching the end of the day! Time for your daily debrief!
Productivity:
Revision Block Length:
Left - Too Short <-- Middle - Perfect --> Right - Too Long
Break Block Length:
Left - Too Short <-- Middle - Perfect --> Right - Too Long
Start Time
Left - Too Early <-- Middle - Perfect --> Right - Too Late
End Time
Left - Too Early <-- Middle - Perfect --> Right - Too Late
Variety:
Left - Too Little <-- Middle - Perfect --> Right - Too Much

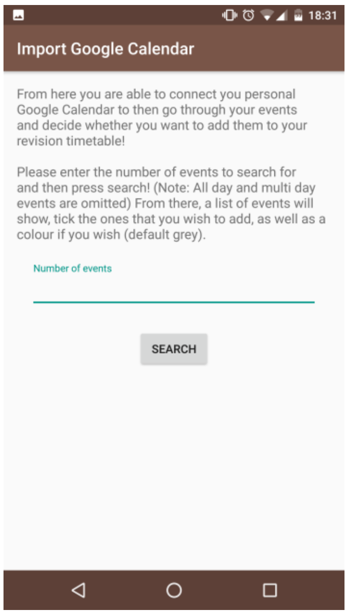
Figure 9-40 Day Debriefs Screen

9.4.7 Cloud Backups

Cloud backups are accessed through the side Drawer Layout from the main screens. It is a custom-made view at the bottom of the drawer containing the time and date of the previous backup, as seen in Figure 9-24. This "offers informative feedback" to the user, one of Shneiderman's rules (Shneiderman et al., 2016).

9.4.8 Import Google Calendar

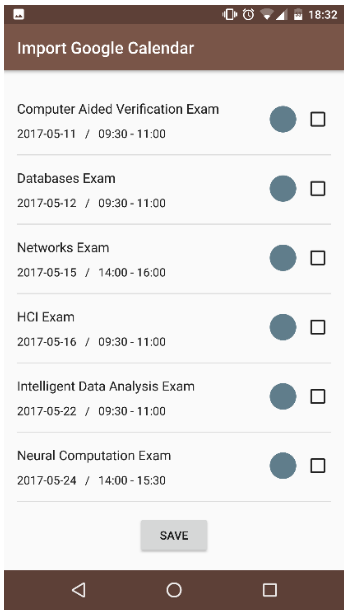
The custom-built UI component for displaying each event consists of: the same colour picker found in adding any other new item, a tick box, and text containing the event title, date, and start and end time. The user is able to select events they wish to add via the tick box, and then change those selected event's colour via the colour picker. Scrolling to the bottom of the page and clicking the "Save" button will add the selected items to the system.



From here you are able to connect you personal Google Calendar to then go through your events and decide whether you want to add them to your revision timetable!

Please enter the number of events to search for and then press search! (Note: All day and multi day events are omitted) From there, a list of events will show, tick the ones that you wish to add, as well as a colour if you wish (default grey).

Figure 9-41 Import Calendar Search



Computer Aided Verification Exam
2017-05-11 / 09:30 - 11:00
Databases Exam
2017-05-12 / 09:30 - 11:00
Networks Exam
2017-05-15 / 14:00 - 16:00
HCI Exam
2017-05-16 / 09:30 - 11:00
Intelligent Data Analysis Exam
2017-05-22 / 09:30 - 11:00
Neural Computation Exam
2017-05-24 / 14:00 - 15:30

Figure 9-42 Import Calendar Results

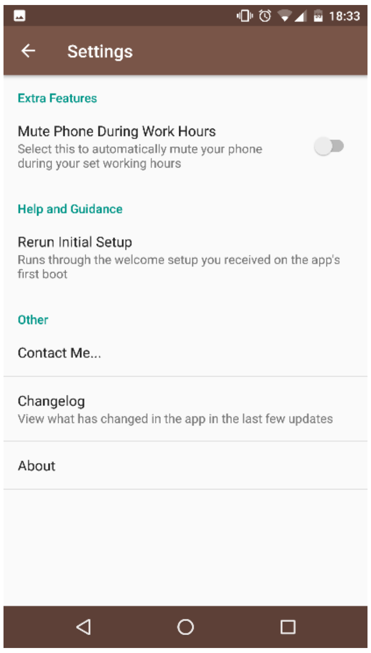
9.4.9 Mute Phone

The mute phone option is a simple toggle switch found from within the settings, as seen in section 9.4.10. The alarms previously described are set in the background, however a toast notification is displayed informing the user that this is taking place, "offering informative feedback".

9.4.10 Settings

The settings screen and the smart calendar data screen are both PreferenceFragments. Whilst PreferenceFragment provides a platform in which to access the shared preferences, the persisting data within a system, it also allows for the following of the visual style of the system preferences ("PreferenceFragment", 2017). This therefore follows the "consistency and standards" of all settings screens throughout the majority of Android applications, satisfying a Nielsen Heuristic (Nielsen, 1994).

Prior to the discovery of PreferenceFragments, the GUI for the Smart Calendar Data screen had been created based on the mock-ups in the sketches and the wireframes, as seen in Figure 9-45. After PreferenceFragments had been implemented, as seen in Figure 9-44, the user is given more useful information about what each entry is and does.

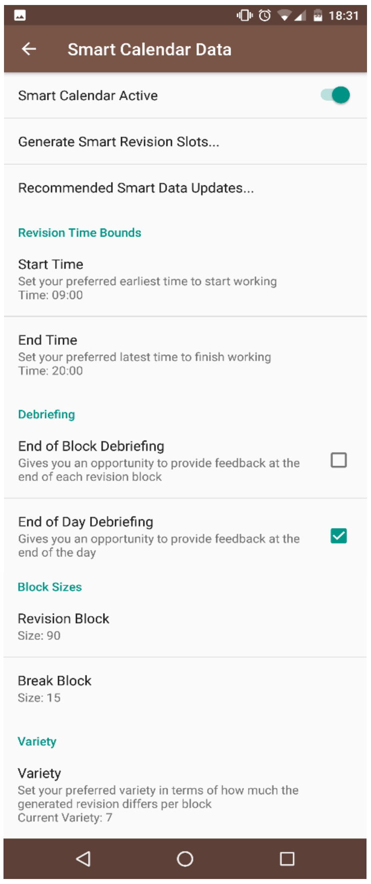

18:33
Settings

Extra Features
Mute Phone During Work Hours
Select this to automatically mute your phone during your set working hours

Help and Guidance
Rerun Initial Setup
Runs through the welcome setup you received on the app's first boot

Other
Contact Me...
Changelog
View what has changed in the app in the last few updates
About

Figure 9-43 Settings Screen


18:31
Smart Calendar Data

Smart Calendar Active
Generate Smart Revision Slots...
Recommended Smart Data Updates...

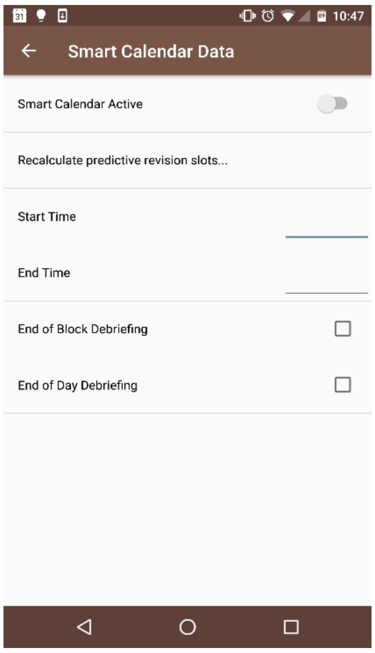
Revision Time Bounds
Start Time
Set your preferred earliest time to start working
Time: 09:00
End Time
Set your preferred latest time to finish working
Time: 20:00

Debriefing
End of Block Debriefing
Gives you an opportunity to provide feedback at the end of each revision block
End of Day Debriefing
Gives you an opportunity to provide feedback at the end of the day

Block Sizes
Revision Block
Size: 90
Break Block
Size: 15

Variety
Variety
Set your preferred variety in terms of how much the generated revision differs per block
Current Variety: 7

Figure 9-44 Smart Calendar Data Screen


10:47
Smart Calendar Data

Smart Calendar Active
Recalculate predictive revision slots...
Start Time
End Time
End of Block Debriefing
End of Day Debriefing

Figure 9-45 Old Calendar Data Screen

9.5 App Icon

The icon for the application is a light bulb with a brain inside, as seen in Figure 9-46 and Figure 9-47. The light bulb represents the sparking of an idea, as well as the synonym of being switched on. The brain is representative of the smart algorithm in place to generate revision.

The focus image is adapted from “Brain Light Bulb” (GDJ, 2016), which was then further edited using the Android Asset Studio (Nurik , 2017). The colour of the icon is the same of that used throughout the application, Material Brown #795548, keeping the consistency, as well as having that recognisable colour. Both a square and a circle app icon were created in order to have compatibility with Android Nougat 7.1 round app icons ("Round Icon Resources", 2017). In addition to all this, both the square and round icons were generated in multiple sizes to render efficiently on a wide variety of screen densities ("Size and Format", 2017).

In addition to the app icon, the notification and status bar icon had to also be created. This notification icon is required to only consist of white pixels on a transparent backdrop ("Status Bar Icons", 2017). This was achieved by taking the same light bulb brain image, and editing it using Paint.net, a free image and photo editing software. The colours were inverted to white along with the required transparent elements.

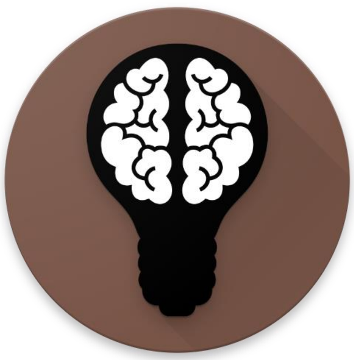


Figure 9-46 Round App Icon

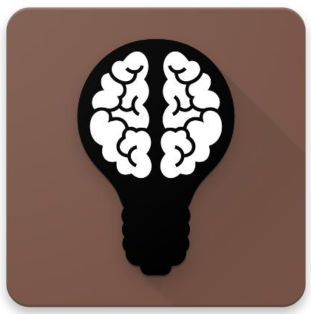


Figure 9-47 Original Square App Icon

10 TESTING

Throughout the implementation of the application, as well as upon completion, an extensive amount of testing was carried out to ensure the highest quality code has been developed, as well as determining whether it satisfied all of the detailed requirements.

10.1 Unit Testing

Unit testing is a white box testing technique whereby each unit, being the smallest testable part of the application, is independently inspected for its precise and expected operation (Rouse, 2017). Unit tests are commonly automated, however a large proportion of the unit tests carried out within this project were completed manually. An extensive list of all the unit tests completed can be found in Appendix D.

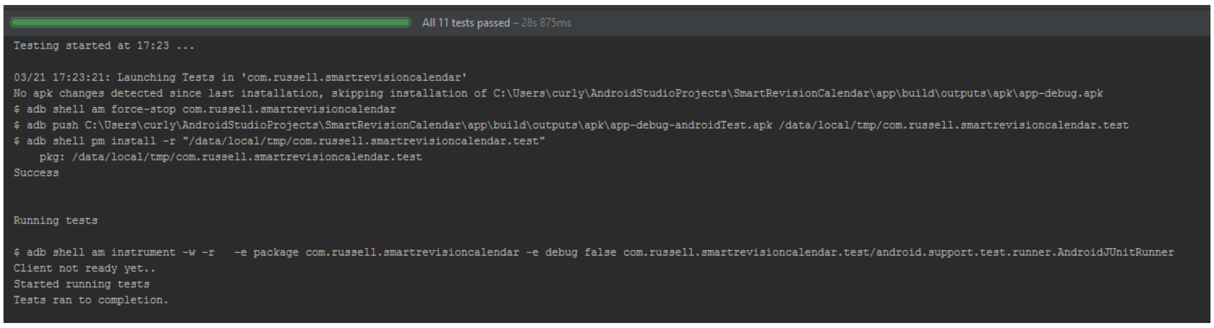
10.2 Instrument Espresso Testing

Instrumented tests are a type of unit test that is written with support for the Android framework APIs, as components are required in the testing, and are therefore ran on a physical device or emulator ("Building Instrumented", 2017). The Espresso testing framework is a tool for writing UI tests and simulating user interactions within the application ("Testing UI", 2017). The instrumented unit tests are written as a JUnit 4 test class using assertions and annotations.

These tests are located within the …/src/androidTest/… directory as outlined in Appendix A. The following instrumented tests were written and carried out, including an 11th dummy test case:

1.	Add Activity –	checks it exists when returning to the activity fragment
2.	Add Class –	checks it exists when returning to the class fragment
3.	Add Exam –	checks it exists when returning to the exam fragment
4.	Delete Activity –	checks the activity does not exist when returning to its fragment
5.	Delete Class –	checks the class does not exist when returning to its fragment
6.	Delete Exam –	checks the exam does not exist when returning to its fragment
7.	First start up –	checks all the steppers during the initial start-up are present
8.	Add Empty Activity –	checks all the validation is in place when entering an empty activity
9.	Add Empty Class –	checks all the validation is in place when entering an empty class
10.	Add Empty Exam –	checks all the validation is in place when entering an empty exam

All above tests passed when ran on a Nexus 6P emulator running Android 6.0, as seen in Figure 10-1.



```
Testing started at 17:23 ...

03/21 17:23:21: Launching Tests in 'com.russell.smartrevisioncalendar'
No apk changes detected since last installation, skipping installation of C:\Users\curly\AndroidStudioProjects\SmartRevisionCalendar\app\build\outputs\apk\app-debug.apk
$ adb shell am force-stop com.russell.smartrevisioncalendar
$ adb push C:\Users\curly\AndroidStudioProjects\SmartRevisionCalendar\app\build\outputs\apk\app-debug-androidTest.apk /data/local/tmp/com.russell.smartrevisioncalendar.test
$ adb shell pm install -r "/data/local/tmp/com.russell.smartrevisioncalendar.test"
    pkg: /data/local/tmp/com.russell.smartrevisioncalendar.test
Success


Running tests

$ adb shell am instrument -w -r   -e package com.russell.smartrevisioncalendar -e debug false com.russell.smartrevisioncalendar.test/android.support.test.runner.AndroidJUnitRunner
Client not ready yet..
Started running tests
Tests ran to completion.
```

Figure 10-1 Espresso Instrument Test Log

10.3 Functional Testing

Functional testing is a testing technique that is used to test the functionality of the application against the functional requirements and specifications previously documented ("Functional Testing", 2016). It is a form of black box testing whereby the internal logic of the application is not known by the tester, thus better simulating actual real world usage. The functional testing carried out for this application is documented below, with the intention of covering all functional requirements previously detailed in section 6.3.

