

**Mobile Platform Development**

**Design**

**Design Report- 1735 words**

**Name: Scott Thompson**

**Programme: Computing**

**Matriculation Number: S1507806**

# Links to access on GitHub

Repo: <https://github.com/Scott-Was-Taken/recycler>

APK: Unfortunately the project would not generate a signed apk and I was faced with a continuous error - 

After much troubleshooting I have not found a resolution to this issue, as the project runs fine on both my own mobile device and emulator (this is demonstrated in the submitted video), I am unsure how to further troubleshoot this. For this reason with the deadline in mind, I hope that the submitted GitHub repo will suffice, please do not hesitate to contact me directly regarding this. (sthomp202@caledonian.ac.uk)

Video: <https://github.com/Scott-Was-Taken/MPD-video>

**“Except where explicitly stated, all work in this report, is my own original work and has not been submitted elsewhere in fulfilment of the requirement of this or any other award”**

**Signed by Student: Scott Thompson Date: 11/04/19**

# Introduction

The following documentation outlines the design process carried out with the aim of developing a mobile platform solution that allows the user to access earthquake data retrieved from <http://quakes.bgs.ac.uk/feeds/MhSeismology.xml>. This data is retrieved in XML format however the solution aims to use this data to deliver a rich dynamic data driven user experience. This should provide the user with an understanding of the earthquake data and all information that it provides in an easily consumable format(s).

When a user accesses the application on their mobile device, they should be able to easily identify any earthquakes by:

* Date or Time of the earthquake.
* Where the earthquake occurred.
* The Depth below the ground that the earthquake occurred.
* The Magnitude of the earthquake

Additionally, the application should allow the user a visual method to identify patterns using:

* Location
* Magnitude
* Depth
* Date

This must be achieved independently from the British Geological website and must only make use of the XML data provided above.

The approach to carry this out involves use of an XMLPullParser to parse the data string from the site into a format that can then be manipulated and displayed onto a Java mobile application. It is important to clarify that the core goal of the application is to achieve usability and human readability. This goal is set with the awareness that

* XML data link fails to do this from a readability perspective.
* British geological site fails on usability on a mobile platform.

The application therefore offers the user a method of accessing British Geological data on a mobile platform in a format that is both usable and readable on a small form factor touch screen device.

# Discussion of Components Portrait/Landscape

The following section identifies the components used for each section and why they are selected. This includes the HCI principles considered.

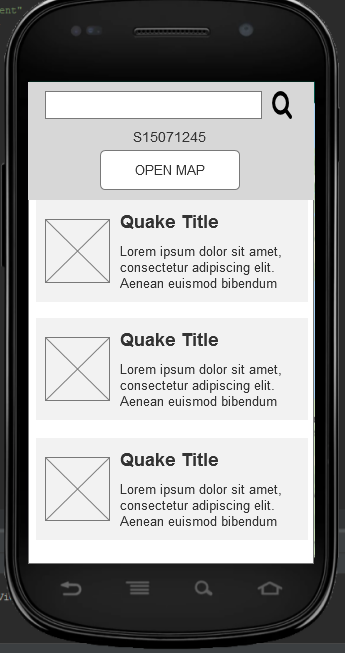
## MainActivity

The main activity page is formed of a scrollable linear layout which allows the user to scroll throughout the entire contents of the list that fills the page. This approach is chosen as gives clear structure to the activity and adapts accordingly to landscape orientation. This is as opposed to absolute positioning which is non responsive and would look completely different on different screens.

The main activity is composed of the following elements:

* Menu
* Open Map button
* Recyclerview of adapter items.

The linear list design contains a menu at the top of the screen where searches can be performed. This approach is justified as such an approach then allows the recyclerview to be adapted dynamically responding to user input. Since the alternative to this is a static list of objects pulled from the site the menu provides the user a simple method of filtering the list.



1 Wireframe for MainActivity, portrait

## MapActivity

The map activity page is dominated by the google maps API fragment tag with the rest of the interface built on the top of this map. The map itself fills the entirety of the page and automatically facilitates portrait and landscape orientation. Upon loading the MapActivity page, the maps camera is set a such a zoom that it is clear to the user in both portrait and landscape what is being displayed. (i.e if it is too zoomed in/out to be recognizable when in landscape mode then this is no use). Additionally, marker placement is a consideration when determining zoom since it is important to zoom out far enough to show the user the extent of the data available without zooming out so far that all markers become an unrecognizable cluster. It is important as far as usability is concerned to evaluate this when considering map zoom.

