Mobile Platform Development

Shaymaa Al-Sharbati

S1512130

Coursework

2019

Table of Contents

[Design Report 2](#_Toc5956592)

[About the Project 2](#_Toc5956593)

[What this chapter will include 2](#_Toc5956595)

[Android Version 2](#_Toc5956597)

[Functional Requirements 4](#_Toc5956598)

[Non-Functional Requirements 4](#_Toc5956599)

[Tools that will be used 5](#_Toc5956600)

[The Design Stage 5](#_Toc5956601)

[Developing Stage 9](#_Toc5956602)

[HCI Principles 10](#_Toc5956603)

[Testing 11](#_Toc5956604)

[Importance of Testing 11](#_Toc5956605)

[Why is Testing important overall? 11](#_Toc5956606)

[Blackbox Testing 12](#_Toc5956607)

[White Box Testing 12](#_Toc5956608)

[Usability Testing 13](#_Toc5956609)

# Design Report

## About the Project

## The desired goal for the Mobile Platform Development coursework project is to create an Android Mobile Application regarding information about Earthquakes across the British Isles and beyond from the British Geological Survey Records. This mobile application would show users, in a visually pleasing manner, summary statistics of such records.

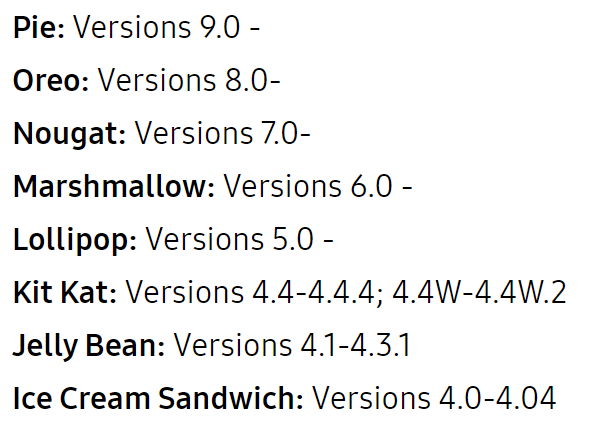
## What this chapter will include

## The following section will outline what will be covered within this section;

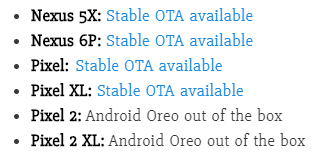
* Android Version
* Functional Requirements
* Non-Functional Requirements
* Tools that will be used
* HCI Principles

## Android Version

One of the Project Specifications was to ensure that the App that is to be created could run on Android 6.0 or later. *Figure 1* illustrates the names of the Android versions – from Version 4 to Version 9.

From *Figure 1*, we can see that Android Version 6.0 is acceptable,  
known as Marshmallow – as well as the other 3 versions: Version 7  
(Nougat), Version 8 (Oreo) and Version 9 (Pie).  
  
Once having researched this, in order to have a clear idea of what   
kind of device the application would be designed for – further  
research was conducted in order to determine this. This was crucial  
to be aware of, as an appropriate emulator must be selected within   
Android Studio to ensure that the application was running smoothly  
on both the screen ratio aspect, and the Android Version.

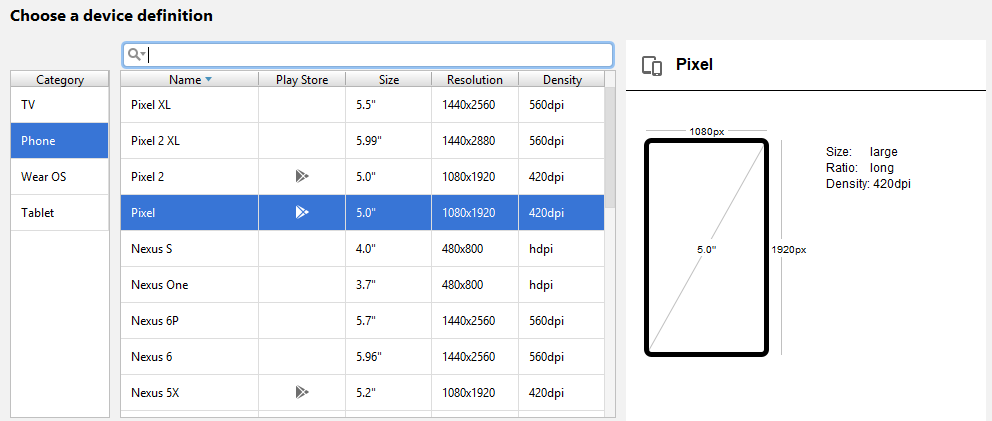
*Figure 1 – Android Versions*

 *Figure 2* illustrates Google smartphones, whom are known best for their speedy Android Updates. Amongst these smartphones is the Nexus 5X and Pixel 2 – with the Google Pixel 2 having Android Oreo (Version 8) straight out of the box.

