**Mobile App for the University**

Final Report for CS39440 Major Project

*Author*: Mark Lewis ([mal60@aber.ac.uk](mailto:mal60@aber.ac.uk))

*Supervisor*: Dr. Bernie Tiddeman (bpt@aber.ac.uk)

6th March 2016

Version 1.0 (Draft)

This report is submitted as partial fulfilment of a BSc degree in  
Computer Science (G401)

Department of Computer Science

Aberystwyth University

Aberystwyth

Ceredigion

SY23 3DB

Wales, UK

**Declaration of originality**

In signing below, I confirm that:

* This submission is my own work, except where clearly indicated.
* I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
* I have read the regulations on Unacceptable Academic Practice from the University’s Academic Quality and Records Office (AQRO) and the relevant sections of the current Student Handbook of the Department of Computer Science.
* In submitting this work I understand and agree to abide by the University’s regulations governing these issues.

Name …………………………………………

Date ……………………………………………

**Consent to share this work**

In signing below, I hereby agree to this dissertation being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name …………………………………………

Date ……………………………………………

**Acknowledgements**

I am grateful to…

I’d like to thank…

**Abstract**

Include an abstract for your project. This should be no more than 300 words.

**Contents**

Contents

[1. Background, Analysis & Process 6](#_Toc447547559)

[1.1. Background 6](#_Toc447547560)

[1.2. Analysis 7](#_Toc447547561)

[1.2.1. Deciding topics areas for the app 7](#_Toc447547562)

[1.2.2. Professional finish and feel 8](#_Toc447547563)

[1.2.3. Storage of data offline 9](#_Toc447547564)

[1.2.4. Longevity of the app 9](#_Toc447547565)

[1.2.5. App platform 10](#_Toc447547566)

[1.3. Process 11](#_Toc447547567)

[2. Design 12](#_Toc447547568)

[2.1. Overall Architecture 13](#_Toc447547569)

[2.1.1. Data Storage 13](#_Toc447547570)

[2.2. Detailed Design 14](#_Toc447547571)

[2.2.1. Class – AppData 14](#_Toc447547572)

[2.2.2. Class – MainMenu 15](#_Toc447547573)

[2.2.3. Classes containing a single layout resource 15](#_Toc447547574)

[2.2.4. Classes containing multiple layout resources - Accommodation 16](#_Toc447547575)

[2.3. User Interface Design 16](#_Toc447547576)

[2.4. Other Relevant Sections 16](#_Toc447547577)

[3. Implementation 17](#_Toc447547578)

[4. Testing 18](#_Toc447547579)

[4.1. Overall Approach to Testing 18](#_Toc447547580)

[4.2. Automated Testing 18](#_Toc447547581)

[4.2.1. Unit Tests 18](#_Toc447547582)

[4.2.2. User Interface Testing 18](#_Toc447547583)

[4.2.3. Stress Testing 18](#_Toc447547584)

[4.2.4. Other Types of Testing 18](#_Toc447547585)

[4.3. Integration Testing 18](#_Toc447547586)

[4.4. User Testing 18](#_Toc447547587)

[5. Critical Evaluation ~2000 19](#_Toc447547588)

[6. Appendices 20](#_Toc447547589)

[A. Third-Party Code and Libraries 20](#_Toc447547590)

[B. Ethics Submission 21](#_Toc447547591)

[C. Code Samples 22](#_Toc447547592)

[Annotated Bibliography 23](#_Toc447547593)

# Background, Analysis & Process

This section should discuss your preparation for the project, including background reading, your analysis of the problem and the process or method you have followed to help structure your work. It is likely that you will reuse part of your outline project specification, but at this point in the project you should have more to talk about.

**Notes:**

* **All of the sections and text in this example are for illustration purposes. The main Chapters are a good starting point, but the content and actual sections that you include are likely to be different.**
* **Look at the document on the Structure of the Final Report for additional guidance.**

## Background

What was your background preparation for the project? What similar systems did you assess? What was your motivation and interest in this project?