ID	Name	Description	Initial System State	Input	Expected Output	Result
1	Run through Initial Boot	Upon the first boot of the app, the initial setup is ran	Phone home screen	Click on the Smart Revision Calendar app icon, and enter the relevant data for each screen hitting next when done until complete	The app will launch and as it is the first boot it will show the initial setup, containing screens for welcome, add classes, add activities, google calendar sync, cloud backup, smart calendar explanation, add exams, learning style, and debriefs. When complete, the system will show a dialog explaining how to navigate the application.	Pass 08/03/17
2	Add Empty Class	Attempt to add a new class into the system contain all empty fields	App main screen	Click the class tab, click to add class, no data is entered, click "add"	The system responds by informing the user of all the incorrectly entered fields: module, title, start and end time.	Pass 08/03/17
3	Add Real Class	Add a new class into the system using real data	App main screen	Click the class tab, click to add class, enter real data to all data fields, click "add"	The system accepts the data and adds it to the class database. It then returns to the class tab.	Pass 08/03/17
4	View Class	View the previously entered class in the class tab and view it as an individual item	App main screen	Click the class tab, click the previously entered class	The system shows a list of classes in the DB, one of which being the previously entered class, click on that class and the system shows all its data on an individual screen.	Pass 08/03/17
5	Edit Class to Erroneous Data	Attempt to edit the previously entered class by entering all empty fields	App main screen	Click the class tab, click the previously entered class, click the "edit" floating action button, delete all data, click "edit"	Once edit has been clicked, the system shows the original add class screen but with populated fields. Entering all empty fields will show the same error message as before informing the user of all the incorrectly entered	Pass 08/03/17

					fields: module, title, start and end time.	
6	Edit Class	Edit the previously entered class using real data	App main screen	Click the class tab, click the previously entered class, click the “edit” floating action button, enter real data to all data fields, click “edit”	Once edit has been clicked, the system shows the original add class screen but with populated fields. The system accepts the data and adds it to the class database. It then returns to the class tab.	Pass 08/03/17
7	Delete Class	Delete the previously entered class	App main screen	Click the class tab, click the previously entered class, click delete and accept the dialog	The system deletes the class from the class database.	Pass 08/03/17
8	Add Empty Activity	Attempt to add a new activity into the system contain all empty fields	App main screen	Click the activity tab, click to add activity, no data is entered, click “add”	The system responds by informing the user of all the incorrectly entered fields: activity name, start and end time	Pass 08/03/17
9	Add Real Activity	Add a new activity into the system using real data	App main screen	Click the activity tab, click to add activity, enter real data to all data fields, click “add”	The system accepts the data and adds it to the activity database. It then returns to the activity tab.	Pass 08/03/17
10	View Activity	View the previously entered activity in the activities tab and view it as an individual item	App main screen	Click the activity tab, click the previously entered activity	The system shows a list of activities in the DB, one of which being the previously entered activity, click on that activity and the system shows all its data on an individual screen.	Pass 08/03/17
11	Edit Activity to Erroneous Data	Attempt to edit the previously entered activity by entering all empty fields	App main screen	Click the activity tab, click the previously entered activity, click the “edit” floating action button, delete	Once edit has been clicked, the system shows the original add activity screen but with populated fields. Entering all empty fields will show the same error message as before informing the user of all the incorrectly entered	Pass 08/03/17

				all data, click "edit"	fields: activity name, start and end time.	
12	Edit Activity	Edit the previously entered activity using real data	App main screen	Click the activity tab, click the previously entered activity, click the "edit" floating action button, enter real data to all data fields, click "edit"	Once edit has been clicked, the system shows the original add activity screen but with populated fields. The system accepts the data and adds it to the activities database. It then returns to the activity tab.	Pass 08/03/17
13	Delete Activity	Delete the previously entered activity	App main screen	Click the activity tab, click the previously entered activity, click delete and accept the dialog	The system deletes the activity from the activity database.	Pass 08/03/17
14	Add Empty Exam	Attempt to add a new exam into the system contain all empty fields	App main screen	Click the exam tab, click to add exam, no data is entered, click "add"	The system responds by informing the user of all the incorrectly entered fields: module, title, date, duration. And the content size and priority will be set to a default of 0.	Pass 08/03/17
15	Add Real Exam	Add a new exam into the system using real data	App main screen	Click the exam tab, click to add exam, enter real data to all data fields, click "add"	The system accepts the data and adds it to the exam database. It then returns to the exam tab.	Pass 08/03/17
16	View Exam	View the previously entered exam in the exams tab and view it as an individual item	App main screen	Click the exam tab, click the previously entered exam	The system shows a list of exams in the DB, one of which being the previously entered exam, click on that exam and the system shows all its data on an individual screen.	Pass 08/03/17
17	Edit Exam to Erroneous Data	Attempt to edit the previously entered exam by entering all empty fields	App main screen	Click the exam tab, click the previously entered exam, click the "edit" floating action button, delete	Once edit has been clicked, the system shows the original add exam screen but with populated fields. Entering all empty fields will show the same error message as before	Pass 08/03/17

				all data, click "edit"	informing the user of all the incorrectly entered fields: module, title, date, duration. As well as setting the content size and priority to a default of 0.	
18	Edit Exam	Edit the previously entered exam using real data	App main screen	Click the exam tab, click the previously entered exam, click the "edit" floating action button, enter real data to all data fields, click "edit"	Once edit has been clicked, the system shows the original add exam screen but with populated fields. The system accepts the data and adds it to the exams database. It then returns to the exam tab.	Pass 08/03/17
19	Delete Exam	Delete the previously entered exam	App main screen	Click the exam tab, click the previously entered exam, click delete and accept the dialog	The system deletes the exam from the exam database.	Pass 08/03/17
20	View Daily Time-table	View the daily timetable in the timetables tab to see all the entered items in each database for an individual day	App main screen	Ensure that data is entered in each of the databases, Click the timetable tab, click on day view from the dropdown	The system will show the current day along with all the classes, activities, exams, and revision for that day. The user can scroll up and down to change time, left and right to change day, and can zoom in and out by using a pinching gesture. Individual items can also be clicked to show all their info in a separate screen.	Pass 08/03/17
21	View Weekly Time-table	View the weekly timetable in the timetables tab to see all the entered items in each database for a particular week	App main screen	Ensure that data is entered in each of the databases, Click the timetable tab, click on week view from the dropdown	The system will show the current week along with all the classes, activities, exams, and revision during that week. The user can scroll up and down to change time, left and right to change weeks, and can zoom in and out by using a pinching gesture. Individual items can also be clicked to show all their info in a separate screen.	Pass 09/03/17

22	View Monthly Time-table	View the monthly timetable in the timetables tab to see a month by month view	App main screen	Click the timetable tab, click on month view from the dropdown	The system will show the current month. The user can scroll left and right to change months.	Pass 08/03/17
23	Manually Change Smart Data to Valid Data	Change previously entered smart data	App main screen	Click the smart calendar data screen from the side navigation bar, change the start and end time, revision and break block size, and variety	The values saved in the system will change to the entered values, which will be taken into account when recalculating revision slots	Pass 08/03/17
24	End of Block Debrief	Fill in the end of block debrief form	Any screen on the phone, even out-side the app	Click the notification when displayed, complete the data form presented, click done	At the end of a revision block, a notification will be displayed on the user's phone. Clicking the notification will display the end of block debrief form. Entering data and pressing done will save the data into the debrief database	Fail – end of block debriefs has not been impleme nted due to time constrain ts
25	Daily Debrief	Fill in the daily debrief form	Any screen on the phone, even out-side the app	Click the notification when displayed, complete the data form presented, click done	At the end of a revision day, a notification will be displayed on the user's phone. Clicking the notification will display the daily debrief form. Entering data and pressing done will save the data into the debrief database	Pass 08/03/17
26	Recomm end Smart Data Updates	After a number of daily debriefs, select to recommend start data updates, and accept the changes	App main screen	Complete a number of daily debriefs, click the smart calendar data screen from the side navigation bar, choose to recommend smart data updates, accept the	A dialog will be displayed showing a new revision and break length, new start and end times, and a new variety. Choosing yes will change the smart data to the displayed options, which will then be taken into account when recalculating revision slots	Pass 08/03/17

				proposed changes		
27	Calculate Smart Revision Slots for the first time	The system will calculate revision slots based on the user's entered exams and learning style	App main screen	Click the smart calendar data screen from the side navigation bar, choose to recalculate revision slots	Provided the smart features are switched on, the system will generate revision slots and populate the revision database based on existing items and the user's learning style, all whilst a progress dialog is displayed. The dialog is presented to explain to the user that the revision has been created and that subsequent generation will get the opportunity to compare with old data.	Pass 14/03/17
28	Recalculate Smart Revision Slots and Compare	The smart revision slots previously allocated will be recalculated and entered into the system	App main screen	Click the smart calendar data screen from the side navigation bar, choose to recalculate revision slots	As above, but after generation, a dialog is presented giving the user the option to keep the old, keep the new, or compare the old and new. Comparing the two will show a list of all the old and new revision side by side on a per day basis. In addition the number of revision slots for each module is compared.	Pass 12/03/17
29	View Revision	View a generated revision from the timetable tab and view it as an individual item	App main screen	Click the timetable tab, click a previously generated revision	The system shows all the revision's data on an individual screen.	Pass 08/03/17
30	Edit Revision to Erroneous Data	Attempt to edit a previously generated revision by entering all empty fields	App main screen	Click the timetable tab, click a previously generated revision, click the "edit" floating action button, delete all data, click "edit"	Once edit has been clicked, the system shows the add revision screen with populated fields. Entering all empty fields will show an error message informing the user of all the incorrectly entered fields: module, title, date, start and end time.	Pass 08/03/17

31	Edit Revision	Edit a previously generated revision using real data	App main screen	Click the timetable tab, click a previously generated revision, click the "edit" floating action button, enter real data to all data fields, click "edit"	Once edit has been clicked, the system shows the add revision screen with populated fields. The system accepts the data and adds it to the revision database. It then returns to the timetable tab.	Pass 08/03/17
32	Delete Revision	Delete a previously generated revision	App main screen	Click the timetable tab, click a previously generated revision, click delete and accept the dialog	The system deletes the revision from the revision database.	Pass 08/03/17
33	Backup Database Text File to Google Drive	After entering data into each of the databases, back them up to Google Drive in the form of a text file	App main screen	Click the last backup button from the bottom of the side navigation bar, click yes to backup text file	If it is the first backup, the system will ask the user which Google Drive account they wish to back up to, they are then presented with the ability to backup a generated text file of the database to Drive, clicking yes will save this file to their selected location in Google Drive	Pass 08/03/17
34	Backup Raw Database File to Google Drive	After entering data into each of the databases, back them up to Google Drive in the raw database form	App main screen	Click the last backup button from the bottom of the side navigation bar, click yes or no to backup text file, then click yes to backup the database file	If it is the first backup, the system will ask the user which Google Drive account they wish to back up to, after clicking through the text backup, they are presented with the option to backup the raw database file saved on the phone's memory, clicking yes will save this file to their selected location in Google Drive	Pass 08/03/17
35	Import Event from Google Calendar	Connect app to Google Calendar and import an event from	App Main Screen	Click the import Google Calendar screen from the side navigation bar,	When the application is connected to the user's Google Calendar, a list of existing events will be displayed. The user can then select all the events	Pass 18/03/17

		existing calendar		enter the number of events, tick events, click import	they wish to add. When save is clicked, all the chosen events will be added to the local events database	
36	Activate Phone Muting	Activate phone muting so that during work hours, the phone will mute automatically	Any screen on the phone, even out-side of app	Click the settings screen from the side navigation bar, activate Mute Phone During Work Hours	When active, the mobile device will automatically set its ring profile to silent at the user's preferred start time, and then to normal at the user's preferred end time	Pass 14/03/17

10.4 UI and Application Exerciser Monkey

The Monkey is a program that runs on an Android device designed to stress-test an application in development by generating a pseudo-random stream of user events, such as clicks, touches, and gestures, as well as several system-level events, allowing for testing in a random, yet repeatable manner ("UI/Application Exerciser", 2017). During the running of Monkey, it watches the system under tests and will stop generating events if either the application crashes or becomes unresponsive, or if it receives any sort of unhandled exception.

Monkey is run from the command line, where the application package to be tested can be specified, as well as the number of events to be injected. The Smart Revision Calendar application was tested with 50,000 events injected into it using the command as seen below. Throughout the total elapsed time of 228627ms (3 minutes, 48.627 seconds), the application did not crash, become unresponsive, or throw any unhandled exceptions, therefore passed the Monkey tests.

```
$ adb shell monkey -p com.russell.smartrevisioncalendar -v 50000
```

10.5 Performance Testing

In conjunction with the stress testing done in section 10.4, performance testing was also carried out, and therefore complying with the performance non-functional requirements laid out in section 6.4. Performance testing is a testing technique performed to examine the responsiveness and stability under varying workloads ("Performance Testing", 2017). The performance tests undertaken looked to observe the execution times, memory usage, and CPU stress, during normal operations and intensive operations. This was done using the Android Monitor built into Android studio. Figure 10-2 shows a snippet of the memory and CPU usage during this time. It shows that there were no memory leaks, and never a point where the CPU usage was above 20%.