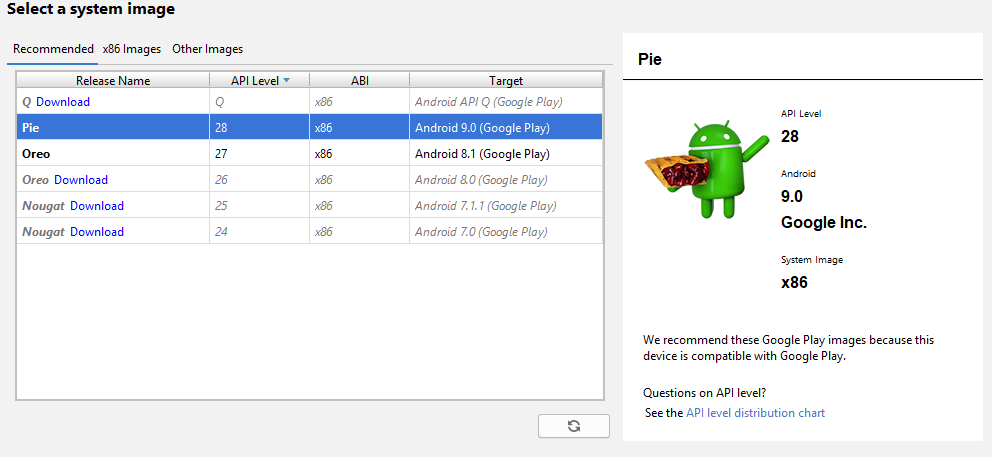
*Figure 2 – Google Smartphones*

Having researched and discovered this, the appropriate Emulator could then be selected within Android Studio. This would ensure two things:

* The Application that would be created would successfully run on Android Version 6 and greater, as requested by the client.
* The Application can be successfully tested through the use of an Emulator, without having the need to use a physical smartphone.

*Figure 3* illustrates the selection of Emulators within Android Studio – Google Pixel was amongst the many choices.

*Figure 3 – Available Emulators*

To double check, and confirm, that the research conducted about the Google Pixel in regard to the Android Version it could run – the Pixel was selected within Android Studio and as illustrated in *Figure 4,* the following results confirmed that the Pixel could run beyond Android Version 6.

*Figure 4 – Pixel Android Versions*

Thus, from this brief research, the Google Pixel was selected to be the Emulator within Android Studio for this project.

## Functional Requirements

When designing an Application, one must think about and clearly define the functional requirements in order to successfully define the functionality of the App itself. Functional requirements are essentially functions, or features, that must be included within the App itself in order to satisfy the clients needs as well as the Projects aims.

The Functional Requirements are as follows, in no particular order;

* Provide Users with the summary statistics of Earthquake data
* Allow Users to search
* Allow Users to enter a specific day to filter out Earthquake data
* Allow Users to enter a specific period of time to filter out Earthquake data
* Provide Users with the following information when searched: Most Northerly/Southerly/Westerly/Easterly earthquake, Largest magnitude Earthquake and Deepest/Shallowest earthquake
* Information that is being processed must update on a regular basis
* Allow Users to have the ability to view and use the app in a portrait and landscape mode

## Non-Functional Requirements

On the other hand, non-functional requirements must also be thought of and clearly defined in order to set goals to achieve when creating the App to ensure that the App is User Friendly, and the best it can be. Non-Functional requirements essentially define how well the App needs to function, compare to functional requirements – which focuses on what the App should do.

The three main Non-Functional requirements for this project are as follows, in no particular order, which will be explained briefly once introduced below;

* Performance
* Usability
* Scalability

**Performance**

App response time should be fast when loading pages, and when user is interacting with the App itself.

**Usability**

Usability of the App should be easy and clear to use. To learn how to use the App should be as simplistic as possible, with no need for a “manual” or “how to guide” on how to use it.

**Scalability**

The ability to handle large quantities of data from the html/XML page should be handled within the App.

## Tools that will be used

There are various software’s that will be used in order to create this project – and this section will briefly explain each tool, as well as the stage of the project it will be used in.