Prior to starting this project I had no previous experience with the Android SDK, therefore one of my first tasks was to familiarise myself with it. I started by visiting the site for android developers [1], where I was able to find tutorials on many aspects of android development as well as tips for new developers on getting started.

Even though I had no previous experience of developing android applications I still chose this project as it was an area of the course which I wanted to learn more about. I have studied Java during my undergraduate studies and when I learnt that android applications are written in Java I thought it would be something which I could use skills I had gained from studying Java. I also like the design aspects of systems and producing an application at the end of a process which you can use and interact with.

The first tutorial I followed walks you through creating a basic app in Android Studio; the official IDE for Android. The tutorial includes setting up Android Studio, creating an application with a simple user interface and starting another activity. The tutorial walks you through the basic functions of Android Studio and gives you a basis to start developing your own application.

This Android developer’s tutorial introduced me to the basic functions and structures of android. I learnt how to add items such as buttons and textboxes using xml and how to link the layout xml files to java classes. Another lesson in the tutorial was to move between android activities, a core function needed when developing applications.

As this was my first time using Android Studio it gave me a chance to use the UI designer, a feature of the IDE which would be heavily used to design and implement layouts in my application. Creating layouts can be done using the device preview screen or programmatically using xml code. Initially I found the xml code confusing as it is something I have only briefly used. Therefore I focused my attention on designing pages using the device preview screen.

There was a two week period between the start of the project and my first meeting with the Marketing Department. This meant I had two weeks where I didn’t have a specification or list of requirements wanted by the Marketing Department. Therefore I decided to conduct some research into apps published by other higher education institutes in the United Kingdom.

I noticed that many universities had published apps, a large proportion of which were aimed at current students and only a small number of universities were also targeting potential students. This meant I only had a dozen apps to get an idea about the content, layout and functions which would be the most useful in my app.

I spent some time studying these apps in more detail, taking notes about the topics they covered, any libraries they were using and also the layout of content. All this information helped me build a picture of the type of data and layout I could potentially include in my app.

I also studied the webpages on the Aberystwyth University website to view the range of content which could be covered in the app. Many of the webpages had a natural flow between each other, pages which were heavily advertised and promoted were the pages which would most appropriate for the app as the information on them was important for recruiting students.

I could then combine the information about competitor’s apps and content available on the University’s website in order to start building a requirements list to present to the Marketing Department.

After my meeting with the Marketing Department I found that their ideas were very similar to what I had been working towards. They were keen to find out what their competitors were doing and if there was a general theme they wanted to match it.

## Analysis

Taking into account the problem and what you learned from the background work, what was your analysis of the problem? How did your analysis help to decompose the problem into the main tasks that you would undertake? Were there alternative approaches? Why did you choose one approach compared to the alternatives?

There should be a clear statement of the objectives of the work, which you will evaluate at the end of the work.

In most cases, the agreed objectives or requirements will be the result of a compromise between what would ideally have been produced and what was felt to be possible in the time available. A discussion of the process of arriving at the final list is usually appropriate.

From my background work I had learnt that many other institutes were using their prospective student apps in two ways; either to show off courses and life at their University or to view publications such as magazines and prospectuses. From meetings with the Marketing Department I knew they wanted me to develop an app similar to the first style as they had been in conversation with a company to develop a system which they could distribute publications on.

### Deciding topics areas for the app

I knew my main task was to decide which topics I would address in the app. I was able to use the knowledge of competitor’s apps and an understanding of the available content on the University’s website gained during my background research to start putting together some ideas.

Due to the University’s website containing so much content aimed at prospective students the number of topics and possible combinations available was very large. This meant it is important to not fill the app with too much content and select information which is going to be the most informative and useful for the users of the app.

A final list of ten topics was decided to be covered in the app. Listed here;

* Undergraduate Courses
* Postgraduate Courses
* Scholarships and Funding
* Study Abroad
* Accommodation
* Open Days
* Virtual Open Days
* Getting Here

The ten topics were grouped into three categories; Academic Information, Student Life and Open Days.