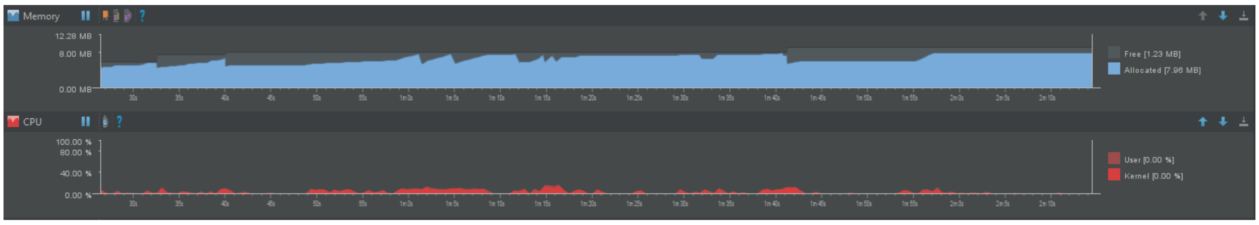


Figure 10-2 Android Monitor during Performance Test

10.6 Acceptance Testing – Alpha Testing

Acceptance testing is a testing technique that is used to determine the level of satisfaction against the requirements and verify it is acceptable for the end user ("Acceptance Testing", 2017). Of the various forms of acceptance testing, alpha tests were the ones carried out, of which consisted of two parts; the first was to carry out routine tasks expected in the day-to-day operation of the app, and the second was to try and break the app in whatever way possible. Whether this was through intentionally inputting erroneous data, or cancelling operations midway through, the aim was to bring to surface any bugs overlooked in the previous rounds of testing.

The routine tasks consisted of:

- Create at least 4 exams
- Create at least 4 classes
- Create at least 2 activities
- Enter smart revision data for your learning style (and activate daily debriefs)
- Generate revision
- After a couple of days of revision and debriefs check for recommendations
- Backup your data to Google Drive
- Now, try and break the application!

When reporting findings, this not only included the bugs found, but also any suggested improvements, what they liked, what they disliked, and what they expected to see but did not. The bugs that were found, required a log of the exact steps taken to trigger it, as well as the consequences of that bug. These were then all taken, and included in the bug reports filed from the other test stages, as explained further in section 10.7.

Another advantage for the alpha testing, was the ability to test the application on a wider range of devices, of which had different screen resolutions, Android versions, and custom original equipment manufacturer (OEM) skins. Included in the brief were questions asking for these details, from which it could be concluded that the application was confirmed working on a wide range of devices as seen in the table below:

Device	**Android Version**	**Screen Resolution**	**DPI and density**	**OEM Skin**	**Rooted?**
Huawei Nexus 6P	7.1.1	1440 x 2560	~518 / *xxhdpi*	Stock	No
LG Nexus 4	5.1	768 x 1280	~318 / *hdpi*	Stock	Yes
Huawei P9	7.0	1080 x 1920	~423 / *xhdpi*	Emotion UI 4.1	No
OnePlus One	6.0.1	1080 x 1920	~401 / *xhdpi*	CyanogenMod 13	No
Samsung Galaxy S6	7.0	1440 x 2560	~577 / *xxhdpi*	TouchWiz UI	No
Sony Xperia Z3 Compact	6.0	720 x 1280	~319 / *hdpi*	Sony Xperia Home	No

10.7 Bug Report

Over the course of the previously mentioned tests, a detailed and extensive Bug Report file was maintained. The report covers all the essential details required to reproduce the bug and eventually resolve it. With the addition of the ID and name, the fields detailed were:

- Priority: the more critical the bug, the higher the priority and therefore the quicker it was resolved
- Type: whether it was a bug or suggested feature or improvement
- Description: including details on how to reproduce the bug
- Found during: what means of testing the bug was unearthed
- Status: whether the bug had been set as resolved, infeasible, intended behaviour, or not reproducible, along with the date it was set

11 Project Management

11.1 Software Engineering Process

During the course of this project, a strict quality software engineering approach was taken. Once the functional requirements of the project had been established, they were split up into deliverable components and estimated development times assigned to each. This allowed a view into the total timescale of completion for the project's core functionality. As with all projects, contingency must be factored into the plan, as at any stage, something can go wrong, as detailed in section 6.5. Figure 11-1 shows the week by week plan created at the very start of the project. Whilst the eventual project did not strictly follow this timeline, as shown in the Gantt Chart in Appendix C, it was a very good indicator of roughly where the production should be at any given moment. The only major deviation from the original plan, was the swapping of intensive testing with the linking of the personal calendar via Google Calendar API, of which is not a fundamental feature of the application. This was mainly due to the fact that there was potential for a large number of issues to be brought to the surface during testing, as projected in section 6.5, and this allowed additional time to resolve them.

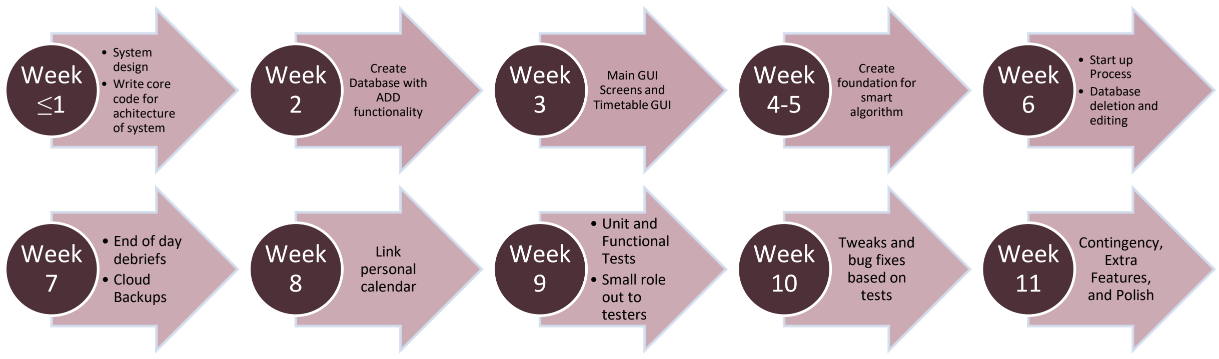

Week ≤1
• System design
• Write core code for achitecture of system
Week 2
Create Database with ADD functionality
Week 3
Main GUI Screens and Timetable GUI
Week 4-5
Create foundation for smart algorithm
Week 6
• Start up Process
• Database deletion and editing
Week 7
• End of day debriefs
• Cloud Backups
Week 8
Link personal calendar
Week 9
• Unit and Functional Tests
• Small role out to testers
Week 10
Tweaks and bug fixes based on tests
Week 11
Contingency, Extra Features, and Polish

Figure 11-1 Week Plan

Once this breakdown was in place, the agile development approach was undertaken, whereby the delivery of working software was continuous and frequent, thus meeting one of the twelve principles outlined in the agile manifesto. "Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale" – (Beck et al., 2001). Using agile, also allowed for an evolutionary development, where the planning was allowed to be adaptable enabling easy response to any change that may occur during this time. This was paramount, as later in development, it was decided that an option to view the difference in new and old revision was to be added, which would not have otherwise been possible.

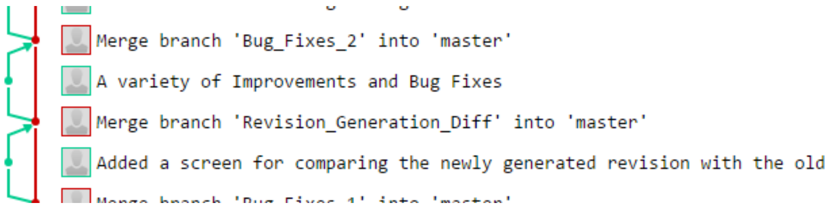
An alternative method of software development would have been to use a more traditional approach called waterfall. This is a sequential process where the stages of development follow on from each other, and include, but not exclusive to, plan, design, build, test, production, and maintain (Bowes, 2014). This method however was not undertaken due to the fact that it is non-iterative, meaning that any changes in the requirements would be very difficult to incorporate into the project once development had begun.

11.2 Version Control

As explained in section 7.5, Git was used a means of version control for the development of this application. A consistent process was used throughout the project to ensure that new code was kept stored safely on the remote repository, and no code was overwritten or lost.

As explained in section 11.1, agile development was used as a software engineering practise. When a new feature was to be implemented, a new branch was created, with a name similar to the functionality to be added. All the code for this feature was programmed on this new branch. At milestones of the feature's development, as well as at the conclusion of, the code was "committed" and "pushed" to the remote repository. A link to the Git repository can be found in within the supplied project zip file, as detailed in Appendix A. Once completion of the feature was believed to have been achieved, the pushed code on the branch would be submitted for a merge request. For the branch's merger into the master code to be accepted, it had to go through a code review, where it would be evaluated to see if it met the overall standards of the project. Once it has passed code review, the branch would be merged into master, and redundant branch deleted. All local repositories would then be updated with the latest code.

For example, Figure 11-2 shows a snippet from the Git graph of events moving in chronological order from bottom to top. It firstly shows code being pushed to a new branch, later identified as "Revision_Generation_Diff". An explanation of the code being added to that branch is given, ensuring that the exact purpose of that commit is known to anyone who views that branch. After this, and after the code review, the new branch is merged into master. The same thing applies for the next branches commit, push, and merge. Clicking on the summary of the push, will display the full explanation of the push, where for this "Bug_Fixes_2" branch, is an extensive list of fixes and improvements.


```
Merge branch 'Bug_Fixes_2' into 'master'
A variety of Improvements and Bug Fixes
Merge branch 'Revision_Generation_Diff' into 'master'
Added a screen for comparing the newly generated revision with the old
```
Figure 11-2 Git history graph snippet

Included within the Git repo was a readme file. To write an effective and well formatted readme file on Git, use of text-to-HTML conversion tool called Markdown is required (Gruber, 2004). The readme file included, of which utilised Markdown, briefly outlined the abstract of the project that was stored within the repository, and was therefore displayed on the projects home page and repository files page.

11.3 Supervisor Meets

Once a week, a supervisor meeting was set up, where the progress from the previous week was discussed as well as the aims for the coming week. In addition to this, weekly logs were written also containing the events and progress made in the preceding week.

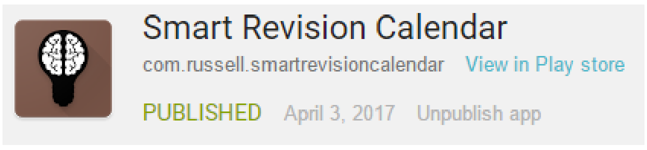
11.4 Project Issues

During the project's development, fortunately, there were no major setbacks encountered; there were however minor problems. These included time consuming tasks taking longer to complete than originally predicted, and a large number of bugs unearthed during the extensive testing. Fortunately, both of these eventualities had been accounted for. Firstly, contingency had been factored into the overall timeline, both before and after testing, enabling for any tasks that had overran to be properly completed to a high standard, without the worry of running out of time. Secondly, a significant proportion of time had been dedicated to bug fixes and improvements, therefore any bugs found during testing were able to be fixed before the application was pushed to production.

12 Results and Evaluation

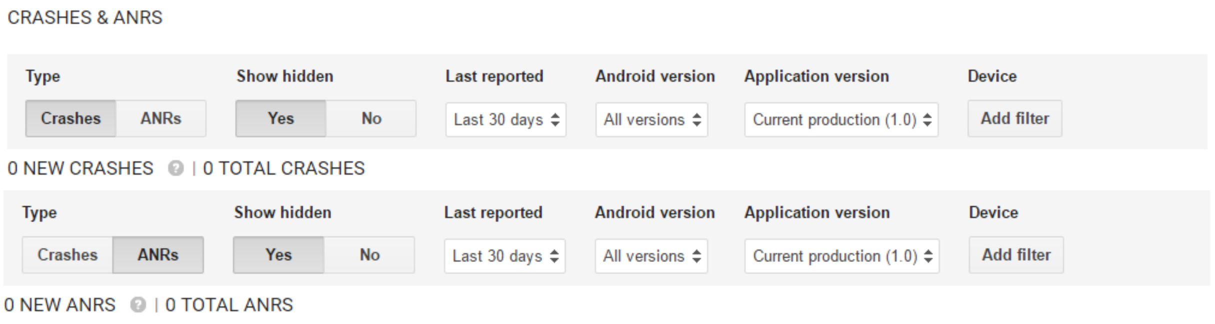
12.1 Public Release

After the completion of the testing and development of the application, it was decided that it would be uploaded to the Google Play Store, the official app store for Android. This was done by creating a release ready apk file signed with an official developer signature key. As a new key was used, the APIs had to be re-authorised. A developer account on the Play Store was then created, and the one-time payment of $25 was made. Uploaded alongside the application, was a description, a series of screenshots, a feature graphic, and a host of other information including content ratings and app category.


Smart Revision Calendar
com.russell.smartrevisioncalendar View in Play store
PUBLISHED April 3, 2017 Unpublish app

The Smart Revision Calendar application is currently released to the public via the Google Play Store; it can be viewed via the link in the supplied zip file as explained in Appendix A. Since its date of publication, on the 3rd April 2017, it has gained between 50-100 downloads, along with 9 ratings with an average score of 4.7 stars out of 5 *(as of 12th April 2017).*

Within the Google Play Developer Console, as well as being able to see number of downloads and user ratings, there is also the ability to observe an overall crash report of the application. At the time of writing, there have been no crashes, and no instances of the application not responding (ANR).


CRASHES & ANRS

Type	Show hidden	Last reported	Android version	Application version	Device
Crashes	Yes	Last 30 days	All versions	Current production (1.0)	Add filter

0 NEW CRASHES | 0 TOTAL CRASHES

Type	Show hidden	Last reported	Android version	Application version	Device
ANRs	Yes	Last 30 days	All versions	Current production (1.0)	Add filter

0 NEW ANRS | 0 TOTAL ANRS

The public release of the application looks to be an initial success, which should give it the momentum to continue to grow. Given the number of downloads and the positive ratings, the application appears to deliver on the goals initially set out.

12.2 Comparison to Original Specification

In addition to publishing the application, the project was to be evaluated; this includes the product *and* the process in which the product was achieved. The initial step for evaluation of the system, was the comparison of the completed solution with the original specification.

The evaluation of the product was done by looking to the proposed functional requirements, as outlined in section 6.3, and observing the functional tests in section 10.3, tests based solely on ensuring the requirements are met. All but one test passed, therefore indicating that the development of the product was a success, as was in addition the process as well. The reason for the failing test, was due mainly to the fact that the estimated time taken to implement said feature, would supersede the functionality's importance within the system, especially given that a similar feature had previously already been implemented.