On top of this map there is a key which can be found at an additional xml resource and positioned over the map. map\_key.mxl. This approach means that there is no need to design an additional layout to facilitate positioning of elements when in landscape as the XML layout can simply be overlayed from an external resource and function correctly in both orientations

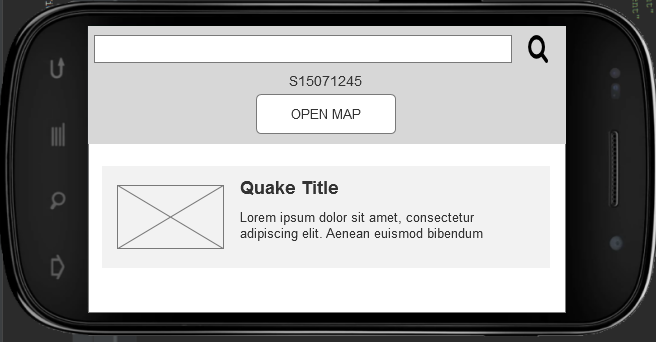


2Wireframe for MapActivity, portrait

## MainActivity (landscape)

The approach taken for MainActivity Design allows the page to be filled completely in landscape orientation meaning that no space is wasted as the recyclerview automatically spreads its item contents across the page. This is beneficial as it decreases scrolling necessary for the user to navigate the list. An alternative implementation that implements a similar list style system may not take advantage of this increased horizontal space and would therefore offer no benefit to the user for using their device in this orientation.

This consideration means that the entire screen is used therefore increasing the usability on the mobile platform. This is not a feature that is present on the current system used by the British Geological Society as the pages are not designed with portrait and landscape in mind.



3Wireframe for MainActivity, landscape

## MapActivity(landscape)

As mentioned above the decision to overlay an external xml file on the google map means that the map is still entirely usable on smaller screens and this translates well onto landscape orientation. In practice, the key continues to sit at the top of the screen while the map continues to max out the screen underneath. As the key is small in size, the removal of the back button in landscape mode allows the user to maintain a full view of the map and operate it unobstructed. The absence of the back button is not a significant loss as explained in the user interaction section of the document it is not required since moving to the main activity means that MapActivity is “finished” i.e the user can use their android phone’s built in back button functionality here and it will operate as expected.



# How does the user interact with the data?

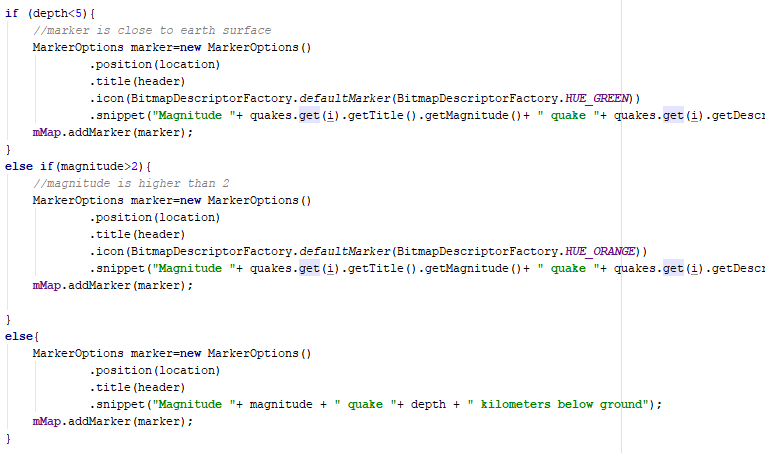
On load, the application loads the URL’s XML content into a result string. The solution submitted takes this string and converts it to an input stream. This approach allows a simple pullparser method to take the data by identifying tags required and taking the data next to the tag to pass into an object of type Item (a class built to contain data about earthquake items). This class contains the following values:

**private** Title **title**; (an item of type title that holds information parsed from the title field)  
**private** Description **description**; (an item of type description that holds the information parsed from the description field)  
**private** String **link**;(the link to the URL for that earthquake)  
**private** String **pubDate**;(the date published)  
**private** String **category**;(the category of the quake)  
**private** String **lat**;(latitude on map of earthquake location)  
**private** String **lon**;(longitude on map of earthquake location)

To fill these values the parser method calls the appropriate parser method as required to pull out only relevant data meaning the data is in an immediately usable format. For example, if Item.Description.depth is 2 KM in the XML file then is parsed to an integer equal to “2” as opposed to remaining a string equal to “2 KM”. This approach to parsing allows the application to act in a data rich way where data can be displayed depending on what information it holds. After parsing of an item object is complete, it is put into an arraylist and this arraylist is where view methods access data from. This approach means that all view methods can be written with the assumption that they will receive an arraylist of item objects to build with.