The list of topics was generated by comparing competitor’s apps to understand the information which is being conveyed and also checking whether similar information is available for use from the Aberystwyth University website. Once the list had been generated it was approved by the Marketing Department, this was an important aspect of the requirements gathering phase as it meant the app was going to address information which the customer wanted.

There were other topic areas which could have been covered in the app which didn’t make the final list. One idea was to try and base the app around the University’s Online Virtual Open Day [2]. The Online Virtual Open Day gives potential students a taste of the University without having to come to Aberystwyth. One feature of the Online Virtual Open Day which I looked at implementing in the app was the ability to chat with lectures and support staff via an instant chat function.

I knew that the University used an instant messaging service called LibraryH3lp [3] which ran on the XMPP chat protocol. There was no API or support for building an Android App on their developer’s site so I set out to find a tutorial which I could follow to see if I could get a prototype working.

I found a tutorial on TutorialsFace [4] which I started to follow before the final list of requirements was decided. I hadn’t managed to get a working prototype of the instant chat facility before it was ruled out as a feature of the App, but it did further my understanding of using Android Studio.

Another content area which was not included was information about service departments within the University. The university has departments which offer support to students with non-academic issues such as Student Support, Information Services, Campus Services and many others. I had a discussion with the Marketing Department as to whether we should include content on these departments, as I thought it may be beneficial for educating users about other aspects of the University, but it was decided that it was not needed and would not be crucial information for recruiting students.

### Professional finish and feel

If the end product is of a high quality then it could be adopted by the Marketing Department and published on the App Store. This meant another task was to create an application which is professional enough to represent the University.

This means the app needs to have an appearance which is aesthetically pleasing as well as being efficient and easy to use. There are many interface design options available when creating an app, therefore it was important to select a style which could display the large amount of content in a readable manner as the content will be displayed on a variety of devices with different screen sizes.

The interface also needs to be consistent throughout the app. This means pages which have a similar amount of content or structure need to use the same layout. This will help to give the app a more professional look as well as improving the user’s ability to navigate through the app.

When deciding on layout types it is important to choose styles which can be re-used and can be shared on pages with similar structures. If styles are picked which are too specific then the app could end up containing multiple layout types which will damage the appearance.

Another aspect of design which needs to be taken into consideration is the colour scheme. In order to look professional and represent the University the app needs to adopt the University’s colour scheme.

An app is not of a high quality just because of its appearance, it also needs to behave as the user expects in order to fall into the category of a professional app. The final app needs to be sturdy and not prone to crashing, by putting in methods to ensure proper error checking and handling this can be achieved.

### Storage of data offline

Another aspect of the app the Marketing Department were keen to see was being able to access the content at any time, making the app similar to a prospectus in your pocket which you could refer to wherever you are.

Therefore the app needs to download and save content for each activity of the app somewhere on the device and then be able to load it back in when needed. There are multiple options which can be chosen for this feature and this created a new task of choosing the option which would be most beneficial to the application.

The amount of data which is within the app is quiet large and therefore storing it on the device was chosen to reduce the load time between activities as well as improving the likelihood of a user using the app. If the user is going to have to switch on their mobile data to view the use the app it may discourage them from doing so as data charges can be quiet expensive.

### Longevity of the app

In order for the app to be reusable each year it is important that there is some way in which the content on the app is updated. If the app needs to be re-written each year to contain the latest information then it has been poorly developed. For example it would be a bad idea to hard code the content into the app on creation as it couldn’t be updated at a later date without re-building the whole app.

This means another task was to find a way to keep the content of the app up to date. There are multiple ways in which this can be achieved, using a content management system is one option. The University already has a content management system for its website, if it is able to work with an Android application then it would be a good option; if not then one would have to be created specifically for the app.

Another option is to find if there is a way to scrape data from websites and use it within the app, this requires a small amount of research into Java libraries. If there is the possibility of using a library to do this then it would be extremely helpful as it would save me having to write my own methods to do so.