The evaluation of the process, was carried out by looking to the project planning discussed in section 11.1, as well as the risk management discussed in section 6.5. This ensured that a clear structure and process would be undertaken throughout the course of this project. It was concluded that this plan was an unwavering success without a single oversight. The necessary precautions were made in terms of contingency planning and allowing for dedicated time for testing and bug fixing. As the development process stayed on track without any deviations, and was not over ambitious in terms of time constraints allowed, it meant that additional functionality outside of the initial core requirements could be implemented; this came in the form of mobile device muting during working hours. The agile development process worked well for the creation of this product, as it allowed for evolutionary requirements.

12.3 User Feedback

To further evaluate the product, a user inspection method was employed, requesting that a collection of 5 individuals perform a heuristic evaluation.

This evaluation consisted of 3 steps, first being a briefing session explaining to the individuals what to do. Of which, was over a period of 1-2 hours, they each work separately taking the one run through of the product to familiarise themselves, and the second run through to focus on specific features. Then, to create a list of usability problems and the feature that instigates it. After its completion, a debriefing session occurred in which the individuals come together in order to prioritise problems, where the severity of the problem is indicated by a combination of its frequency, impact, and persistence.

12.3.1 Reliability and Robustness

To determine the reliability of robustness of the product, both the feedback from the user evaluation and the results of the unit tests were assessed. Of the 307 unit tests undertaken, 305 of them passed with only 2 failures; that is a success rate of 99.3%. With the failing two tests being due to unimplemented features as opposed to unstable software. The feedback from the user evaluation indicates that the product is both reliable and robust, given the number of problems unearthed were minimal, and those that were, were trivial and of negligible severity. Further demonstrating this, is the positive response elicited with the public release of the application to the world.

12.3.2 Performance

To ascertain the performance of the product, the user evaluation and the performance tests of section 10.5 were reviewed. As stated in the previous section, the responses collected from the user evaluation were of resounding positivity, and as such, included that of the performance of the system. The performance tests undertaken and the conclusion of the development, also clearly demonstrate that the final product has no shortcomings in its performance, both in terms of speed of calculations, and in terms of responsiveness of all the UI elements.

12.3.3 Heuristic Evaluation

During the design of the product, Jakob Nielsen's 10 usability principles for interactive design were utilised to minimise any potential usability issues. Therefore, during heuristic evaluation of the product, also done during user evaluation, all 10 of the topics encompassed were addressed, despite being a rule of thumb rather than a specific guideline. Examples for each of the principles are as follows:

Visibility of system status: when generating revision, the system shows a progress dialog; during the initial start-up steppers, the system always displays the number of steps remaining.

Match between system and real world: all icons in the system represent a real-life counterpart, for example, the delete icon is a bin, and the edit icon is a pen.

User control and freedom: users often choose system functions by mistake, so every section of the system can be exited from without any repercussions.

Consistency and standards: the use of Material Design conforms to the standards across most major Android applications.

Error prevention: all inputs in the system are validated so erroneous data cannot be entered; when deleting an item, the user is asked if they are sure that they wish to complete the deletion.

Recognition rather than recall: minimising the user's memory load by using familiar interfaces such as those found in highly popular applications like Google Calendar.

Flexibility and efficiency of use: whilst the system does not provide any shortcuts for experienced users, the interface does however remain in a fixed formation allowing for the user to build a muscle memory induced impression of the system.

Aesthetic and minimalist design: the use of the meticulously crafted Material Design guidelines; the separation of key components of the system to ensure only the essential and relevant items are on any one screen at a time.

Help users recognise, diagnose and recover from errors: when a user makes a mistake, or the system experiences any unexpected behaviour, an alert dialog is displayed showing the user any errors that may have occurred.

Help and documentation: during the initial boot of the system, as well as at any time via the settings, the user is presented with instructions showing them how to operate the system.

12.4 Lessons Learned

Over the course of this project, there were many opportunities for learning and numerous chances to acquire new skills. These were not limited to coding experience, but also in terms of significant fields of computer science, as well as acceptable software engineering practices.

The skill of Android application development was one gained during the course of this process, whereby it is best learned through physical practise and application, rather than mere research. In addition, the previous brevity of which software engineering was covered, was allowed to be put into practise in a real-world system, thus solidifying the knowledge.

Additional skills acquired, but not limited to, include probabilistic algorithms, database management and manipulation, Google API handling, and UX processes including sketching, wireframes and Google's Material Design Guidelines.

13 Discussion

13.1 Achievements

At the completion of this project, the final product stands to be one of great success. Not only does it meet all the aims and requirements originally set out at the commencement of the project, but it also has been well received by a large quantity of the public. As of the 12th April 2017, Smart Revision Calendar stands at a 4.7 star rating with between 50-100 downloads, and a host of positive reviews. The application looks to have added value within its field, improving productivity and quality of work to student looking to revise for exams.

As for the process, a very regimented and well thought out strategy was constructed, and throughout it was followed meticulously, with the only deviations being those from perceptive foresight and planning.

13.2 Deficiencies

Whilst overall this application and its development were successful, containing a great number of triumphs, there is always room for improvement. In the product itself, there is the unfortunate lack of inclusion of the end of revision block debriefs, where the user is able to provide feedback and notes about the productivity of each individual revision block. As previously stated, its lack of inclusion was due mainly to that fact that its perceived value was less than the overhead of its estimated development time.

13.3 Further Development

Where the application currently stands, is at a fully completed and well-polished package that meets the requirements laid out prior its creation. However, it was developed in such a way, that future improvements and further development could be implemented. If there were 5 more years' worth of development time, here are some features that would be considered for implementation.

Term time: a feature that would allow for greater streamlining of term time work, including improved scheduling for homework, coursework, and other deadlines. Currently, whilst the app supports the addition of classes into the system, it currently lacks the ability to specify specific deadlines such as homework and coursework. The user is able to add them in as exams, and then vary the content size and priority, but this kind of work is not always carried out in the same way as revision. Adding this feature should allow the user to be able to continue using the application all year round, thus improving their productivity outside of the exam season too.

Class configuration sharing: a feature that would allow a user to create their timetable of classes, exams, and other items, and be able to share their created configuration with, say, a classmate who has the exact same modules as them. Thus further mitigating the timetable creation time and decreasing its complexity, alleviating the shortcomings of other timetable solutions, as explained in section 5.2**.**

Social aspects: a feature expanding deeper into class config sharing, is the ability to share progress as well as difficulties in specific tasks with friends. This should allow for an increased productivity due to the ease of access of targeted, and specific help and support.

Gamification: according to a study done by John Shindler in 2009, games and competition can "raise the level of interest and excitement while accomplishing essentially the same degree of content processing" – (Shindler, 2009). As such, a potential addition could be the development of a feature to incorporate some form of gamification into the application, whereby there are incentives and rewards for completing targets and set task, perhaps without any distractions from the user's phone. Digging deeper into the social aspects, being able to share these accolades, should further enhance the competition.

Smart watch features: according to a press release by the International Data Corporation, wearables are increasing in popularity, reaching an all-time high with 33.9 million units shipped in the fourth quarter of 2016, growing 16.9% year over year (Ubrani et al., 2017); including incorporating the use of smart watch features within the application is apparent. A simple addition so that when the user should be starting their next revision block, a push notification is sent to their smart watch. Additionally, this could be useful with users without a smart watch, as the same thing could happen but just to the phone instead.

Android O: as of 21st of March 2017, the developer preview has been released for the next version of Android, Android O (Burke, 2017). This gives developers the chance to develop applications to include specific features upcoming in the next release. Evaluating these features, there are two potential ones that could be incorporated within this application. First is the use of notification channels, this involves creating different channels for each distinct type of notification ("Notification Channels", 2017). For example, one channel could be used for debriefs, one for revision commencement reminders, and one for social features. The user will then have complete control over each channel including the notification's importance, sound, lights, vibration, whether it shows on the lock screen, and whether it overrides do not disturb. The second feature is the use of the multi-display support. This could be incorporated by allowing the user to view their timetable on a remote display or second display, allowing for a different perspective on their calendar, as well as the opportunity for collaborative work ("Multi-display support", 2017). These two features are just example of what could be included with the new version of Android, but with continued development, all future releases could be evaluated for interesting new features to be implemented.

Smart Formula: whilst the smart formula for creating revision is fully functional and operates and behaves exactly as intended, it can easily become an evolving part of the system, undergoing continuous enhancements as the system matures. Incorporating pattern matching for example, would be a great addition, whereby the sequence of generated revision could be checked against some pre-predicted pattern, potentially improving the consistency of the smart algorithm.

14 Conclusion

The main aim of this project has been to help increase the organisation and productivity of people, particularly students, through the use of an automated timetable creation based on their goals and learning styles. The system was designed and implemented to meet the goals of an ideal timetable and scheduling application, whilst including features to incorporate the individuality of learning styles. These goals were met, thus filling the void where there was previously no perfect solution for getting the most out of productivity using timetabled revision and learning.

The project has proven that it is possible to evaluate all the results from research and studies into the field of maximising productivity through various means, and create an all-in-one application to exploit the peak performances of the human mind.

This project represents an excellent opportunity to deliver a large software project as an individual and to learn the workings and implementations of Android application development as well as the workings of the mechanisms to increase productivity and performance.

15 References

OMG Unified Modeling Language (OMG UML). (2015) [Online]. Available from: http://www.omg.org/spec/UML/2.5/PDF/ [Accessed 04/05 2017].

Software Testing: An ISEB Intermediate Certificate. (2009) **Kybernetes,** 38 (9): 66-85.

"About SQLite" (2017) **About SQLite.** [Online]. Available from: http://www.sqlite.org/about.html [Accessed 04/06 2017].

"About TensorFlow" (2017) **About TensorFlow.** [Online]. Available from: https://www.tensorflow.org/ [Accessed 04/09 2017].

"Acceptance Testing" (2017) **Acceptance Testing.** [Online]. Available from: https://www.tutorialspoint.com/software_testing_dictionary/acceptance_testing.htm [Accessed 04/06 2017].

"ADT Plugin" (2015) **ADT Plugin (DEPRECATED).** [Online]. Available from: https://developer.android.com/studio/tools/sdk/eclipse-adt.html [Accessed 04/03 2017].

"Android Dashboards" (2017) **Home Android Dashboards.** [Online]. Available from: https://developer.android.com/about/dashboards/index.html [Accessed 04/05 2017].

"Android Studio System" (2017) **Android Studio System Requirements.** [Online]. Available from: https://developer.android.com/studio/index.html#Requirements [Accessed 04/04 2017].

"Ant Apache" (2017) **Apache Ant Apache.** [Online]. Available from: http://ant.apache.org/ [Accessed 04/06 2017].

"Apple Developer Program" (2017) **Apple Developer Program.** [Online]. Available from: https://developer.apple.com/programs/ [Accessed 04/04 2017].

"Application Fundamentals" (2017) **Develop API Guides Introduction Application Fundamentals.** [Online]. Available from: https://developer.android.com/guide/components/fundamentals.html [Accessed 04/04 2017].

"Bottom navigation" (2017) **Material Design Components Bottom navigation.** [Online]. Available from: https://material.io/guidelines/components/bottom-navigation.html [Accessed 04/08 2017].

"Building Instrumented" (2017) **Develop Training Best Practices for Testing Building Effective Unit Tests Building Instrumented Unit Tests.** [Online]. Available from: https://developer.android.com/training/testing/unit-testing/instrumented-unit-tests.html [Accessed 04/06 2017].

"Color" (2017) **Material Design Style Color.** [Online]. Available from: https://material.io/guidelines/style/color.html [Accessed 04/07 2017].

"Configure Your Build" (2017) **Configure Your Build.** [Online]. Available from: https://developer.android.com/studio/build/index.html [Accessed 04/06 2017].

"Cross-Platform Mobile" (2017) **Cross-Platform Mobile Development in Visual Studio.** [Online]. Available from: https://msdn.microsoft.com/library/dn771552.aspx [Accessed 04/03 2017].

"Data Storage Options" (2017) **Develop API Guides Data Storage Storage Options.** [Online]. Available from: https://developer.android.com/guide/topics/data/data-storage.html [Accessed 04/06 2017].

"DrawerLayout" (2017) **Developers Develop DrawerLayout.** [Online]. Available from: https://developer.android.com/reference/android/support/v4/widget/DrawerLayout.html [Accessed 04/08 2017].

"Expansion panels" (2017) **Material Design Components Expansion panels.** [Online]. Available from: https://material.io/guidelines/components/expansion-panels.html [Accessed 04/07 2017].

"Floating Action Button" (2017) **Mat erial Design Components Buttons: Floating Action Button.** [Online]. Available from: https://material.io/guidelines/components/buttons-floating-action-button.html [Accessed 04/08 2017].

"Functional Testing" (2016) **Functional Testing.** [Online]. Available from: http://softwaretestingfundamentals.com/functional-testing/ [Accessed 04/06 2017].

"Getting Started with" (2017) **Getting Started with the NDK.** [Online]. Available from: https://developer.android.com/ndk/guides/index.html [Accessed 04/04 2017].

"Getting Started" (2017) **Google Drive APIs Android Getting Started.** [Online]. Available from: https://developers.google.com/drive/android/get-started [Accessed 04/09 2017].

"Google Inc." (2017) **Google Inc Google Play Store Developer Page.** [Online]. Available from: https://play.google.com/store/apps/dev?id=5700313618786177705 [Accessed 04/07 2017].

"Google Play Developer Console" (2017) **Google Play Developer Console.** [Online]. Available from: https://play.google.com/apps/publish/ [Accessed 04/04 2017].

"Intel® Core™ i5-4690K" (2017) **Support Home Product Specifications Processors Intel® Core™ i5-4690K Processor.** [Online]. Available from: http://ark.intel.com/products/80811/Intel-Core-i5-4690K-Processor-6M-Cache-up-to-3_90-GHz [Accessed 04/04 2017].

"Introduction" (2017) **"Material design Introduction".** [Online]. Available from: https://material.io/guidelines/material-design/introduction.html [Accessed 04/07 2017].

"Java Language Environment" (2017) **The Java Language Environment.** [Online]. Available from: http://www.oracle.com/technetwork/java/intro-141325.html [Accessed 04/03 2017].

"Javadoc Tool" (2004) **Javadoc Tool.** [Online]. Available from: http://www.oracle.com/technetwork/java/javase/documentation/index-jsp-135444.html [Accessed 04/05 2017].

"javadoc" (2010) **javadoc - The Java API Documentation Generator.** [Online]. Available from: http://docs.oracle.com/javase/1.5.0/docs/tooldocs/solaris/javadoc.html [Accessed 04/05 2017].