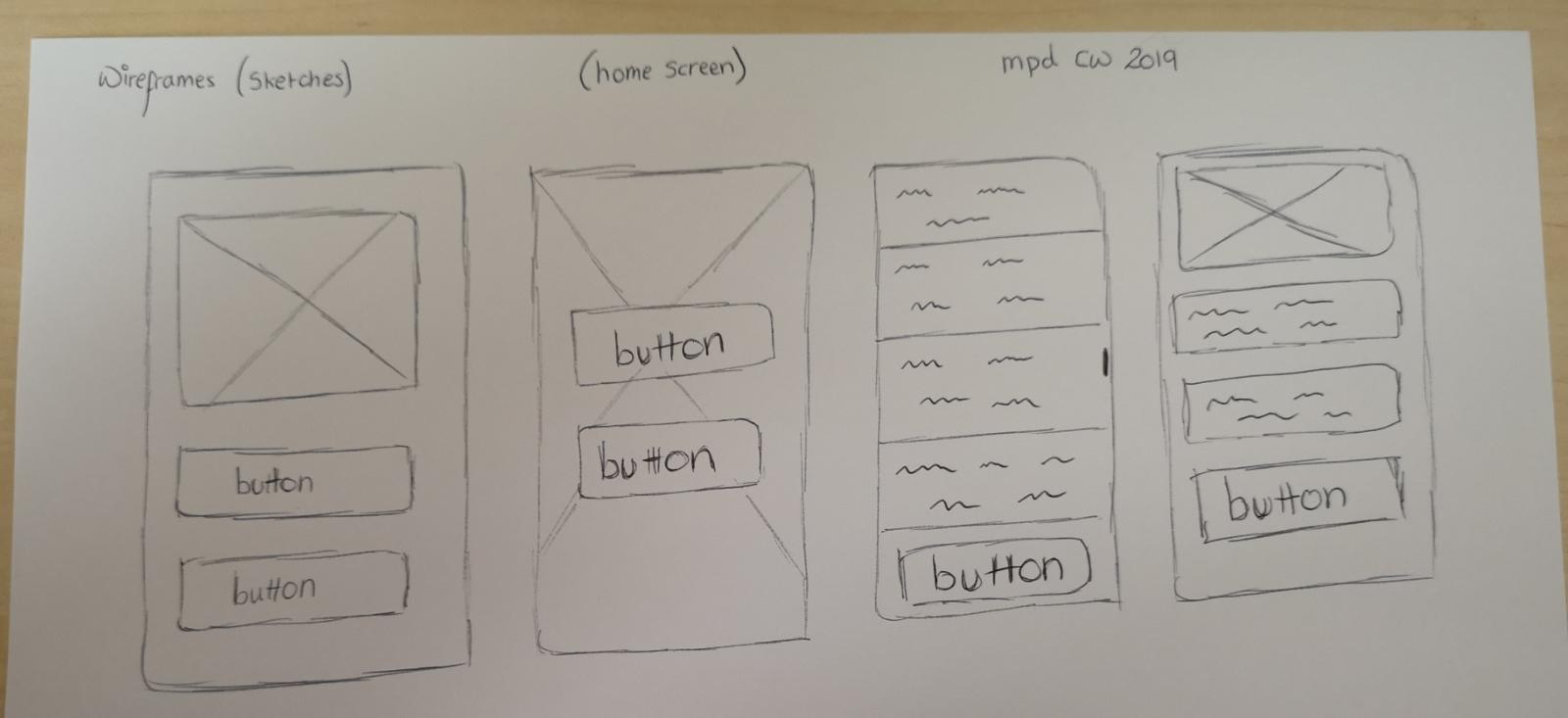
### The Design Stage

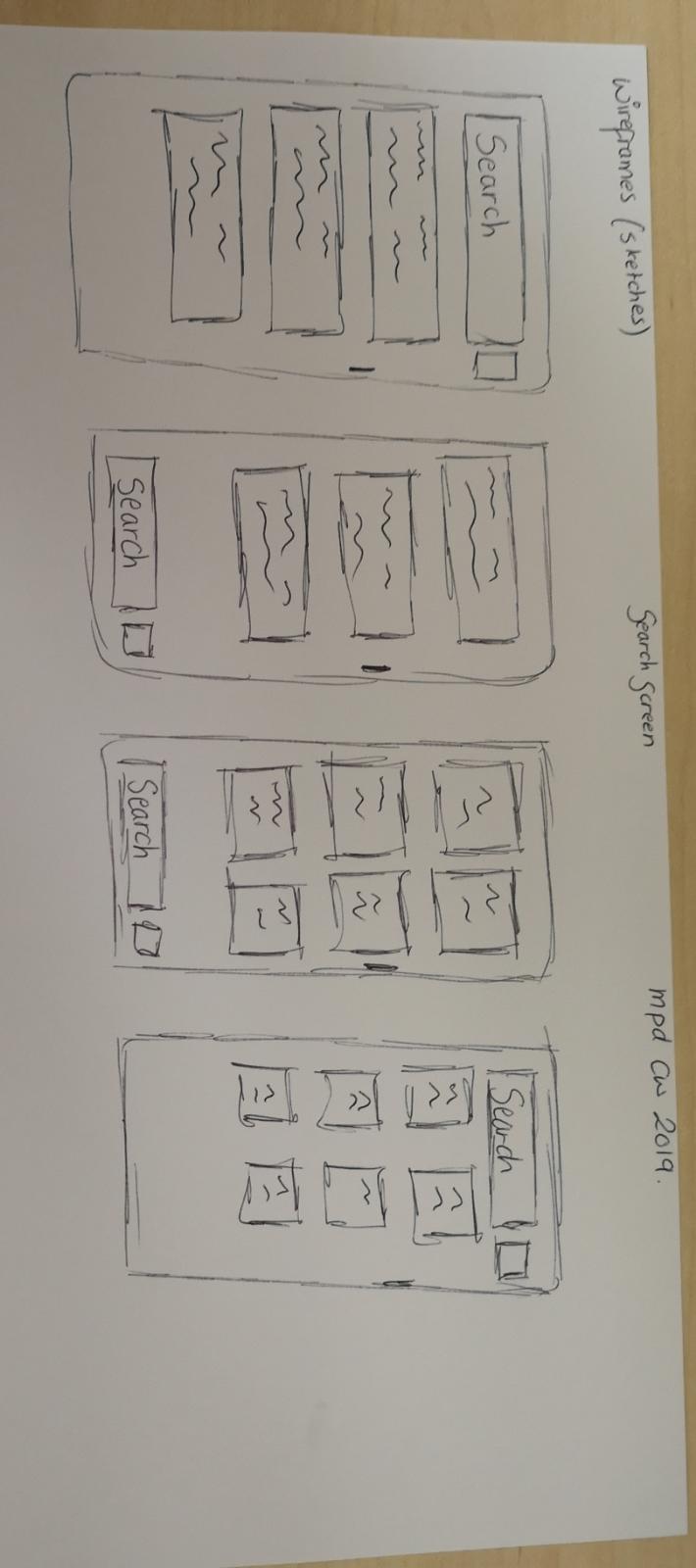
This section will explain the Tools used for the Design Stage of this project.

Pen and Paper

In order to successfully design and essentially build the App from the functional and non-functional requirements that were established as well as the research conducted, sketches must be made first in order to have a clear idea of the layout that will be used for the App – taking account both portrait and landscape modes.