I discovered that the University’s content management system didn’t work with Android applications, ruling out the option of using it to populate the app with content. I then turned my attention to whether or not I could create my own content management system or find an alternative. I looked into other content management systems online and found none which were suitable for my project.

There was the possibility of creating my own content management system. It would need a web interface in order for the Marketing Department to input the content and a database to store the content which they had uploaded. The data could then be read straight from the database and into my Android application. One drawback to this method was that the topics which were going to be covered in the app already had content available on the University’s website. This meant that the data would just duplicated by copying it straight from the site into the content management system.

Therefore I looked for a way of extracting data from the website and using it within my application. I found a Java library called Jsoup [5] which is a HTML parser and it looked as if it was able to fit my needs. You are able to get content from a given URL or from inputting a file, something which I would need in order to edit the data if the device was online and also to load content which had been saved for offline use.

Once a webpage is parsed as a document you are able to manipulate the data by selecting HTML attributes, elements and text. Once the content you require has been selected you can then output it as either plain text or wrapped with its existing HTML code. This means it could be displayed in either Android text views or web views.

### App platform

Developing mobile apps for multiple operating systems can be done in different ways. One way is to build an app which is cross-platform compatible, this usually involves building the app in HTML and using a program which will wrap code around your app in order to display it on each different operating system. Another alternative is to develop the app so it is native to one operating system and repeat this process for each operating system you want to target.

In an ideal situation the app would be useable on all mobile operating systems. Therefore I looked into the possibility of using PhoneGap [5], a program which allows you to build cross-platform apps using HTML, CSS and JavaScript. I did some research into PhoneGap and found that it had mixed reviews online, an article on Roadfire Software’s [6] website debated the pros and cons of using PhoneGap verses developing a native app.

The article suggests there is a noticeable drop in performance and many user interface bugs when developing an application using PhoneGap. There is also some design issues due to mobile differences, for example android phones have a back button whereas iOS does not, and this is something you would have to overcome in your app.

Another reason why I decided to develop an app native to Android is because I had experience using Java and had found some Java libraries which I intended to use. I felt I was more likely to produce an app of higher quality due to previous experience in developing Java then I would by starting from scratch with PhoneGap.

I also had a discussion with my Major Project Supervisor Bernie about which option may be the best. We decided that it was best to target a native Android App and if the finished product was good enough to be used then the Marketing Department could fund someone to build the app in iOS too.

A final factor which influenced choosing Android was its larger market share than any other mobile operating system available. According to IDC, at the second quarter of 2015 82.8% of mobile units shipped were Android devices [8]. This should result in the app having a larger target audience then choosing an alternative operating system.

## Process

You need to describe briefly the life cycle model or research method that you used. You do not need to write about all of the different process models that you are aware of. Focus on the process model that you have used. It is possible that you needed to adapt an existing process model to suit your project; clearly identify what you used and how you adapted it for your needs.

The life cycle model I have used for this project is based around feature driven development as it seemed like a perfect fit for both this project and myself. An important aspect of feature driven develop is the features; gathering them and planning how you will implement them. Features are aspects of the system which are important to the customer, they are used to gather requirements and plan the design.

Initially there were no requirements set for this project and I had to meet with the Marketing Department in order to find what they wanted from the application. This helped me to work towards building a feature list and also developing an overall model of the system.

Once a feature list has been created you can then start the next phase of planning by feature. Traditionally in feature driven development you assign features as classes to programmers in your development team, as this is an individual project the ownership of all the features belongs to myself. This then leads on to the next stage of designing by feature.

During the design by feature phase a design package is created for all features. The lead programmer meets with each programmer to create sequence diagrams for their features. Once all features have been assessed then the lead programmer selects a group of features to be developed over the coming iteration. These features are then developed and promoted to the main build after they have passed unit tests.

For my project this is the stage where I have decided to make an alteration to feature driven development. Instead of working in iterations I decided to work through my features in a progressive order. I tried to start off with features which were necessary in order for other features to be implemented. For example implementing the offline storage of content is a simpler task if you have already implemented the content as you have a better understanding of how the content can be stored.