"Layouts" (2017) **Develop API Guides User Interface Layouts.** [Online]. Available from: https://developer.android.com/guide/topics/ui/declaring-layout.html [Accessed 04/04 2017].

"Meet Android Studio" (2017) **Meet Android Studio.** [Online]. Available from: https://developer.android.com/studio/intro/index.html [Accessed 04/03 2017].

"Multi-display support" (2017) **Preview Android O Features and APIs Multi-display support.** [Online]. Available from: https://developer.android.com/preview/api-overview.html#mds [Accessed 04/08 2017].

"Nexus 6P Specifications" (2017) **Huawei Mobile Phones Nexus 6P Specifications.** [Online]. Available from: http://consumer.huawei.com/en/mobile-phones/nexus6p/specifications.htm [Accessed 04/04 2017].

"Notification Channels" (2017) **Develop Android O Features and APIs Notification Channels.** [Online]. Available from: https://developer.android.com/preview/features/notification-channels.html [Accessed 04/08 2017].

"OAuth 2.0" (2017) **OAuth 2.0.** [Online]. Available from: https://oauth.net/2/ [Accessed 04/09 2017].

"Performance Testing" (2017) **Performance Testing.** [Online]. Available from: https://www.tutorialspoint.com/software_testing_dictionary/performance_testing.htm [Accessed 04/07 2017].

"Platform Architecture" (2017) **Develop API Guides Platform Architecture.** [Online]. Available from: https://developer.android.com/guide/platform/index.html [Accessed 04/04 2017].

"PreferenceFragment" (2017) **PreferenceFragment.** [Online]. Available from: https://developer.android.com/reference/android/preference/PreferenceFragment.html [Accessed 04/09 2017].

"Round Icon Resources" (2017) **Android 7.1 for Developers Round Icon Resources.** [Online]. Available from: https://developer.android.com/about/versions/nougat/android-7.1.html#circular-icons [Accessed 04/03 2017].

"Run Apps on the" (2017) **Run Apps on the Android Emulator.** [Online]. Available from: https://developer.android.com/studio/run/emulator.html [Accessed 04/04 2017].

"Size and Format" (2017) **Launcher Icons Size and Format.** [Online]. Available from: https://developer.android.com/guide/practices/ui_guidelines/icon_design_launcher.html#size [Accessed 04/03 2017].

"Status Bar Icons" (2017) **Develop API Guides Status Bar Icons.** [Online]. Available from: https://developer.android.com/guide/practices/ui_guidelines/icon_design_status_bar.html [Accessed 04/03 2017].

"Steppers" (2017) **Material Design Components Steppers.** [Online]. Available from: https://material.io/guidelines/components/steppers.html [Accessed 04/07 2017].

"Testing UI" (2017) **Develop Training Best Practices for Testing Automating UI Tests Testing UI for a Single App.** [Online]. Available from: https://developer.android.com/training/testing/ui-testing/espresso-testing.html [Accessed 04/06 2017].

"Toasts" (2017) **Develop API Guides User Interface Toasts.** [Online]. Available from: https://developer.android.com/guide/topics/ui/notifiers/toasts.html [Accessed 04/08 2017].

"Tooltips" (2017) **Material Design Components Tooltips.** [Online]. Available from: https://material.io/guidelines/components/tooltips.html [Accessed 04/08 2017].

"UI/Application Exerciser" (2017) **Android Studio User Guide UI/Application Exerciser Monkey.** [Online]. Available from: https://developer.android.com/studio/test/monkey.html [Accessed 04/07 2017].

"What is Wireframing" (2017) **Home FAQs What is wireframing?** [Online]. Available from: http://www.experienceux.co.uk/faqs/what-is-wireframing/ [Accessed 04/07 2017].

"XML Tutorial" (2017) **XML Tutorial.** [Online]. Available from: https://www.w3schools.com/xml/ [Accessed 04/04 2017].

Ariely, D. and Wertenbroch, K. (2002) PROCRASTINATION, DEADLINES, AND PERFORMANCE: Self-Control by Precommitment. **American Psychological Society,** 13 (3): 219-224.

Beck, K., Beedle, M., van Bennekum, A. et al. (2001) **Manifesto for Agile Software Development.** [Online]. Available from: http://agilemanifesto.org/ [Accessed 04/08 2017].

Bolton, D. (2016) **5 Reasons to Use C++ for Android Development.** [Online]. Available from: http://insights.dice.com/2016/02/03/5-reasons-to-use-c-for-android-development/ [Accessed 04/04 2017].

Bowes, J. (2014) **Agile vs Waterfall: Comparing project management methods.** [Online]. Available from: https://manifesto.co.uk/agile-vs-waterfall-comparing-project-management-methodologies/ [Accessed 04/08 2017].

Burke, D. (2017) **O-MG, the Developer Preview of Android O is here!** [Online]. Available from: https://android-developers.googleblog.com/2017/03/first-preview-of-android-o.html [Accessed 04/08 2017].

Cannizzaro, F. (2016) **Material Stepper.** 1.2.2 edn, GitHub, https://github.com/fcannizzaro/material-stepper.

Chacon, S. and Straub, B. (2014) **Pro Git.** 2nd ed. Apress.

Cirillo, F. (2007) **The Pomodoro Technique (The Pomodoro).** Vol. 1.3.

Darrow, B. (2015) **Google Drive claims one million paying customers, er, organizations.** [Online]. Available from: http://fortune.com/2015/09/21/google-drive-1m-paid-users/ [Accessed 04/09 2017].

Eason, J. (2015) **An update on Eclipse Android Developer Tools.** [Online]. Available from: https://android-developers.googleblog.com/2015/06/an-update-on-eclipse-android-developer.html [Accessed 04/03 2017].

GDJ (2016) **Open Clip Art Brain Light Bulb.** [Online]. Available from: https://openclipart.org/detail/266221/brain-light-bulb [Accessed 04/03 2017].

Gosling, J. Joy, B. Steele, G. et al. (2015) **The Java® Language Specification Java SE 8 Edition.** 5th ed. New Jersey, United States: Pearson Education (US), Addison-Wesley Educational Publishers Inc.

Goyvaerts, J. (2016) **Regular-Expressions.info.** [Online]. Available from: http://www.regular-expressions.info/ [Accessed 04/09 2017].

Gruber, J. (2004) **Daring Fireball: Markdown.** [Online]. Available from: https://daringfireball.net/projects/markdown/ [Accessed 04/03 2017].

Hall, E.M. (1998) **Managing Risk: Methods for Software Systems Development.** Longman, Harlow, Essex, U.K: Addison-Wesley.

Halverson, T. and Hornof, A.J. (2004) Link colors guide a search. **CHI '04 Extended Abstracts on Human Factors in Computing Systems,** 1367-1370.

Kainulainen, P. (2004) **Getting Started With Gradle.** [Online]. Available from: https://www.petrikainulainen.net/getting-started-with-gradle/ [Accessed 04/06 2017].

Kanak, A. (2016) **Android Week View.** 1.2.6 edn, GitHub, https://github.com/alamkanak/Android-Week-View.

Morocho, D. (2017) **OneCalendarView.** 3.1.1 edn, GitHub, https://github.com/MorochoRochaDarwin/OneCalendarView?utm_source=android-arsenal.com&utm_medium=referral&utm_campaign=5414.

Munoz, A. (2014) **Machine Learning and Optimization**PhD, Courant Institute of Mathematical Sciences, New York, NY.

Newport, C. (2008) **Fixed-Schedule Productivity: How I Accomplish a Large Amount of Work in a Small Number of Work Hours.** [Online]. Available from: http://calnewport.com/blog/2008/02/15/fixed-schedule-productivity-how-i-accomplish-a-large-amount-of-work-in-a-small-number-of-work-hours/ [Accessed 04/10 2017].

Nielsen, J. (1994) **Usability Inspection Methods.** John Wiley & Sons.

Nurik, R. (2017) **Android Asset Studio.** GitHub, http://romannurik.github.io/AndroidAssetStudio/.

Pearce, S. (2013) **GitSvnComparison.** [Online]. Available from: https://git.wiki.kernel.org/index.php/GitSvnComparsion [Accessed 04/05 2014].

Pearson, R. and van Schaik, P. (2003) The effect of spatial layout of and link color in Web pages on performance in a visual search task and interactive search task. **International Journal of Human-Computer Interaction,** 59 327-353.

Pope, N.G. (2016) HOW THE TIME OF DAY AFFECTS PRODUCTIVITY: EVIDENCE FROM SCHOOL SCHEDULES. **THE REVIEW OF ECONOMICS AND STATISTICS,** 98 (1): 1-11.

Robertson, S. and Robertson, J.C. (2006) **Mastering the Requirement Process.** 2nd ed. Boston, Mass.; London: Addison-Wesley Professional.

Rouse, M. (2017) **unit testing.** [Online]. Available from: http://searchsoftwarequality.techtarget.com/definition/unit-testing [Accessed 04/06 2017].

Rouse, M. (2016) **integrated development environment (IDE).** [Online]. Available from: http://searchsoftwarequality.techtarget.com/definition/integrated-development-environment [Accessed 04/03 2017].

Rouse, M. (2014) **entity relationship diagram.** [Online]. Available from: http://searchcrm.techtarget.com/definition/entity-relationship-diagram [Accessed 04/06 2017].

Shindler, J. (2009) **Transformative Classroom Management: Positive Strategies to Engage All Students and Promote a Psychology of Success.** 1st ed. Jossey Bass.

Shneiderman, B. Plaisant, C. Cohen, M. et al. (2016) **Designing the User Interface : Strategies for Effective Human-Computer Interaction.** 6th Revised ed. United States: Pearson Education (US).

StepstoneTech (2017) **Android Material Stepper.** 3.2nd edn, GitHub, https://github.com/stepstone-tech/android-material-stepper.

Stevens, P. and Pooley, R.J. (2000) **Using UML: Software Engineering with Objects and Components.** Updated ed. Addison-Wesley.

Sui, L. (2016) **Strategy Analytics: Android Captures Record 88 Percent Share of Global Smartphone Shipments in Q3 2016.** [Online]. Available from: https://www.strategyanalytics.com/strategy-analytics/news/strategy-analytics-press-releases/strategy-analytics-press-release/2016/11/02/strategy-analytics-android-captures-record-88-percent-share-of-global-smartphone-shipments-in-q3-2016#.WO14C9LysuV [Accessed 04/04 2017].

Ubrani, J., Llamas, R. and Shirer, M. (2017) **Wearables Aren't Dead, They're Just Shifting Focus as the Market Grows 16.9% in the Fourth Quarter, According to IDC.** [Online]. Available from:

http://www.idc.com/getdoc.jsp?containerId=prUS42342317 [Accessed 04/08 2017].

16 Appendices

A Structure of project ZIP

Readme file
This readme file will explain the structure of the supplied ZIP file, as well as how to run the code of the application.

Code
All the code for the Smart Revision Calendar application can be can be found in the *SmartRevisionCalendar/app/src/main* directory within the zip. All of the instrument espresso test cases can be found in the *SmartRevisionCalendar/app/src/androidTest* directory.

How to run the application
The application can be run in a number of ways, from easiest to hardest:

- Download the application onto an Android smart phone via the provided app store link
- Download the application onto an Android smart phone using the provided apk file
- Import the code to the application into Android Studio, either via the Git repo through the link provided (VCS -> Checkout from version control -> Git) or using the provided code (File -> New -> Import Project), build the project, then run it either via an emulator or via an Android smart phone.

The unit tests can be run by following the steps in the third option above, and instead of choosing run, locate the androidTest folder, right click on it and select “Run Tests”.

Write up
The project write up, is the “Russell Waterson 1330057 Final Year Project Report.pdf” file.

Wireframe
The interactive wireframe created as part of the UI design, can be accessed via the *Revision Calendar Wireframe* folder, and opening *index.html* with a web browser. A browser extension called JustInMind is required to operate the wireframe.

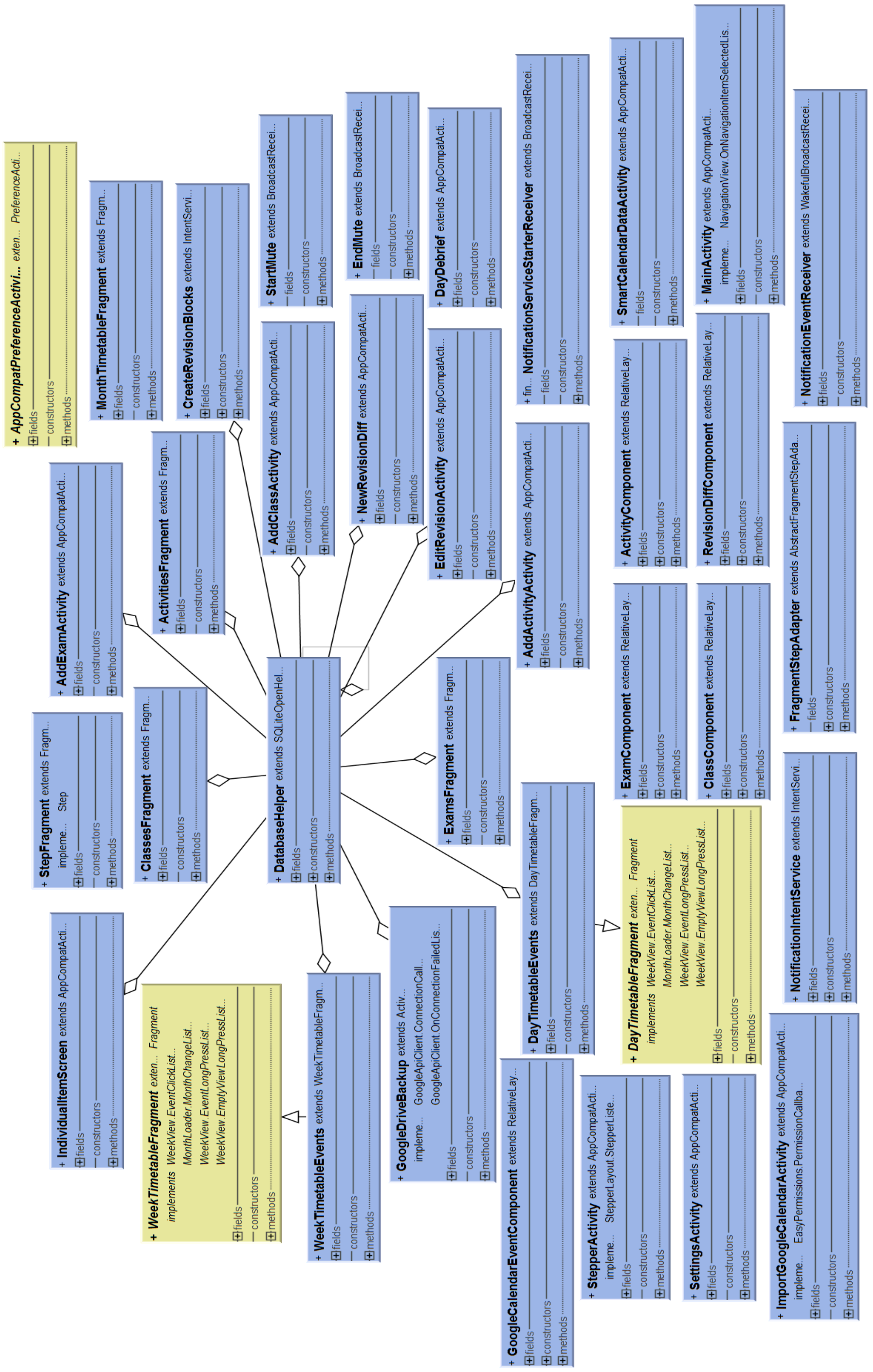
Javadoc
The generated Javadoc for the project can be accessed via *Javadoc* folder, and opening *index.html* with a web browser.