For the scrollable recyclerview on the Main Activity, the data is then loaded into items of type adapter in the interest of performance before they are displayed to the page. This is important as a large dataset being immediately loaded onto the page would cause significant performance issues or even crashing. Once this has been performed the user can search the list and filter it to display based on the contents of the descriptions for the adapter items. In future iterations the image can be customised based on values such as magnitude or depth as they are stored in variable types that can be manipulated easily.

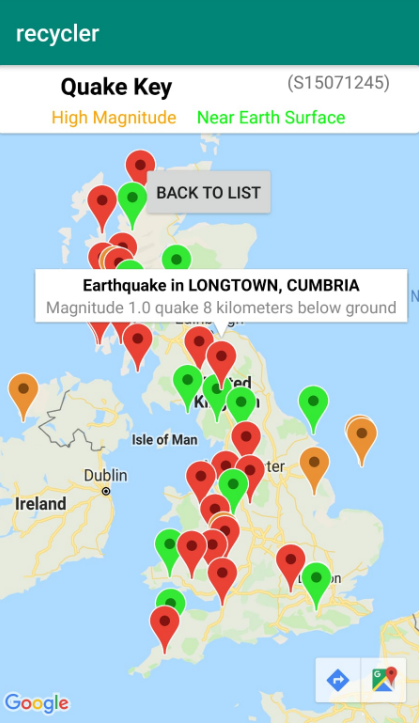
This approach for data manipulation is taken into practice on the maps activity where the string is passed as an extra on the intent. After it is parsed it is used to display markers conditionally depending on calculations performed. The example provided in the submission of this is the following condition:



4Code sample from marker placement method showing data manipulation

This effectively translates to “if depth is less than 5 then it is close to the earth so display it green. If magnitude is greater than 2 then it has high magnitude so make it orange and if neither are the case display a regular marker”. This code can be adapted in future iterations to display all manner of calculation results.

The user can then tap on these markers to see the information for the earthquake they are interested in. The result of the code above is as follows:



5Map Activity in use after user has tapped on a marker

# Evaluation

In hindsight, I would argue that the specimen produced is a suitable response to the specification and is built with future maintainability in mind. The approach taken to parsing facilitates coding of further view methods and activities on the assumption that the data will be usable in the format identified above (Arraylist <Item>). For example, the search method on the main activity could be further refined without the need to refactor the entire code base and this is the same for the map activity. Passing the string between pages means that the application does not have to access the web source after initial load resulting in improved performance. A future implementation would look to implement parcelable to pass the arraylist of items between activities as this would be far more efficient than performing pullparsing on both activities.

As identified in the introduction, a goal of development was “allow the user to identify patterns and understand the broad meaning of the data”. This is achieved on the map data where the user gets a clear overview of earthquake location by depth and magnitude. This means that the user can identify trends and understand the data far more clearly than reading it line by line. Overall, the specimen produced offers functionality to users who wish to access the BGS data in a readable format on a mobile platform and it is therefore argued that the specification is met.



**Mobile Platform Development**

**Testing and Documented Testing**

**Testing Report - 903 words**

**Name: Scott Thompson**

**Programme: Computing**

**Matriculation Number: S1507806**

**“Except where explicitly stated, all work in this report, is my own original work and has not been submitted elsewhere in fulfilment of the requirement of this or any other award”**

**Signed by Student: Scott Thompson Date: 11/04/19**

# ParseXML

Thorough testing of parseXML has been carried out and can be seen through the extensive logging that the methods perform as they run. This ensures that if the format of the XML file remains valid, the pullparser will parse the file into an arraylist of Item objects without issue. Test Driven Development was the approach for most of the parsing code written as bugs were identified by logging all data that passes through the parser and ensuring it matches that which is expected. Logging to this extent was required to debug the parser as it was written.

A test-driven development approach meant that the parser was set test cases to pass throughout its creation starting from “can it pull things out of tags and put them into strings” and progressively improving on a value by value basis parsing unrequired information out of and assigning an appropriate type to each value. By taking this approach the developer has a clear minimum viable pullparser early on and can work on improving its data handling incrementally.