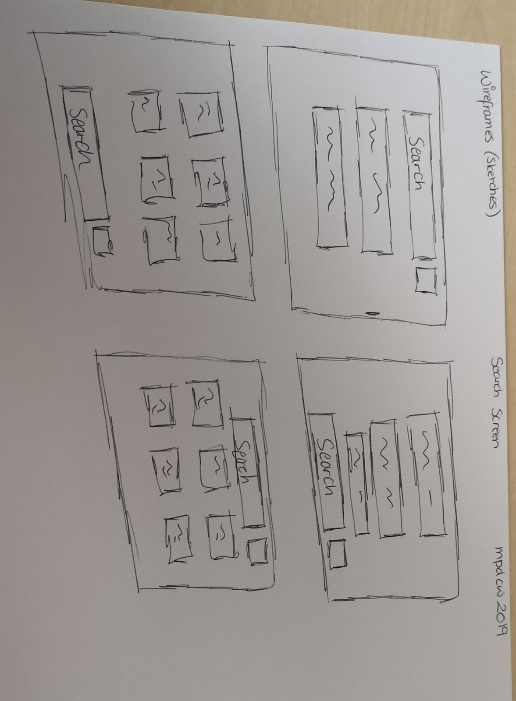
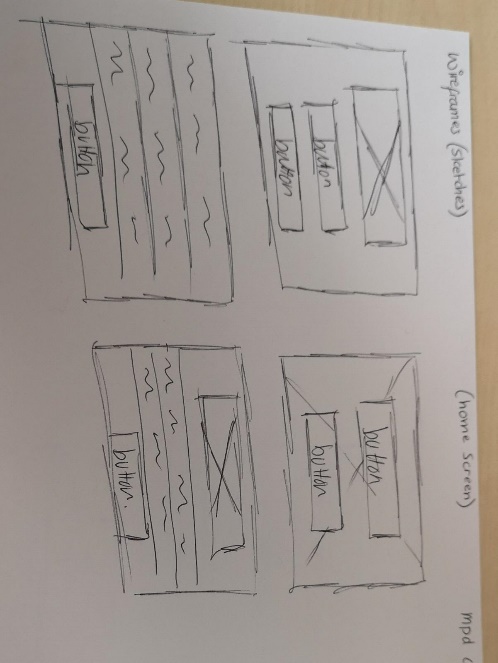
Sketches were first made from pen and paper – as this ensured a vast amount of ideas were looked at, before the best design ideas were combined and resulted in the final sketches for both portrait and landscape mode. *Figure 5, Figure 6, Figure 7 and Figure 8* illustrates the initial sketch phase – which consisted of sketching various ideas for both the portrait and landscape mode.