Link to app store
A link to the applications page on Google Play Store can be found through the “App in Google Play Store” link.

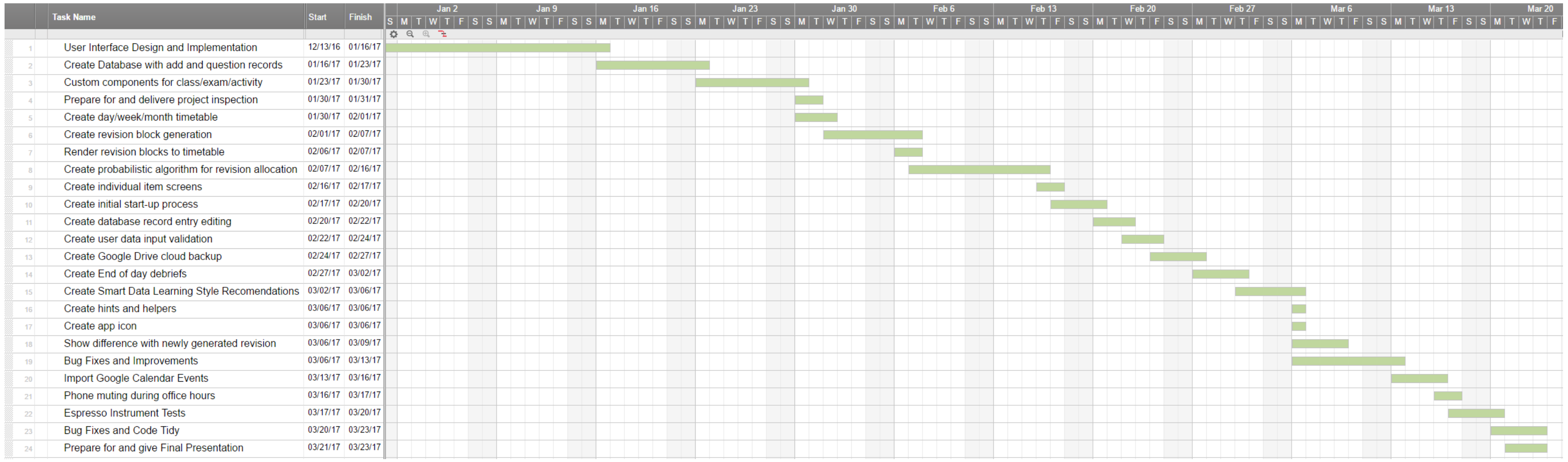
Link to repository
A link to the projects Git repository can be found through the “Git Repository” link.

Survey
A link to the online survey created at the start of the project for requirements gathering, can be found through the “Requirements Gathering Survey” link.

B Class Diagram


+ AppCompatPreferenceActivi... exten... PreferenceActi...
+ MonthTimetableFragment extends Fragm...
+ CreateRevisionBlocks extends IntentServi...
+ StartMute extends BroadcastRecei...
+ EndMute extends BroadcastRecei...
+ DayDebrief extends AppCompatActi...
+ fin... NotificationServiceStarterReceiver extends BroadcastRecei...
+ SmartCalendarDataActivity extends AppCompatActi...
+ MainActivity extends AppCompatActi... impleme... NavigationView.OnNavigationItemSelectedLis...
+ NotificationEventReceiver extends WakefulBroadcastRecei...
+ AddExamActivity extends AppCompatActi...
+ ActivitiesFragment extends Fragm...
+ AddClassActivity extends AppCompatActi...
+ NewRevisionDiff extends AppCompatActi...
+ EditRevisionActivity extends AppCompatActi...
+ AddActivityActivity extends AppCompatActi...
+ ActivityComponent extends RelativeLay...
+ RevisionDiffComponent extends RelativeLay...
+ FragmentStepAdapter extends AbstractFragmentStepAda...
+ StepFragment extends Fragm... impleme... Step
+ ClassesFragment extends Fragm...
+ DatabaseHelper extends SQLiteOpenHel...
+ ExamsFragment extends Fragm...
+ ExamComponent extends RelativeLay...
+ ClassComponent extends RelativeLay...
+ NotificationIntentService extends IntentServi...
+ DayTimetableEvents extends DayTimetableFragm...
+ DayTimetableFragment exten... Fragment implements WeekView.EventClickList... MonthLoader.MonthChangeList... WeekView.EventLongPressList... WeekView.EmptyViewLongPressList...
+ IndividualItemScreen extends AppCompatActi...
+ WeekTimetableFragment exten... Fragment implements WeekView.EventClickList... MonthLoader.MonthChangeList... WeekView.EventLongPressList... WeekView.EmptyViewLongPressList...
+ WeekTimetableEvents extends WeekTimetableFragm...
+ GoogleDriveBackup extends Activ... impleme... GoogleApiClient.ConnectionCall... GoogleApiClient.OnConnectionFailedLis...
+ GoogleCalendarEventComponent extends RelativeLay...
+ StepperActivity extends AppCompatActi... impleme... StepperLayout.StepperListe...
+ SettingsActivity extends AppCompatActi...
+ ImportGoogleCalendarActivity extends AppCompatActi... impleme... EasyPermissions.PermissionCallba...
fields
constructors
methods

C Gantt Chart



	Task Name	Start	Finish
1	User Interface Design and Implementation	12/13/16	01/16/17
2	Create Database with add and question records	01/16/17	01/23/17
3	Custom components for class/exam/activity	01/23/17	01/30/17
4	Prepare for and delivere project inspection	01/30/17	01/31/17
5	Create day/week/month timetable	01/30/17	02/01/17
6	Create revision block generation	02/01/17	02/07/17
7	Render revision blocks to timetable	02/06/17	02/07/17
8	Create probabilistic algorithm for revision allocation	02/07/17	02/16/17
9	Create individual item screens	02/16/17	02/17/17
10	Create initial start-up process	02/17/17	02/20/17
11	Create database record entry editing	02/20/17	02/22/17
12	Create user data input validation	02/22/17	02/24/17
13	Create Google Drive cloud backup	02/24/17	02/27/17
14	Create End of day debriefs	02/27/17	03/02/17
15	Create Smart Data Learning Style Recomendations	03/02/17	03/06/17
16	Create hints and helpers	03/06/17	03/06/17
17	Create app icon	03/06/17	03/06/17
18	Show difference with newly generated revision	03/06/17	03/09/17
19	Bug Fixes and Improvements	03/06/17	03/13/17
20	Import Google Calendar Events	03/13/17	03/16/17
21	Phone muting during office hours	03/16/17	03/17/17
22	Espresso Instrument Tests	03/17/17	03/20/17
23	Bug Fixes and Code Tidy	03/20/17	03/23/17
24	Prepare for and give Final Presentation	03/21/17	03/23/17

D Unit Tests

Test	Expected Output	Result
Google Drive Backups		
Attempt backup with all empty databases	Backs up to Google Drive with a text file of just headers and a DB file with no records	Pass
Attempt backup with all databases containing data	Backs up to Google Drive with a text file and DB file both containing all previously entered records	Pass
Attempt backup with mixture of empty and populated databases	Backs up to Google Drive with a text file and DB file both containing all previously entered records with some empty tables	Pass
Cancel the upload of the text file	The text file should not be uploaded to Drive	Pass
Cancel the upload of the database file	The database file should not be uploaded to Drive	Pass
Check date and time of last upload	After a backup, the last backup time should change to the current time	Pass
Activity Component		
Create new Activity Component	The reusable custom component for displaying an activity will be created, one for each activity	Pass
Set activity	The component will show the activity title to be the same as the text passed into this method	Pass
Set time day text	The component will show the activity time and day to be the text passed into this method	Pass
Set repeat – true	Will display the repeat icon	Pass
Set repeat – false	Will not display the repeat icon	Pass
Set coloured line	Will set the line to the colour that was passed in	Pass
Class Component		
Create new Class Component	The reusable custom component for displaying a class will be created, one for each class	Pass
Set module title text	The component will show the class module and title to be the same as the text passed into this method	Pass
Set teacher room text	The component will show the class teacher and room to be the same as the text passed into this method	Pass
Set time day text	The component will show the class time and day to be the text passed into this method	Pass
Set repeat – true	Will display the repeat icon	Pass
Set repeat – false	Will not display the repeat icon	Pass
Set coloured line	Will set the line to the colour that was passed in	Pass
Exam Component		
Create new Exam Component	The reusable custom component for displaying an exam will be created, one for each exam	Pass
Set module title text	The component will show the exam module and title to be the same as the text passed into this method	Pass
Set date time text	The component will show the exam date and time to be the text passed into this method	Pass
Set days to go – future	The method will calculate and then show in the component how many days to go based on the date value that was passed in	Pass
Set days to go – today	The method will calculate and then show in the component 0 days to go given the date value that was passed in is today	Pass
Set days to go – past	The method will calculate and then show in the component how many days has elapsed since the date value that was passed in	Pass
Set days to go – invalid date	The method will attempt to parse the date and then show in the component that it is an unknown number of days to go	Pass

Set priority – priority of 0	The component will show 0 filled in drawable views all with the outline of the passed in colour	Pass
Set priority – priority between 1 and 5 (inclusive)	The component will show filled in drawable views the same as the priority number passed in from left to right, all with the same colour and outline colour as the passed in colour	Pass
Set priority – priority greater than 5	This case should never happen due to input validation; however, it handles it by filling in all the drawable views, all with the same colour and outline colour as the passed in colour	Pass
Set coloured line	Will set the line to the colour that was passed in	Pass
Revision Diff Component		
Create new Revision Diff Component	The reusable custom component for comparing old revision in the database with new revision will be created, one for each day to compare	Pass
Set revision date	The component will show the date of revision to be the same as the text passed into this method	Pass
Set old revision	The component will show the all the old revision to be the text passed into this method	Pass
Set new revision	The component will show the all the new revision to be the text passed into this method	Pass
Database Helper		
Insert class data	The new class record will be added to the class table in the database	Pass
Get all class data	All records from the class table is returned	Pass
Update class data	A selected class record will be updated with the new values passed in	Pass
Delete class data	A selected class record will be deleted from the class table in the database	Pass
Insert activity data	The new activity record will be added to the activity table in the database	Pass
Get all activity data	All records from the activity table is returned	Pass
Update activity data	A selected activity record will be updated with the new values passed in	Pass
Delete activity data	A selected activity record will be deleted from the activity table in the database	Pass
Insert exam data	The new exam record will be added to the exam table in the database	Pass
Get all exam data	All records from the exam table is returned	Pass
Update exam data	A selected exam record will be updated with the new values passed in	Pass
Delete exam data	A selected exam record will be deleted from the exam table in the database	Pass
Insert revision data	The new revision record will be added to the revision table in the database	Pass
Get all revision data	All records from the revision table is returned	Pass
Update revision data	A selected revision record will be updated with the new values passed in	Pass
Delete revision data	A selected revision record will be deleted from the revision table in the database	Pass
Insert debrief data	The new debrief record will be added to the debrief table in the database	Pass
Get all debrief data	All records from the debrief table is returned	Pass
Day Debrief		
Submit daily debrief	A new debrief record containing all the entered data will be added to the debrief table in the database	Pass
Do not submit daily debrief	A debrief record will not being added to the debrief table	Pass
Notification Event Receiver		

Setup alarm	An alarm is setup to be trigger at the set time and repeating at the set interval	Pass
Cancel alarm	The previously setup alarm is cancelled as well as any other repeating occurrences	Pass
Notification Intent Service		
Process start notification	A notification is built and shown from anywhere on the phone, containing the set title, context text, colour, and icon.	Pass
Try to display notification when there has been no revision on that day	The notification will not be displayed when there is no revision for that particular day	Pass
Notification Service Start Receiver		
Reboot System while before end of day and daily debriefs on	The debrief notification should not be displayed	Pass
Reboot System while after end of day and daily debriefs on	The debrief notification should be displayed	Pass
Change System Time Zone while before end of day and daily debriefs on	The debrief notification should not be displayed	Pass
Change System Time Zone while after end of day and daily debriefs on	The debrief notification should be displayed	Pass
Change System Clock while before end of day and daily debriefs on	The debrief notification should not be displayed	Pass
Change System Clock while after end of day and daily debriefs on	The debrief notification should be displayed	Pass
All above tests while daily debriefs are off	The debrief notification should not be displayed	Pass
Reboot System while before start of day and mute phone on	The phone will be set to normal ring setting	Pass
Reboot System while after start of day and before end of day and mute phone on	The phone will be set to mute setting	Pass
Reboot System while after end of day and mute phone on	The phone will be set to normal ring setting	Pass
Change System Time Zone while before start of day and phone mute on	The phone will be set to normal ring setting	Pass
Change System Time Zone while after start of day and before end of day and mute phone on	The phone will be set to mute setting	Pass
Change System Time Zone while after end of day and mute phone on	The phone will be set to normal ring setting	Pass
Change System Clock while before start of day and mute phone on	The phone will be set to normal ring setting	Pass
Change System Clock while after start of day and before end of day and mute phone on	The phone will be set to mute setting	Pass
Change System Clock while after end of day and mute phone on	The phone will be set to normal ring setting	Pass
All above tests while daily debriefs are off	The phone will not change ringing state	Pass
Import Google Calendar Activity		