Following this the developer then further improved upon this by writing parsing methods for the title and description tags as they contain a great deal of valuable data such as depth and magnitude. The initial test case that was passed only saw the entire contents of this field emptied into a string value:

String Title= “UK Earthquake alert : M 0.7 :NEWDIGATE,SURREY, Thu, 11 Apr 2019 00:18:34”

While this is arguably effective for minimum viable product the following test cases breakdown involved taking each value visible in the string, taking its contents and removing the unnecessary information so that data can be stored in appropriate types.

An example of this is magnitude, where a successful test pass would be where input data is the string above (title) and output would be a double variable:

double magnitude = 0.7

Once this test case is passed we now have a parser that can reliably determine which value in the title string is magnitude and it can get that information and return it as a value that can be used in calculations throughout the application.

This was the consistent approach taken throughout testing and development of the application’s parser.

# Map marker placement

As with the parser Test Driven Development was used to develop the marker functionality on the maps. Testing on the map marker placement was carried out as the map was developed. Initially this testing was performed on a single hardcoded marker where the marker lat, long and title was hardcoded. This test was passed early on and this allowed the developer to approach developing a solution that made use of the parsed data to generate a map of icons. Initially this was done without colour functionality initially to prove concept for a map filled with icons each representing an earthquake. This was an important test to run to ensure performance of the application was not affected by the early assumption that the number of markers necessary will be handled gracefully by the android platform.



Map in testing phase running without colours

Once this test was performed it was then possible to implement and then test conditions for markers, testing at this stage unearthed several issues with markers being placed multiple times for one quake when several conditions were met by one quake. Examples of these early tests would be quakes that meet the conditions to be classed as both close to earth and high magnitude. Using test driven development at this stage allowed the developer to unearth these issues at the development stage and ensure that any crashes caused by conflicting markers or by infinite loops are resolved. The resolution for these conflicts was to prioritize high magnitude earthquakes since they are significantly rarer, at this point the algorithm then places markers for any remaining markers near the ground and finally places the remaining basic markers.

Testing was then performed for a random selection of markers ensuring that they corresponded with the expected marker colour assignment. I.e. markers were manually selected from the xml file and checked to see if their colour corresponded with the expected colour. This ensures that the colour assigned to a marker is correct (with the exception of markers that should be both colours, at which point the current implementation prioritizes high magnitude)

# Usability testing

Once a completed minimum viable product was produced with functioning parsing and map markers, it was then possible to conduct basic usability testing with an example end user. For ethical reasons no details on the identity of this user were kept. This was conducted by providing the end user with the solution’s apk on a physical device without telling the user any details about the purpose of the application or giving any insight into the expected user journey. The goal of this was to identify if the user would discover any user interface issues or unearth anything that wasn’t clear. This test was a success as the user reported no issues with the recycleview and the search. However, on discussion with the developer, the user identified the fact that there was no indication to them that the colour of the markers on the map was significant. This test resulted in the creation of the map legend/key feature that exists in the submitted prototype. Additionally this test indicates that the solution provided is suitable as a response to the problem specification.



**Mobile Platform Development**

**Testing and Documented Testing**

**Documented Testing**

**Name: Scott Thompson**

**Programme: Computing**

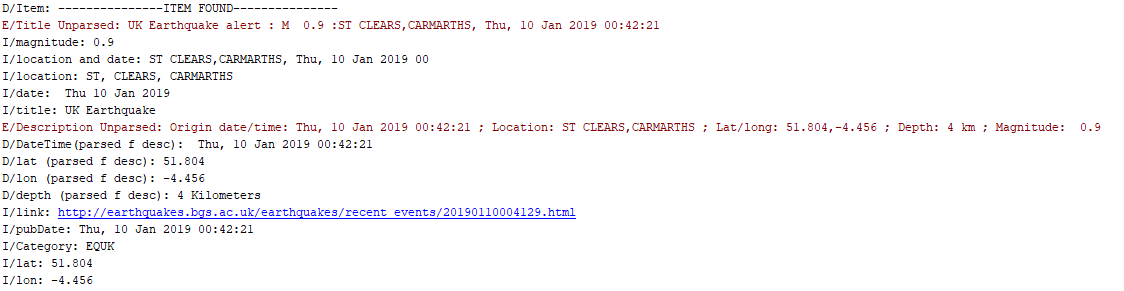
**Matriculation Number: S1507806**

**“Except where explicitly stated, all work in this report, is my own original work and has not been submitted elsewhere in fulfilment of the requirement of this or any other award”**

**Signed by Student: Scott Thompson Date: 11/04/19**

# ParseXML Testing

As development of these methods was test driven the attached console logs show the application receiving a full pass for expected output with a correctly formatted XML dataset.