*Figure 5 – Portrait Home Screen sketches*

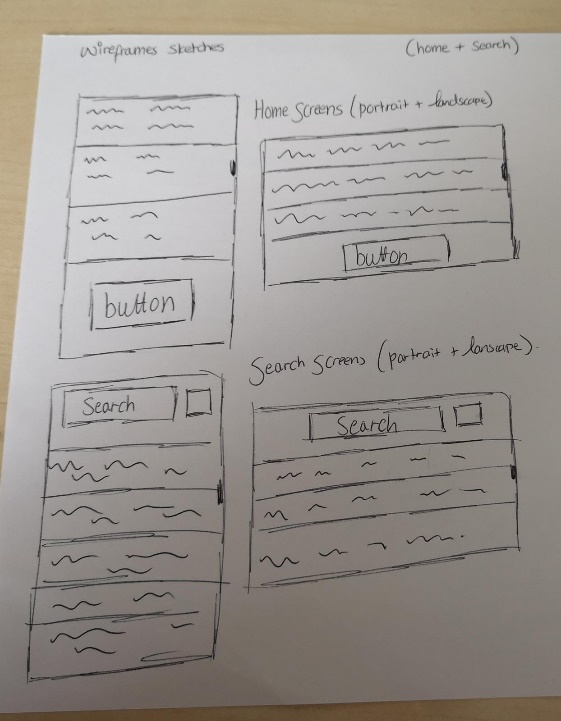
*Figure 6 – Portrait Search Screen sketches*



*Figure 8 – Landscape Home Screen sketches*

*Figure 7 – Landscape Search Screen sketches*

Once these sketches were created, a finally decision on the layout was made as illustrated in *Figure 9*, which resulted in the final decided screens for both the portrait and landscape mode of the App.



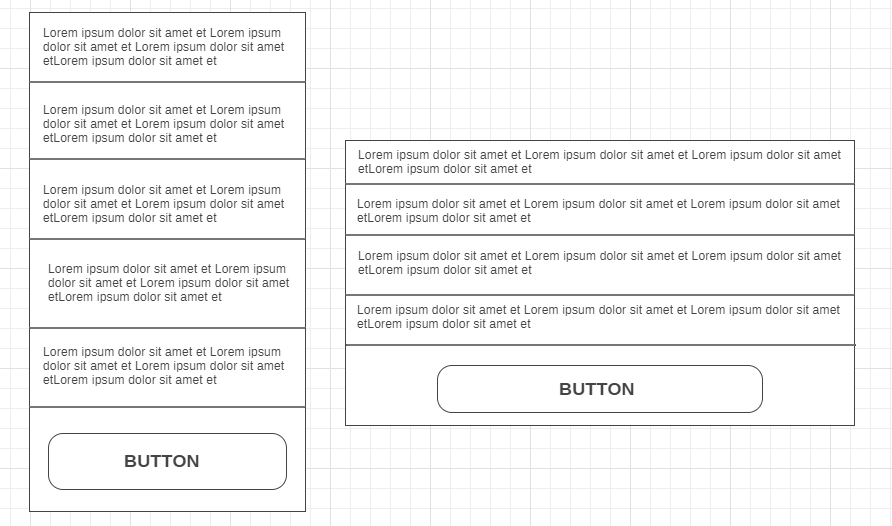
*Figure 9 – Final sketches for Home + Search Screen – both portrait and landscape sketches*