Enter nothing into the events field	An error message should be displayed informing the user they must enter a number of events to search	Pass
Enter an invalid number of events	An error message should be displayed informing the user of the range of events that can be entered	Pass
Enter a valid number of events without Google Play Services installed	An error message should be displayed informing the user they must install Google Play Services	Pass
Enter a valid number of events without a network connection	An error message should be displayed informing the user there is no network connection available	Pass
Enter a valid number of events without permission on a device with API level 23 and above	A runtime permission request form should be displayed requesting the GET_ACCOUNTS permission	Pass
Enter a valid number of events without permission on a device with API level 23 and below	The GET_ACCOUNTS permission should be automatically accepted	Pass
The GET_ACCOUNTS permission is accepted	The system should ask the user which Google Account they wish to use to connect their Google Calendar	Pass
The GET_ACCOUNTS permission is declined	The system should not continue with importing of the Google Calendar	Pass
A Google Account is selected	The Google Calendar API should be called, then a list of events should be displayed	Pass
Permission and account had been previously selected	As above	Pass
Events are ticked and save is pressed	The events should be added to the Events table in the database with all the correct details and the selected colour	Pass
Activities Fragment		
View fragment whilst there are no activities in the DB	No activities should be displayed, and in its place a message informing the user there are no activities	Pass
View fragment whilst there is 1 activity in the DB	One activity custom component should be displayed	Pass
View fragment whilst multiple activities are in the DB (enough to fill more than a screen)	All the activities should be displayed as individual custom components one after each other, and there should be the ability to scroll up and down through them all	Pass
Click on an activity component	The individual item screen should be displayed, showing all the information that the chosen activity contains	Pass
Click on the “add” floating action button	The add new activity screen should be displayed	Pass
Return to the activities screen after new activity has been added	The fragment should be updated to show the newly added activity at the end of the existing list	Pass
Return to the activities screen after a new activity was not added	The fragment should not change and show the same unchanged activities displayed previously	Pass
Class Fragment		
View fragment whilst there are no classes in the DB	No classes should be displayed, and in its place a message informing the user there are no classes	Pass
View fragment whilst there is 1 class in the DB	One class custom component should be displayed	Pass
View fragment whilst multiple classes are in the DB (enough to fill more than a screen)	All the classes should be displayed as individual custom components one after each other, and there should be the ability to scroll up and down through them all	Pass
Click on a class component	The individual item screen should be displayed, showing all the information that the chosen class contains	Pass
Click on the “add” floating action button	The add new class screen should be displayed	Pass
Return to the classes screen after new class has been added	The fragment should be updated to show the newly added class at the end of the existing list	Pass

Return to the classes screen after a new class was not added	The fragment should not change and show the same unchanged classes displayed previously	Pass
Exams Fragment		
View fragment whilst there are no exams in the DB	No exams should be displayed, and in its place a message informing the user there are no exams	Pass
View fragment whilst there is 1 exam in the DB	One exam custom component should be displayed	Pass
View fragment whilst multiple exams are in the DB (enough to fill more than a screen)	All the exams should be displayed as individual custom components one after each other, and there should be the ability to scroll up and down through them all	Pass
Click on an exam component	The individual item screen should be displayed, showing all the information that the chosen exam contains	Pass
Click on the “add” floating action button	The add new exam screen should be displayed	Pass
Return to the exams screen after new exam has been added	The fragment should be updated to show the newly added exam at the end of the existing list	Pass
Return to the exams screen after a new exam was not added	The fragment should not change and show the same unchanged exams displayed previously	Pass
Individual Item Screen		
A class has been clicked to show individual screen	All text and icon views that are not part of class records should be removed, and module, title, day, start time, end time, repeats, room, and teacher will remain, all showing the selected class’s information. It should show a collapsing toolbar layout whereby the module and title are in the toolbar, and it is of the colour of the class.	Pass
Edit class is clicked	The same screen for add a class should be displayed, but instead with title edit class, and all the entries pre-filled with the selected class	Pass
A class has been edited	The class table in the database should update the selected record with the newly entered fields. It should return to the individual item screen showing the recently made changes	Pass
A class has not been edited	The class table in the database should not be updated. When returning to the individual item screen, it should remain the same as before with no changes	Pass
A class is deleted	A dialog should appear asking to confirm whether or not to delete the class. When the class is deleted, it should be removed from the class table in the database	Pass
A class is not deleted	A dialog should appear asking to confirm whether or not to delete the class. When the class is not deleted, it should remain in the class table unchanged	Pass
An activity has been clicked to show individual screen	All text and icon views that are not part of activity records should be removed, and title, day, start time, end time, and repeats will remain, all showing the selected activity’s information. It should show a collapsing toolbar layout whereby the title is in the toolbar, and it is of the colour of the activity.	Pass
Edit activity is clicked	The same screen for add an activity should be displayed, but instead with title edit activity, and all the entries pre-filled with the selected activity	Pass
An activity has been edited	The activity table in the database should update the selected record with the newly entered fields. It should return to the individual item screen showing the recently made changes	Pass
An activity has not been edited	The activity table in the database should not be updated. When returning to the individual item screen, it should remain the same as before with no changes	Pass

An activity is deleted	A dialog should appear asking to confirm whether or not to delete the activity. When the activity is deleted, it should be removed from the activity table in the database	Pass
An activity is not deleted	A dialog should appear asking to confirm whether or not to delete the activity. When the activity is not deleted, it should remain in the activity table unchanged	Pass
An exam has been clicked to show individual screen	All text and icon views that are not part of exam records should be removed, and module, title, date, start time, end time, content size, and priority will remain, all showing the selected exam's information. It should show a collapsing toolbar layout whereby the module and title are in the toolbar, and it is of the colour of the exam.	Pass
Edit exam is clicked	The same screen for add an exam should be displayed, but instead with title edit exam, and all the entries pre-filled with the selected exam	Pass
An exam has been edited	The exam table in the database should update the selected record with the newly entered fields. It should return to the individual item screen showing the recently made changes	Pass
An exam has not been edited	The exam table in the database should not be updated. When returning to the individual item screen, it should remain the same as before with no changes	Pass
An exam is deleted	A dialog should appear asking to confirm whether or not to delete the exam. When the exam is deleted, it should be removed from the exam table in the database	Pass
An exam is not deleted	A dialog should appear asking to confirm whether or not to delete the exam. When the exam is not deleted, it should remain in the exam table unchanged	Pass
A revision slot has been clicked to show individual screen	All text and icon views that are not part of revision records should be removed, and module, title, date, start time and end time will remain, all showing the selected revision's information. It should show a collapsing toolbar layout whereby the module and title are in the toolbar, and it is of the colour of the revision.	Pass
A free revision slot has been clicked to show individual screen	As above, but the title should read Free Revision Slot and the colour of the collapsing toolbar should be black. It should also inform the user of what a free revision slot is and what their options are.	Pass
Edit revision is clicked	A screen for edit revision should be displayed and all the entries pre-filled with the selected revision	Pass
A revision block has been edited	The revision table in the database should update the selected record with the newly entered fields. It should return to the individual item screen showing the recently made changes	Pass
A revision block has not been edited	The revision table in the database should not be updated. When returning to the individual item screen, it should remain the same as before with no changes	Pass
A revision block is deleted	A dialog should appear asking to confirm whether or not to delete the revision. When the revision is deleted, it should be removed from the revision table in the database	Pass
A revision block is not deleted	A dialog should appear asking to confirm whether or not to delete the revision. When the revision is not deleted, it should remain in the revision table unchanged	Pass
Main Activity		
The application is loaded and it's the first-time boot	On first boot the app should automatically show the initial start-up welcome screens to guide the user through setup. After this setup is complete, it should show the user a dialog	Pass

	explaining the navigation of the system. The app should then be loaded as normal (as below).	
The application is loaded	The app is loaded showing the default tab of the timetable view. There should be the bottom navigation view allowing for transitions to timetable, exams, classes, and activities screens. There should also be access to a side navigation view granting access to smart calendar data, import google calendar, settings, and backup screens, as well as showing the user's name.	Pass
The application is loaded on the day of an exam	When the app is loaded on the day of an exam in the system, a dialog should be displayed wishing the user good luck and explaining to them the revision that should be undertaken that day.	Pass
Timetable tab is selected	The timetable fragment will be displayed showing the last clicked day/week/month view	Pass
Day view is selected from timetable tab	The day view should be displayed showing the current day along with all the events for that day	Pass
Week view is selected from timetable tab	The week view should be displayed showing the current week along with all the events for that week	Pass
Month view is selected from timetable tab	The month view should be displayed showing the current month along with the number of days until the user's next exam	Pass
Exam tab is selected	The exam fragment should be displayed showing all the exams entered into the system. Along with the number of upcoming exams the user has.	Pass
Exam tab is selected when there are no exams in the future in the system	The exam fragment should be displayed showing all the exams entered into the system. As well as showing the user having 0 upcoming exams	Pass
Class tab is selected	The class fragment should be displayed showing all the classes entered into the system.	Pass
Activity tab is selected	The activity fragment should be displayed showing all the activities entered into the system.	Pass
Side navigation is selected	The side navigation draw should slide from the left to display the menu	Pass
Smart Calendar Data is selected from the side navigation bar	The smart calendar data activity should be started and displayed	Pass
Import Google Calendar is selected from the side navigation bar	The import google calendar activity should be started and displayed	Pass
Settings is selected from the side navigation bar	The settings activity should be started and displayed	Pass
Last Backup is selected from the side navigation bar	The back activity should be started and displayed, the current date and time will be taken and displayed as the last backup in the side navigation bar	Pass
End Mute		
Broadcast is received to start the end mute	The phone should change from a silent ringer mode to a normal ringer mode	Pass
Start Mute		
Broadcast is received to start the start mute	The phone should change from a normal ringer mode to a silent ringer mode	Pass
Add Activity Activity		
The add an activity screen is clicked to add a new activity	The add activity screen should be displayed with the title in the toolbar of "Add Activity" and all the fields should be displayed as empty of default values. Activity (empty), Day (Monday), Start Time (00:00), End Time (00:00), Repeat Weekly (Off), and Colour (Light Blue).	Pass

The activity is edited	When selected, the normal letter based keyboard is displayed where the user can type words, numbers and symbols	Pass
The day is edited	A dropdown containing every day of the week should be displayed. When a day is clicked, it should show as the one selected.	Pass
The start time is edited	A time picker dialog should be displayed, when a time is picked it should show as the selected time.	Pass
The end time is edited	As above	Pass
The repeat is edited	The toggle should alternate between an off and on state	Pass
The colour is edited	A dialog should be displayed containing a list of a variety of colours. When a colour is selected, the colour view should change to the selected colour.	Pass
Save is clicked before any details have been entered	The system should alert the user that the validation checks have failed. In this case, it should fail for the activity name being empty, and the end time not being allowed to be the same or before the start time.	Pass
Save is clicked with all valid details entered	The activity should be added to the system containing all the previously entered data. The user should be notified that the activity has been saved and the system should return to the previous screen (activity tab or activity page during start-up) showing the newly entered activity.	Pass
The add an activity screen is clicked to edit an existing activity	The add activity screen should be displayed with the title in the toolbar of "Edit Activity" and all the fields should be displayed as the values of the selected activity.	Pass
Leave the edit screen without confirming the edit	It should return to the previous screen and the selected activity should be unchanged	Pass
Make changes and click edit button	It should return to the previous screen, alert the user that the edit has been made, and update the activity in the database with the entered data.	Pass
Add Class Activity		
The add a class screen is clicked to add a new class	The add class screen should be displayed with the title in the toolbar of "Add Class" and all the fields should be displayed as empty of default values. Module (empty), Title (empty), Day (Monday), Start Time (00:00), End Time (00:00), Repeat Weekly (Off), Teacher (empty), Room (empty), and Colour (Light Blue).	Pass
The module is edited to be the same as an existing module	When the user starts typing in the module, a dropdown should appear containing a list of all the existing module already in the system. When the user selects one of the modules, it should then be displayed in the modules field, and the colour should automatically change to match the colour previously entered with that module.	Pass
The title is edited	When selected, the normal letter based keyboard is displayed where the user can type words, numbers and symbols	Pass
The day is edited	A dropdown containing every day of the week should be displayed. When a day is clicked, it should show as the one selected.	Pass
The start time is edited	A time picker dialog should be displayed, when a time is picked it should show as the selected time.	Pass
The end time is edited	As above	Pass
The repeat is edited	The toggle should alternate between an off and on state	Pass
The teacher is edited	When selected, the normal letter based keyboard is displayed where the user can type words, numbers and symbols	Pass
The room is edited	As above	Pass
The colour is edited	A dialog should be displayed containing a list of a variety of colours. When a colour is selected, the colour view should change to the selected colour.	Pass