Test Pass for ParseXML - Clears, Camarths earthquake and all its values parsed as expected

# Map Markers Testing

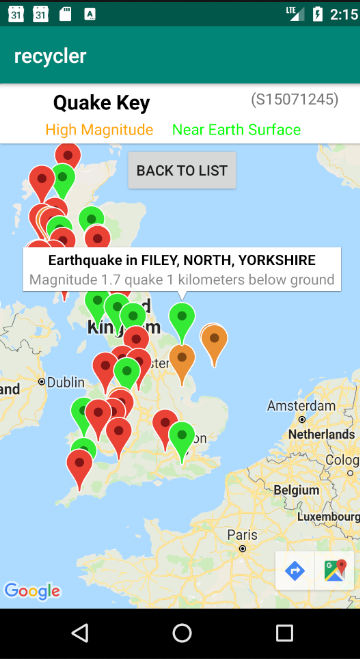
Map marker testing can be seen below:

**An orange marker in Donegal Ireland is expected since the magnitude exceeds 2.0 at 2.3**



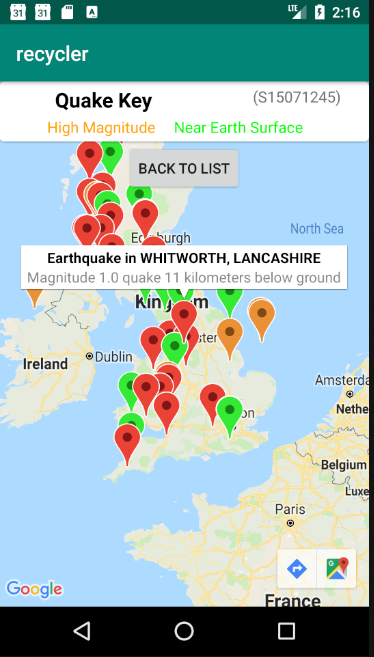
**8** Test Pass for high magnitude - Donegal Ireland displaying as orange

**A green marker in Filey, North Yorkshire is expected since the depth is only 1KM below ground**



**9**Test Pass for low depth - Filey Yorkshire displaying as green

**A red marker in Whitworth Lancashire is expected since the magnitude is below 2 at 1.0 and the depth is 11 (over 5km)**



Test Pass for no depth/mag - Whitworth Lancashire as red

# Usability Testing

At the usability testing stage, as discussed above the user was presented with the solution and prompted with only the following show me/tell me questions:

* What is the application for? - Expected response: Allows me to view BGS data about earthquakes
* Show me how to find all earthquakes with magnitude 0.5 – Expected response: user uses search bar above recycler and types 0.5
* Show me how to access the earthquake map – Expected response: user navigates to mapactivity using button on mainactivity
* Tell me what the marker colour means – Expected response: marker colour indicates low depth or high magnitude.
* Show me how to get back to the list – Expected response: user presses android back button or uses on screen back button
* Show me how to find earthquakes in Ireland? – Expected response: user uses map and moves to Ireland or uses search field.

The results of this testing are as follows.

* What is the app for? – Pass: user understands what app is displaying
* Show me how to find earthquakes with magnitude 0.5 – Pass: user fully understood search and used searchbox as expected.
* Show me how to access the map – Pass: user fully understood how to move between activities
* Tell me what marker colour means – Fail: user was not aware marker colour was relevant.
* Show me how to get back to the list – Pass: as above
* Show me how to find earthquakes in Ireland: Pass: user opted to use map for this.

In response to the failed user test “what does marker colour mean” the design was updated to include a map key/legend. This test was followed up with the user who agreed this resolved their confusion.

# Bibliography

The submission credits the following tutorial sources for aid in learning and implementing the android solution submitted:

* Youtube – CodingWithMitch tutorials on google maps <https://www.youtube.com/playlist?list=PLgCYzUzKIBE-vInwQhGSdnbyJ62nixHCt>
* Youtube – Coding in flow tutorials on recyclerview and search <https://www.youtube.com/watch?v=Nw9JF55LDzE&list=PLrnPJCHvNZuBtTYUuc5Pyo4V7xZ2HNtf4>
* Youtube – Sylvan Saurel XML pull parser tutorial <https://www.youtube.com/watch?v=-deKKeEdpbw>