Wireframes

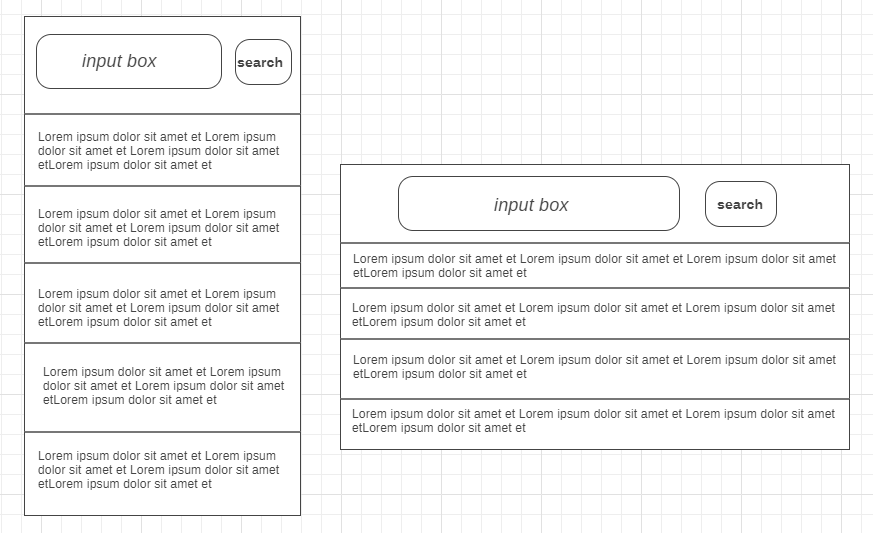
Once the initial and finalized sketches were decided upon for the App itself, Wireframes were then created. This is essentially an “electronic” version of the sketches – to provide a better idea and understanding of what the App would look like visually on a screen.

To create the Wireframes, Wireframe.cc (<https://wireframe.cc/>) was used in order to accomplish this. This tool created a visual appearance of what the initial pen and paper sketches would be like electronically with minimum detail and no colour. Black and white, simple boxes and text were used to illustrate the wireframes clearly.

As shown in *Figure 10 and Figure 11,* both the portrait and landscape wireframes for the App were created through the use of Wireframe.cc. This not only provided a clearer view of what the App layout would be like, but it also ensured that the screens were not too cluttered and confusing – therefore, user friendly.

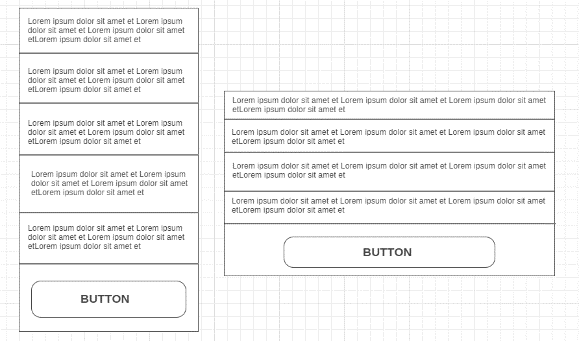


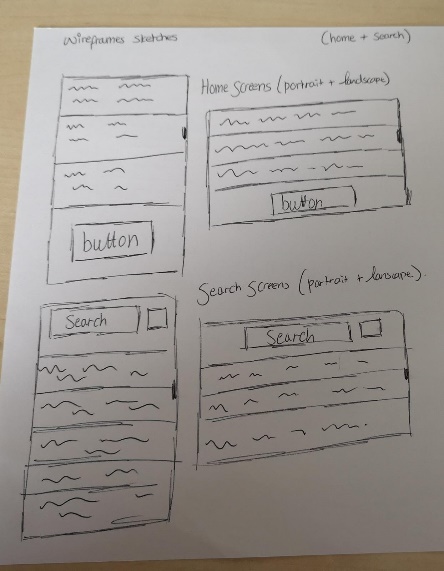
*Figure 10 – Landscape and Portrait Home Screen electronic Wireframes*

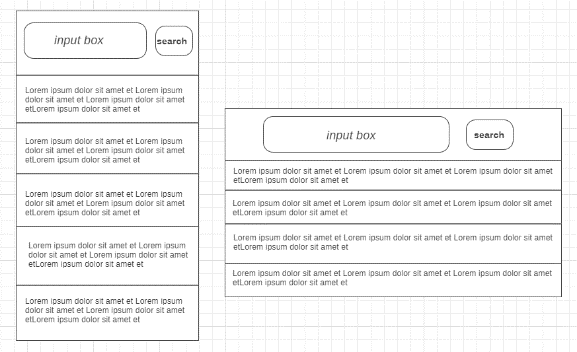


*Figure 11 – Landscape and Portrait Search Screen electronic Wireframes*

*Figure 12* helps to illustrate the initial stage of designing – from pen and paper to wire frames.