Feature driven development has a strong emphasis on project tracking. Features are split into milestones, there are six milestones per feature. This helps you to track the progress of the feature you are currently working on as well as the overall progress of the project. I felt this aspect of feature driven development allowed me to continue to be motivated during the project.

# Design

You should concentrate on the more important aspects of the design. It is essential that an overview is presented before going into detail. As well as describing the design adopted it must also explain what other designs were considered and why they were rejected.

The design should describe what you expected to do, and might also explain areas that you had to revise after some investigation.

Typically, for an object-oriented design, the discussion will focus on the choice of objects and classes and the allocation of methods to classes. The use made of reusable components should be described and their source referenced. Particularly important decisions concerning data structures usually affect the architecture of a system and so should be described here.

How much material you include on detailed design and implementation will depend very much on the nature of the project. It should not be padded out. Think about the significant aspects of your system. For example, describe the design of the user interface if it is a critical aspect of your system, or provide detail about methods and data structures that are not trivial. Do not spend time on long lists of trivial items and repetitive descriptions. If in doubt about what is appropriate, speak to your supervisor.

You should also identify any support tools that you used. You should discuss your choice of implementation tools - programming language, compilers, database management system, program development environment, etc.

Some example sub-sections may be as follows, but the specific sections are for you to define.

The first stage of design on a feature driven development project is to design by feature. As I already had a feature list and had chosen to work through the list sequentially each feature’s design phase happened separately rather than as a group of features which were to be implemented over the coming iteration. This allowed the design to be much more dynamic as I was able to learn from mistakes during the design of earlier feature’s and use similar designs for resembling features.

Implementing features one after another also meant that many of the original designs were no longer adopted. In most cases a design would be made which would allow the feature to be built into the application, then when it came to adding new features which relied on prior features the old design would no longer be viable. This process meant it was important not to become too engrossed in creating concrete designs as they likely would change.

One aspect of design which could have been overlooked on this project was the user interface. There were no requirements set for the appearance of the app as the Marketing Department had given me full freedom to create a layout which I believed worked best for the task. For this reason it is easy to forget that the user interface is also a feature as it wasn’t an aspect of the project which was highly valued by the customer.

Generally each feature is assigned its own class and depending on the complexity of the feature the amount of methods it contains will vary. Some features are considered as activities in an android system, this means the user is able to interact with them and as a result they require their own layout file. Where activities are very much alike it is possible to re-use layout files in order to save time and space, but there may be special occasions where similar activities require different layouts.

## Overall Architecture

Each mobile app will have some form of home screen, depending on the apps function this may be used once or multiple times. I knew that my app would have multiple pages displaying content and would therefore require a main screen which could be used to link all of the pages together.

From the menu page there needs to be a way to access the detailed content on the ten topic areas which have been selected during the requirements gathering process. Therefore each topic area will have its own page within the app, and some topics may require multiple pages; this is something which will be discussed further in the detailed design section of this document.

Before the user can view the content on the application it must first be downloaded and stored on the device. To do this I planned to implement a splash screen as they are commonly used in applications and games whilst they load. I chose to use a splash screen as they can give an app a professional feel if executed correctly. Alternatively the app could have loaded the menu screen immediately but if a user tried to navigate to view content they would be unable to do so.

From when the app is launched it should first check if there is data saved on the device. If there is no data downloaded on the device then the device must be online in order to fetch content. Depending on the result of these checks different methods need to run; download content, proceed to menu or display error for no internet connection.

Once the data has been successfully downloaded the user may proceed to the menu screen, where they are able to view the subject areas within the app. The design of the user interface can be make or break for a mobile application, there is a section of this document outlining the decisions I made in terms of design as well as justification for the mentioned decisions.

On selecting a content area from the menu the user should be taken to a section about that topic. Depending on the topic the section may be a singular page within the app, or it may contain its own menu where the user can find out more specific data. For example the undergraduate and postgraduate courses are likely to broken down into departments and individual course.