Save is clicked before any details have been entered	The system should alert the user that the validation checks have failed. In this case, it should fail for the module and the title being empty, and the end time not being allowed to be the same or before the start time.	Pass
Save is clicked with all valid details entered	The class should be added to the system containing all the previously entered data. The user should be notified that the class has been saved and the system should return to the previous screen (class tab or class page during start-up) showing the newly entered class.	Pass
The add a class screen is clicked to edit an existing class	The add class screen should be displayed with the title in the toolbar of "Edit Class" and all the fields should be displayed as the values of the selected class.	Pass
Leave the edit screen without confirming the edit	It should return to the previous screen and the selected class should be unchanged.	Pass
Make changes and click edit button	It should return to the previous screen, alert the user that the edit has been made, and update the class in the database with the entered data.	Pass
Add Exam Activity		
The add an exam screen is clicked to add a new exam	The add exam screen should be displayed with the title in the toolbar of "Add Exam" and all the fields should be displayed as empty of default values. Module (empty), Title (empty), Date (00-00-0000), Start Time (00:00), Duration (empty), More Info (empty), Colour (Light Blue), Content Size (empty), and Priority (empty).	Pass
The module is edited to be the same as an existing module	When the user starts typing in the module, a dropdown should appear containing a list of all the existing module already in the system. When the user selects one of the modules, it should then be displayed in the modules field, and the colour should automatically change to match the colour previously entered with that module.	Pass
The title is edited	When selected, the normal letter based keyboard is displayed where the user can type words, numbers and symbols	Pass
The date is edited	A date picker dialog should be displayed, when a date is picked it should show as the newly selected date.	Pass
The start time is edited	A time picker dialog should be displayed, when a time is picked it should show as the selected time.	Pass
The duration is edited	When selected, a number pad based keyboard is displayed where the user can enter only positive whole numbers	Pass
More info is edited	As when title is selected	Pass
The colour is edited	A dialog should be displayed containing a list of a variety of colours. When a colour is selected, the colour view should change to the selected colour.	Pass
Course content is edited	As when duration is selected	Pass
Priority is edited	As when duration is selected	Pass
Save is clicked before any details have been entered	The system should alert the user that the validation checks have failed. In this case, it should fail for the module and the title being empty, the date being invalid, and the duration being empty.	Pass
Save is clicked with all valid details entered	The exam should be added to the system containing all the previously entered data. The user should be notified that the exam has been saved and the system should return to the previous screen (exam tab or exam page during start-up) showing the newly entered exam.	Pass
The add an exam screen is clicked to edit an existing exam	The add exam screen should be displayed with the title in the toolbar of "Edit Exam" and all the fields should be displayed as the values of the selected exam.	Pass

Leave the edit screen without confirming the edit	It should return to the previous screen and the selected exam should be unchanged.	Pass
Make changes and click edit button	It should return to the previous screen, alert the user that the edit has been made, and update the exam in the database with the entered data.	Pass
Edit Revision Activity		
The edit a revision screen is clicked to edit an existing revision	The edit revision screen should be displayed with the title in the toolbar of "Edit Revision Block" and all the fields should be displayed as the values of the selected revision.	Pass
The module is edited to be the same as an existing module	When the user starts typing in the module, a dropdown should appear containing a list of all the existing module already in the system. When the user selects one of the modules, it should then be displayed in the modules field, and the colour should automatically change to match the colour previously entered with that module.	Pass
The title is edited	When selected, the normal letter based keyboard is displayed where the user can type words, numbers and symbols	Pass
The date is edited	A date picker dialog should be displayed, when a date is picked it should show as the newly selected date.	Pass
The start time is edited	A time picker dialog should be displayed, when a time is picked it should show as the selected time.	Pass
The end time is edited	As when start time is edited	Pass
Notes is edited	As when title is selected	Pass
The colour is edited	A dialog should be displayed containing a list of a variety of colours. When a colour is selected, the colour view should change to the selected colour.	Pass
Edit is clicked after all the details have been removed	The system should alert the user that the validation checks have failed. In this case, it should fail for the module and the title being empty, the date being invalid, and the end time not being allowed to be the same or before the start time.	Pass
Leave the edit screen without confirming the edit	It should return to the previous screen and the selected revision should be unchanged.	Pass
Make changes and click edit button once all entries are valid	It should return to the previous screen, alert the user that the edit has been made, and update the revision in the database with the entered data.	Pass
Setting Activity		
Mute phone is switched on	The user should be informed that the scheduled muting is being setup, and the system should then setup the start of the mute at the preferred revision start time, and the end of the mute at the preferred revision end time.	Pass
Mute phone is switched off	The phone should be unmuted and the start of the mute and the end of the mute broadcasts should be cancelled.	Pass
Rerun Initial Setup	The original start-up welcome screens should be displayed however showing all the data already in the system.	Pass
Contact Me	A list of installed email clients should be displayed, once one is clicked an email should be drafted with my email is the recipient and "Smart Revision Calendar" as the subject.	Pass
Contact Me, when an email client is not installed	A message should be displayed informing the user than an email client is not currently installed.	Pass
Changelog	A dialog should be displayed showing the user of the dates and changelog of the last updates.	Pass
About	A dialog should be displayed informing the user of what the project is about, as well as a disclaimer.	Pass
Smart Calendar Data Activity		
Smart Calendar Active toggle activated	The toggle should switch to the on position, and revision slots should now be able to be generated	Pass

Smart Calendar Active toggle deactivate	The toggle should switch to the off position, and all the revision currently in the system should be removed	Pass
Generate smart revision when smart features are not active	An error message should be displayed informing the user that smart features are active	Pass
Generate smart revision when there are no exams in the system	An error message should be displayed informing the user that there are no exams in the system	Pass
Generate smart revision when the revision block size is 0	An error message should be displayed informing the user that the revision block size is 0	Pass
Generate smart revision for the first time	Revision slots should be generated whilst a progress dialog is displayed, then a dialog informing the user should be displayed once complete	Pass
Generate smart revision for a subsequent time	Revision slots should be generated whilst a progress dialog is displayed, then a dialog giving the user the option to view the difference, keep the old revision, or to keep the new revision is displayed	Pass
Generate smart revision for a subsequent time and keep old is selected	As above, and the newly generated revision should be deleted	Pass
Generate smart revision for a subsequent time and keep new is selected	As above, and the old revision should be deleted	Pass
Generate smart revision for a subsequent time and view diff is selected	As above, then the New Revision Difference activity is started	Pass
Recommend smart data updates when no debriefs have taken place	An error message should be displayed informing the user that no debriefs have taken place	Pass
Recommend smart data updates	A dialog should be displayed containing recommended smart data updates, as well as a description of how it has been generated	Pass
Recommend smart data updates then decline to change the data	As above, then no change in data takes place	Pass
Recommend smart data updates then accept the data change	As above, then all learning style smart data will be replaced by the suggested values	Pass
Recommend smart data updates then accept the data change when there is invalid data	As above, then an error message should be displayed informing the user that the data includes invalid values	Pass
Change the preferred start time	A TimePicker dialog should be displayed, then the start time should be changed to the newly selected time	Pass
Change the preferred start time to after the current end time	A TimePicker dialog should be displayed, then an error message should be displayed informing the user that the selected time is invalid, the start time will remain as the previous value	Pass
Change the preferred end time	A TimePicker dialog should be displayed, then the end time should be changed to the newly selected time	Pass
Change the preferred end time to before the current start time	A TimePicker dialog should be displayed, then an error message should be displayed informing the user that the selected time is invalid, the end time will remain as the previous value	Pass
Activate end of block debriefs	The notification alarms should be set and therefore block debriefs activated	FAIL: un-imple-ment-ed
Deactivate end of block debriefs	The notification alarms should be cancelled and therefore block debriefs deactivated	
Activate end of day debriefs	The notification alarms should be set and therefore day debriefs activated	Pass

Deactivate end of day debriefs	The notification alarms should be cancelled and therefore day debriefs deactivated	Pass
Change the revision block size	A preference dialog should be displayed, then revision size should change to become the entered value	Pass
Change the revision block size to be less than the minimum value	A preference dialog should be displayed, an error message should be displayed informing the user that the entered value is too small	Pass
Change the revision block size to be more than the maximum value	A preference dialog should be displayed, an error message should be displayed informing the user that the entered value is too large	Pass
Change the break block size	A preference dialog should be displayed, then break size should change to become the entered value	Pass
Change the break block size to be more than the maximum value	A preference dialog should be displayed, an error message should be displayed informing the user that the entered value is too large	Pass
Change the revision variety	A dialog should be displayed containing options 0 to 10, the variety should then be updated to match the value of the selected option	Pass
Create Revision Blocks		
Attempt multiple generation of revision using a vast array of different learning style data, and different exams containing a range of priorities, as well as a number of different other items existing	Revision should be generated up to the last exam in the system, within the start time and end time, and avoiding all classes, activities, and events, where revision is set correctly according to priority, content size, days till exam, and variety	Pass
New Revision Diff		
View the difference between old and new revision	An exam revision counter should be displayed, followed by a day-to-day account of what revision is present, where old is on the left and new is on the right	Pass
Choose to keep the old revision	All the newly generated revision should be deleted from the revision database, and the old remains	Pass
Choose to keep the newly generated revision	All the previously existing old revision should be deleted from the revision database, and the new remains	Pass
Step Fragment		
The initial welcome setup is displayed	The system should show the first step in the welcome setup.	Pass
Next is pressed on steps 1 to 8	The next step should be displayed.	Pass
Back is pressed on steps 2 to 9	The previous step should be displayed.	Pass
Complete is pressed on step 9	The system should alert the user that the setup is complete and close the start-up activity.	Pass
Step 1 is displayed	The first step should be displayed, showing a welcome message, a disclaimer, and a text box so the user can enter their name.	Pass
Step 1 is displayed with previously entered data	The first step should be displayed, showing the name text box containing the previously entered data.	Pass
Step 2 is displayed	The second step should be displayed informing the user about classes, and button to add new classes.	Pass
Step 2 is displayed after a new class has been added	The second step should be displayed showing the newly added class at the end of the list of existing classes.	Pass
Step 2 is displayed with previously entered data	The second step should be displayed showing a list of previously entered classes in the system.	Pass
Step 3 is displayed	The third step should be displayed informing the user about activities, and button to add new activities.	Pass
Step 3 is displayed after a new activity has been added	The third step should be displayed showing the newly added activity at the end of the list of existing activities.	Pass

Step 3 is displayed with previously entered data	The third step should be displayed showing a list of previously entered activities in the system.	Pass
Step 4 is displayed	The forth step should be displayed informing the user about connecting their Google Calendar, as well as a button to connect it.	Pass
Step 5 is displayed	The fifth step should be displayed informing the user about the cloud backup functionality available	Pass
Step 6 is displayed	The sixth step should be displayed informing the user all about the smart revision calendar features	Pass
Step 7 is displayed	The seventh step should be displayed informing the user about exam, and button to add new exams.	Pass
Step 7 is displayed after a new exam has been added	The seventh step should be displayed showing the newly added exam at the end of the list of existing exams.	Pass
Step 7 is displayed with previously entered data	The seventh step should be displayed showing a list of previously entered exams in the system.	Pass
Step 8 is displayed	The eighth step should be displayed informing the user about learning styles, as well as all the input for the user to enter their learning style.	Pass
Step 8 is displayed with previously entered data	The eighth step should be displayed showing all the previously entered learning style inputs	Pass
In step 8 the start time is edited	A time picker dialog should be displayed, when a time is picked it should show as the selected time.	Pass
In step 8 the start time is edited to be after the end time	A time picker dialog should be displayed, when a time after the current end time is selected, an alert should be displayed informing the user that they cannot select that time.	Pass
In step 8 the end time is edited	As with start time	Pass
In step 8 the end time is edited to be before the start time	As with start time	Pass
In step 8 the revision block preferred time is edited	A dialog should be displayed where the user can enter a positive number. When entered, it should be shown as the new revision time.	Pass
In step 8 the revision block is edited to be outside the accepted time	A dialog should be displayed where the user can enter a positive number. When a number outside the range of 15 and 360 minutes is entered, an alert should show explaining the entry was invalid	Pass
In step 8 the break block is edited	As with revision time	Pass
In step 8 the break block is edited to be outside the accepted time	As with revision time, but with value greater than 240 minutes	Pass
In step 8 the break block is edited to be greater than the revision block	A dialog should be displayed where the user can enter a positive number. When entered, it should be shown as the new break time, but also alerting the user that the entered time is greater than the revision time and it is not a good idea.	Pass
In step 8 the variety is edited	The slider should change the value displayed dynamically, and on release save the chosen value.	Pass
Step 9 is displayed	The ninth and final step should be displayed informing the user about debriefs, as well as checkboxes for enable them.	Pass
Step 9 is displayed with previously entered data	The ninth step should be displayed with the checkboxes selected on not depending on existing data.	Pass
In step 9 the daily debrief is activated	The system should setup the daily notification at the preferred end time of revision, which should take the user to the daily debrief screen	Pass
In step 9 the daily debrief is deactivated	The system should cancel the daily notification from showing	Pass
Stepper Activity		
Back nav button pressed	The system should transition to the previous screen in the list of steps	Pass

Back nav button pressed on the first step	The system should remain on the current step	Pass
Complete pressed	The system should alert the user that the setup is complete and close the start-up activity	Pass
Day Timetable Events		
Class added	The system should add the class to the day view timetable	Pass
Activity added	The system should add the activity to the day view timetable	Pass
Exam added	The system should add the exam to the day view timetable	Pass
Revision added	The system should add the revision to the day view timetable	Pass
Event added	The system should add the event to the day view timetable	Pass
Day Timetable Fragment		
Change current date	The system should change the date that is displayed when scrolled left and right	Pass
Change current time	The system should change the time that is displayed when scrolled up and down	Pass
Click on an activity	The individual item screen should be displayed showing the pressed activity	Pass
Click on a class	The individual item screen should be displayed showing the pressed class	Pass
Click on an exam	The individual item screen should be displayed showing the pressed exam	Pass
Click on a revision slot	The individual item screen should be displayed showing the pressed revision	Pass
Click on a Google Calendar event	The individual item screen should be displayed showing the pressed event	Pass
Month Timetable Fragment		
View month view with an exam in the future	The system should display the month view along with the number of days until the next exam	Pass
View month view with no exams in the future	The system should display the month view along with a message explain there are no upcoming exams	Pass
View month view with no exams in the system	The system should display the month view along with a message explain there are no upcoming exams	Pass
Week Timetable Events		
Class added	The system should add the class to the week view timetable	Pass
Activity added	The system should add the activity to the week view timetable	Pass
Exam added	The system should add the exam to the week view timetable	Pass
Revision added	The system should add the revision to the week view timetable	Pass
Event added	The system should add the event to the week view timetable	Pass
Week Timetable Fragment		
Change current week	The system should change the week that is displayed when scrolled left and right	Pass
Change current time	The system should change the time that is displayed when scrolled up and down	Pass
Click on an activity	The individual item screen should be displayed showing the pressed activity	Pass
Click on a class	The individual item screen should be displayed showing the pressed class	Pass
Click on an exam	The individual item screen should be displayed showing the pressed exam	Pass
Click on a revision slot	The individual item screen should be displayed showing the pressed revision	Pass
Click on a Google Calendar event	The individual item screen should be displayed showing the pressed event	Pass