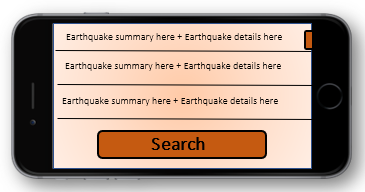
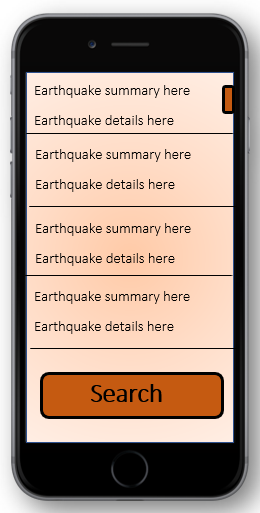
*Figure 12 – Wireframes from pen and paper, to electronic versions*

Screen Visuals

Once the wireframes were established and created electronically, Screen Visuals were then created. This was essentially an “in depth” look at the Wireframe – but in more detail than the electronic and pen and paper wireframes.

Colour were used to provide a “realistic” looking wireframe. This would help aid in the creation of the App, as it would aid in visually recognizing the layout of aspects within the App.

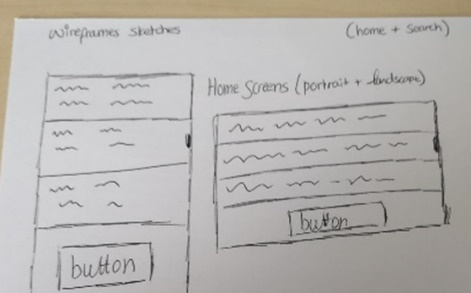
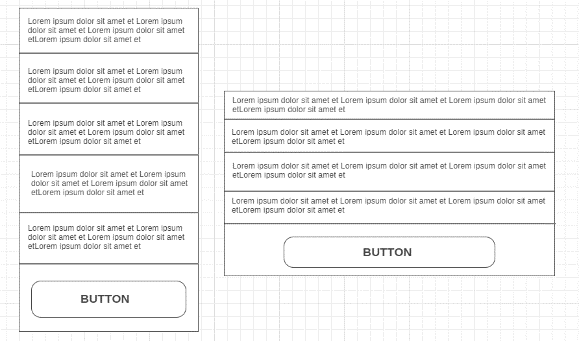
*Figure 13* illustrates some of the Screen Visuals that were created for the App.



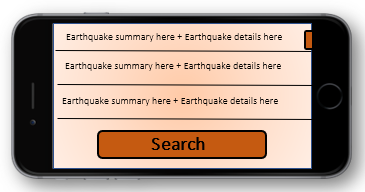
*Figure 13 – Screen Visuals for the Home Page (both portrait and landscape)*

For ease, the basic Screen Visuals were created through the use of Paint and Microsoft Word.

Figure 14 helps to illustrate the Design Stage with Home Screen sample – from start to end: Pen and Paper to Electronic Wireframes to Screen Visuals.







*Figure 14 – Wireframes – from pen to paper, to electronic, to screen visuals*

### Developing Stage

This section will explain the tools used briefly to develop the App itself.

Android Studio

To develop the App, which must run on Android Version 6.0 or above, the software Android Studio was used in order to do so as this was a requirement of the project.

Within Android Studio, the Java language was used in order to code the App – and to adjust and create the layout of the App interface itself, XML was used.

As an approach based on the PullParser class available in Android must be used, the chosen approach for this project is as follows:

* Document Object Model Parsing (DOM)

As Internet access is required for the App to work successfully, the following permission was inserted within the Android Manifest File to allow this:

**<uses-permission android:name="android.permission.INTERNET"/>**

## HCI Principles

In order to design an App that would be successful in terms of User Interface – further research was conducted, and from this research Shneiderman's "Eight Golden Rules of Interface Design" was of importance. These rules were discovered within Ben Schneiderman’s book called Designing the User Interface.

His 8 main rules of Human-Computer Interaction (HCI) are as follows:

1. Strive for Consistency
2. Enable frequent users to use shortcuts
3. Offer informative feedback
4. Design dialog to yield closure
5. Offer simple error handling
6. Permit easy reversal of actions
7. Support internal locus of control
8. Reduce short-term memory load

Upon reading and researching these HCI rules, the following section will provide a brief statement as to how the App will successfully handle each rule.

**Strive for Consistency**

As 2 modes are being created, portrait and landscape, ensure that both modes are similar and consistent with the layout.

**Enable frequent users to use shortcuts**

Ensure there are easy back buttons on screens other than the home page.

**Offer informative feedback**

Show the user a screen when their search result is loading – to inform them that it is loading, rather than leaving it as a blank screen.