### Data Storage

The data used to populate the pages within the app needs to be stored within a data structure. There are five possible storage options listed on the Android Developer’s website [9], I used this information in order to choose the storage method which I felt was most appropriate for the amount of files which would need storing.

Two of the options were easily dismissed straight away; shared preferences and network connection. Shared preferences is generally used to store user choices such as ringtones and are stored as objects on the device. These objects can then be loaded in to the app to set it up to the users liking. As the data which needs to be stored on the phone is consistent across all user’s instances of the app this storage method wouldn’t be required.

Network connection involves the device being online in order to connect to a web-based service to store and load the data. Due to the requirement for offline storage within this app, network connection was removed as a potential option.

That left three conceivable routes. The first was a SQLite database on the device which is private to the application. A SQLite database would be useful if the app required large amounts of normalised data, however the data needed could be stored as string values and would not desire a complex data structure.

Therefore I was left to choose between storing the data on the devices internal or external storage. Ultimately I chose to use the phones internal storage instead of any external storage such as SD cards the device may have. One reason for choosing internal storage is because not all devices have external storage, and users don’t always use external storage even if their device supports it. Secondly the data which was to be stored should not be too large in capacity and should be able to be stored on the device. The final feature was that data stored internally is only accessible by the app which creates it, unlike data stored externally which is available to any person or app. This was not a major factor when choosing between the two storage options as the data stored is not sensitive, but it does stop the user changing or deleting files which may cause the application to become unstable.

## Detailed Design

This section contains added detail on the application’s design, where appropriate classes will be discussed individually, but in some cases classes were very similar and therefore do not require individual deliberation.

### Class – AppData

This is the first class to be ran when the application is launched, the plan for this class is to download data onto the device. An initial check needs to be done in order to test the internet connection on the device, from there the data can be downloaded.

To stop data being downloaded each time the app is launched and connected to the internet another check needs to be done; whether or not there is data already on the device. This check can be combined with the check for internet connection as both will return a Boolean answer. There are three potential answer combinations which will require action; data downloaded true, internet connection true and downloaded data false, finally internet connection false and data downloaded false. We can ignore the status of the internet connection at all times when data downloaded equals true, as the device does not need to be online to download data if it already exists.

This can be one method which is run in AppData’s onCreate after the layout file has been linked and any elements initiated.

If there is no data downloaded as well as no internet connection then an error message must be displayed. There is no method required to do this, as a single line of code can produce a message on the screen to tell the user to enable a connection.

When there is no data downloaded but the device is connected to the internet the class should run a method to download content. This is likely to be the most complex method within the application as it needs to connect to each page, manipulate the data and finally save it on the device. As this method is going to use the device’s network it must be run as part of a thread or AsyncTask.

This method will get each page of the apps content and save it as a separate file for the corresponding page. This means a method is required to download each pages content. These methods will require a URL for the webpage they need to connect to, then using methods from the Jsoup library content can be selected in order to be saved.

I would like each method to download only the content required for the app in order to minimise the amount of storage the application will use. This should also quicken up the process of loading data into the app as there will not be any data manipulation executed as pages are load.

Each method for generating the applications content will need to save the data. For each method whose output is in the same format they can make use of the same save method. This method will require an input variable in order to name the files.

If the content is outputted in different formats then multiple save methods may be required. For example a different save method for saving strings and saving arrays.

### Class – MainMenu

This class runs the code behind the menu’s user interface. Regardless of the layout of the final menu screen this class has one main function, to provide navigation to the applications content. Methods will be required for each page the menu links to, these will be called when items or buttons are selected in order to start the next activity.

From the menu screen I also wanted the user to be able to re-download the apps content. This means that if the University webpages used to generate content get updated then users are still able to download the latest content onto their phone. This method could also be used if there is any corruption of data within the app, by replacing the current corrupted content.

In order to download the latest data the files which are currently on the device must first be deleted. After the files have been deleted there needs to be a way in which the app can download the new content, to do this another method could be implemented within the menu class. But at its current state the app is the same as if it has just been installed on the device, therefore we can re-use the code which runs on start up within the AppData class to generate content. This means instead of creating a new method an intent can be called to start the AppData class again.

The final method used within this app is to override Android’s handling of pressing the back button. Ordinarily if the back button is pressed then the app will move to the current activity’s parent activity which is declared in the Android Manifesto, if there is no parent activity then it will rewind to the last used activity. As the menu is the applications main page there is no need for a parent activity and also no need to move back to the splash screen. For these reasons the back button can be disabled.

### Classes containing a single layout resource

Within the application five of the topic areas only require one page to display their content; study abroad, students union, sports & societies, open days and virtual open days. This means these five classes all share similar structure and methods, to avoid repetition of descriptions they have been categorised together.

Present in all activities within an Android application is an onCreate method and these five activities are no different. The onCreate method sets the activity’s layout by linking the appropriate XML file and initialising any interactive elements. Next the content needs to be loaded onto the screen, to do this the data must first be loaded from a file into a class variable, this requires a separate method. There will also need to be some error handling when calling the loading method in the event of a file not found exception.

The method to load the content from a file is the same for these five classes, it can also be re-used in other classes in the application where the data is to be loaded in as a string. In order to load in the file the method will require the targeted filename as an input, this could either be taken as a variable as it is run or coded into each class’s version of the app.

### Classes containing multiple layout resources - Accommodation

The remaining five topic areas all have a larger amount of content and therefore are unable to be displayed on a single page, resulting in each topic area requiring multiple classes.

The first topic area is for accommodation. An appropriate design for this section would be to first have a list of available accommodation which when clicked on directs the user to a page with details about the selected accommodation block.

For each specific accommodation block on the University accommodation website [10] there are up to eight sections of content. Displaying all eight sections would potentially be too much content to view within a small mobile device and therefore I decided to reduce the number of sections included. I chose to exclude the following sections; what’s in your shared access, location, gallery and tour. There were multiple reasons for not including these sections, for example the location and tour sections included map walkthroughs and were going past the level of detail required in the app. This left me with the following sections; overview, facilities & services, what’s in your room and fees.

The first class I would require would be for the list of accommodation. This class needs a list of accommodation blocks as well as the content for the accommodation blocks; these can be stored as arrays. Therefore it is able to re-use the load from file method which is also present in the five classes containing single layout resources.

The list of accommodation blocks needs to be clickable so a second method is needed to handle the items being clicked. This method must start the accommodation detail activity, passing it the correct content for the chosen accommodation block. This can be done by putting the accommodation block’s content in to an intent and then starting the activity.

A second class is required to display the content specific to each accommodation block. This class will be in control of receiving the pages content, splitting it up into each section and arranging it. The data which has been sent from the intent in the first class will need to be stored as a class variable for use later on, there is no need for a separate method to do this.

There will be a method required in order to separate the received data into the four prescribed sections. This method will make use of Jsoup library functions, parsing string data into a document, selecting the needed content and finally returning four separate outputs. These outputs can be displayed immediately as part of the method, though it is a better idea to store them as class variables and then assign them to android views. This should hopefully reduce data loss when switching between sections.

The exact layout of the accommodation details will be discussed further in the user interface design section of this document. Depending on the chosen layout style there may well be need for a class for each of the four sections.

## User Interface Design

## Other Relevant Sections

# Implementation

The implementation should look at any issues you encountered as you tried to implement your design. During the work, you might have found that elements of your design were unnecessary or overly complex; perhaps third party libraries were available that simplified some of the functions that you intended to implement. If things were easier in some areas, then how did you adapt your project to take account of your findings?

It is more likely that things were more complex than you first thought. In particular, were there any problems or difficulties that you found during implementation that you had to address? Did such problems simply delay you or were they more significant?

You can conclude this section by reviewing the end of the implementation stage against the planned requirements.

# Testing

Detailed descriptions of every test case are definitely not what is required here. What is important is to show that you adopted a sensible strategy that was, in principle, capable of testing the system adequately even if you did not have the time to test the system fully.