**Design dialog to yield closure**

Actions will be thought through carefully – to ensure there is always a beginning, middle and end – rather than a beginning and end.

**Offer simple error handling**

If an error is detected, return to the previous screen.

**Permit easy reversal of actions**

Ensure there is a back button on every screen to allow users to return to their previous action.

**Support internal focus of control**

Enable a search function to allow users to feel more personal and “in control” to the app.

**Reduce short-term memory load**

Ensure all display screens are as simple as possible – if there is a lot of information, arrange it in a visually pleasing way to ensure users are not distracted or confused.

# Testing

The following table (*Figure 15*) will illustrate how each function should be tested, as well as what the expected output is.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Number** | **Test Case Name** | **Input** | **Test Data** | **Expected Output** |
| 1 | Load the Application | Open the Application | Application Runs | The Application successfully loads up |
| 2 | Load Earthquake details summary | Open the Application | List of all earthquake summaries | All earthquake summaries are displayed |
| 3 | Click search | Click on the search button on the bottom | Search page is loaded + all incidents yet to be filtered | Search page is loaded up with the appropriate input search box at the top |
| 4 | Search correctly | Input text in the input text search box at the top that will show a result | Search Thu, 11 Apr 2019 | Earthquake data for the selected input data is displayed for the user |
| 5 | Search incorrectly | Input text in the input text search box at the top that will not show a result | Search + | Error message – asking users to try again |

*Figure 15 – Testing*

## Importance of Testing

This section will essentially describe and explain various Testing techniques as well as their importance within Testing itself. The advantages and disadvantages of the testing techniques explained will also be discussed within this section.

### Why is Testing important overall?

Testing is crucial to any development – including the development of an App, such as this project – for a number of reasons. Testing essentially highlights problems, flaws and issues within the App itself – which will result in the need for improvement in a later on stage. By testing the App before releasing it to clients, it ensures that the App is not flawed – and users will not be disappointed and frustrated with the App itself. Therefore, through testing, the identification of issues can be identified so that they can be fixed before officially being “released”.

### Blackbox Testing

Blackbox Testing essentially tests the App without needing to look at Internal design as well as Code. The tests are based on the functionality and requirements of the App itself.

Methods of Blackbox Testing include, but are not limited, to the following:

* System Testing
* Acceptance Testing

Some of the Advantages of Blackbox Testing are as follows:

* To test the App, you do not need knowledge or understanding of the programming language itself – therefore, you don’t need to be an expert.
* Testing and Testing Results are done from a Users way of looking at the App.
* Testing results can be instant as they can be completed and obtained as soon as the App specification is complete.

Despite the advantages Blackbox Testing can offer, there are also some disadvantages which are as follows:

* The App cannot be fully tested as other program paths are unlooked at
* Tests may have already been conducted by the programmer themselves – therefore, repeating tests are expected
* If there are no clear functional requirements for the App, test cases will be difficult to design.

### White Box Testing

White Box testing essentially tests the Internal structure, design and implementation of the App. The tests usually consist of code that runs in order to test the program itself.

Some of the advantages of White Box Testing method are:

* The testing that is conducted is thorough – therefore, most paths are covered
* Such Testing can be completed in the earlier stages of the project.

Some disadvantages of the White Box Testing method are:

* Tests can be very complex – therefore, high-skilled testers are required to have a great knowledge of programming and its skills.
* If there are many implantation changes, the test script will need changed frequently – and this can cause frustration and repetition.

### Usability Testing

Usability Testing essentially gains testing results by asking users to complete specific tasks within the App itself. This type of testing helps to see how easy it is to use the App and can help identify issues that conflict with the User Interface.

Typically when Usability Testing is conducted, there is an observer present to ensure notes are taken about how the user is interacting with the app – and if there is any errors or difficulties the user has come across within the App.

Usability Testing can provide the following:

1. If the user has successfully, or unsuccessfully, completed the task provided to them
2. The amount of time it took the user to complete the task successfully
3. If the user is satisfied with the task completion, and app as a whole
4. User comments – including the joys and frustrations of the App, as well as any suggestions in regards to changes or improvements that could be made to the App

Some of the advantages of Usability Testing are as follows:

* Direct feedback from potential users of the app allows you to gain suggestions and improvements that could be made to the App
* Flaws and issues are identified easily with the App interface
* Illustrates if the App can successfully do what it was intended to do by completing specific tasks that were taken from the Functional Requirements

On the other hand, Usability Testing also has its disadvantages as follows:

* Testing is not 100% accurate as the testing environment is usually fixed, and the user is focused on the app fully rather than in a normal, real-life scenario
* Testing results gained are mainly qualitive