Have you tested your system on ’real users’? For example, if your system is supposed to solve a problem for a business, then it would be appropriate to present your approach to involve the users in the testing process and to record the results that you obtained. Depending on the level of detail, it is likely that you would put any detailed results in an appendix.

The following sections indicate some areas you might include. Other sections may be more appropriate to your project.

## Overall Approach to Testing

## Automated Testing

### Unit Tests

### User Interface Testing

### Stress Testing

### Other Types of Testing

## Integration Testing

## User Testing

# Critical Evaluation ~2000

Examiners expect to find in your dissertation a section addressing such questions as:

* Were the requirements correctly identified?
* Were the design decisions correct?
* Could a more suitable set of tools have been chosen?
* How well did the software meet the needs of those who were expecting to use it?
* How well were any other project aims achieved?
* If you were starting again, what would you do differently?

Such material is regarded as an important part of the dissertation; it should demonstrate that you are capable not only of carrying out a piece of work but also of thinking critically about how you did it and how you might have done it better. This is seen as an important part of an honours degree.

There will be good things and room for improvement with any project. As you write this section, identify and discuss the parts of the work that went well and also consider ways in which the work could be improved.

Review the discussion on the Evaluation section from the lectures. A recording is available on Blackboard.

# Appendices

* 1. Third-Party Code and Libraries

If you have made use of any third party code or software libraries, i.e. any code that you have not designed and written yourself, then you must include this appendix.

As has been said in lectures, it is acceptable and likely that you will make use of third-party code and software libraries. The key requirement is that we understand what your original work is and what work is based on that of other people.

Therefore, you need to clearly state what you have used and where the original material can be found. Also, if you have made any changes to the original versions, you must explain what you have changed.

As an example, you might include a definition such as:

**Apache POI library** – The project has been used to read and write Microsoft Excel files (XLS) as part of the interaction with the client’s existing system for processing data. Version 3.10-FINAL was used. The library is open source and it is available from the Apache Software Foundation [5]. The library is released using the Apache License [6]. This library was used without modification.

* 1. Ethics Submission

This appendix includes a copy of the ethics submission for the project. After you have completed your Ethics submission, you will receive a PDF with a summary of the comments. That document should be embedded in this report, either as images, an embedded PDF or as copied text. The content should also include the Ethics Application Number that you receive.

* 1. Code Samples

This is an example appendix. Include as many appendices as you need. The appendices do not count towards the overall word count for the report.

# Annotated Bibliography

This final section should list all relevant resources that you have consulted in researching your project. Each reference should also include a brief annotation.

1. Sylvia Duckworth. A picture of a kitten at Hellifield Peel. <http://www.geograph.org.uk/photo/640959>, 2007. Copyright Sylvia Duckworth and licensed for reuse under a Creative Commons Attribution-Share Alike 2.0 Generic Licence. Accessed August 2011.  
     
   This is my annotation. I should add in a description here.
2. Mark Neal, Jan Feyereisl, Rosario Rascunà, and Xiaolei Wang. Don’t touch me, I’m fine: Robot autonomy using an artificial innate immune system. In *Proceedings of the 5th International Conference on Artificial Immune Systems*, pages 349–361. Springer, 2006.   
     
   This paper…
3. W.H. Press et al. *Numerical recipes in C*. Cambridge University Press Cambridge, 1992.  
     
   This is my annotation. I can add in comments that are in **bold** and *italics*and then further content.
4. Various. Fail blog. <http://www.failblog.org/>, August 2011. Accessed August 2011.  
     
   This is my annotation. I should add in a description here.
5. Apache Software Foundation (2014) “*Apache POI - the Java API for Microsoft Documents*” (Online) Available at: <http://poi.apache.org> Accessed: 14th March 2014.
6. Apache Software Foundation (2004) “Apache License, Version 2.0” (Online) Available at: <http://www.apache.org/licenses/LICENSE-2.0> Accessed: 14th March 2014.