Mobile Platform Development (MHI322959-18-B)

Name: Pierce Hepburn

Programme: BSc (Hons) Computing

Student Number: S1711876

Contents

[1 Links 3](#_Toc5918921)

[1.1 Demo Video 3](#_Toc5918922)

[1.2 Android Project 3](#_Toc5918923)

[1.3 Project APK 3](#_Toc5918924)

[2 Design Report 3](#_Toc5918925)

[2.1 Introduction 3](#_Toc5918926)

[2.2 What The Design Should Achieve 3](#_Toc5918927)

[2.3 Screens & Components Used 3](#_Toc5918928)

[2.3.1 Main Activity 3](#_Toc5918929)

[2.3.2 Earthquake Detail Activity 5](#_Toc5918930)

[2.3.3 Other Design Decisions 5](#_Toc5918931)

[2.4 User Application Interaction 5](#_Toc5918932)

[2.5 Conclusion 5](#_Toc5918933)

[3. Testing Report 7](#_Toc5918934)

[3.1 Unit Testing 7](#_Toc5918935)

[3.2 UI Testing 8](#_Toc5918936)

[3.3 Integration Testing 8](#_Toc5918937)

[4 Testing Documentation 9](#_Toc5918938)

[4.1 Unit Testing 9](#_Toc5918939)

[4.2 UI Testing 12](#_Toc5918940)

[4.3 Integration Testing 15](#_Toc5918941)

# 1 Requested Links

## 1.1 Demo Video

<https://github.com/pimihe/MPD/blob/master/video.mp4>

## 1.2 Android Project

<https://github.com/pimihe/MPD/tree/master/Earthquakes>

## 1.3 Project APK

<https://github.com/pimihe/MPD/blob/master/app.apk>

# 2 Design Report

## 2.1 Introduction

I have been tasked with the creation of an earth quake history android app which is supposed to give a history of UK earthquakes over the last one hundred days. There are few hard-set requirements for this project with the only features specified as required being filtering the displayed data and the requirement to display the earthquake data in a meaningful way. This gives a lot of freedom to the approach taken in the design of the project.

## 2.2 What the Design Should Achieve

The focus of the application should be on the display of the earth quake data and displaying this in a meaningful way and creating a user-friendly way to filter and sort the data. Since android can have several different possible screen sizes and potential of orientations I tried to design components and screens which will have a similar look and feel regardless of the orientation or screen size to try and achieve a familiar look regardless of these factors.

When displaying earthquake information, I re-used the same component whenever that information was on screen making slight changes to accommodate different orientations. By doing this I believe I have created a design that allows users to quickly pick up on the information of each earthquake due to consistency in how the earthquakes are displayed in the application.

## 2.3 Screens & Components Used

I decided it would be best to split the app into two activities with multiple fragments making up each activity. The first activity used is a main activity showing a summary of the earthquake in different ways through different fragments. The second activity is an earth quake detail activity which gives as much detail as is available on the selected earthquake.

### 2.3.1 Main Activity

The main activity is made up of a constant toolbar at the top of the screen which features the application title and an icon which when pressed will open a filter menu. The toolbar will also have a menu icon which when pressed will allow the user to select one of three different fragments which will be shown beneath the toolbar. The use of a toolbar at the top of the screen should make it clear to a user what actions are available to them upon opening the app as it contains all the functionality required to filter earth quakes and switch between fragments effectively giving users shortcuts to each different screen of the application. Having a toolbar at the top of the screen also maximises the available screen space when compared to putting tabs on the screen along with the toolbar to switch between fragments.

Using the menu icon on the toolbar the user can switch to different fragments and the first of these is a register screen which will give a summary of the magnitude, depth, time and location of the earthquakes. This list of earth quakes makes use of a component where both the magnitude and depth will have their own display icon which will be colour coded. The colour coding of the depth and magnitude will go from green to red depending on danger for each of these values to allow users to quickly see how dangerous and earthquake is depending on the depth and magnitude. The colour coding of depth and magnitude will go from green to red with red representing more danger and so earthquakes which occur lower in the earth will be greener and higher magnitude earthquakes will be redder. The Register fragment will be made up of a list of these components and depending on screen more of these items will be displayed on a single row of the screen using breakpoints to make the best possible use of the screen space available while avoiding overlapping the list items.

The second fragment available through menu icon would be the map fragment which is made up of a google map with map pointer for each earthquake. This fragment is used as another way of allowing users to visualise the data and could be used to potentially see clusters of earthquakes. The map pointers will also be colour coded from red to green depending on the current filter sort allowing users to see a summary of the current sort based on location. This colour coding could be useful when sorting by magnitude to see clusters of high or low magnitude earthquakes or when sorting by depth to see clusters of high or low depth earthquakes. This colour coding could also allow users to see the distance from their location or the distance from a place they have selected as the markers would slowly change colour depending on their distance from a point creating a gradient effect on the map. The pointers loaded on the map load in slowly and so a progress bar is given to give the user some feedback during this process These pointers are also be clickable to open the earthquake detail activity for that specific earthquake.

The third fragment available through the menu icon is the graph fragment which is made up of three graphs created using an android graphing library from http://www.android-graphview.org/. The three graphs will be made up of a magnitude against the current filter sort line graph, another graph with the depth against the current sort line graph and a final graph showing the magnitude of all earthquakes against the depth to create a scatter diagram. The use of the first two-line graphs could be to allow a user to sort by date and notice a trend during a certain time or sort by magnitude or depth and see how depth or magnitude move with each other. The scatter diagram can be used to allow users to see clusters of earthquakes which often occur with similar depth and magnitude during the last hundred days. Each earthquake is a point on these graphs which will be clickable to open the earthquake detail activity for that specific earthquake.

Alongside the menu icon on the toolbar is the search icon and this can be used to open a filter dialog type fragment over the top of the currently active fragment. When filters are applied through this menu the user receives success or failure feedback through android toast but since the user can always see the earth quake data behind the filter dialog they get to see the filter applied giving instant feedback. This fragment is made up of three sections for filtering the depth, magnitude and date of earthquakes with each section having a checkbox to allow disabling sorting with this field, they will also have dropdown spinners to allow setting equal to, greater than, less than and between two value options. The magnitude and depth sections of the filter will contain number text inputs and the date filter will feature a disabled text input which will open androids date picker asking the user to input the desired date. When using between option for the filter the fragment will change slightly in that the section will have two inputs instead of one to represent the min and max values. The last section of this fragment will be made up of a sentence with embedded dropdown spinners to allow the user to sort by the date, most northerly, most southerly, magnitude, depth, distance from location and distance from the user. For the final section this sentence will remain the same apart from the distance from location option which will display a text input in the middle of the sentence for the input of the location. This final section is ended with an apply button which will apply the selected filters and sorts and a clear button which will reset the entire filter fragment.

### 2.3.2 Earthquake Detail Activity

In the main activity of the app the stats of earthquakes and the exact location can be seen on different fragments, so the earthquake detail activity is supposed to combine these for a specific earthquake to give the user more detail on a single earthquake. This activity will be created using a different toolbar which will feature a back button in place of the filter and menu icons of the main activity’s toolbar to allow for returning to the main activity. Below the toolbar will be made up of a map section and a stats summary section. The map section will be identical to the map fragment from the main activity but will only feature a single location marker which will be the earthquake that was selected. The summary section will feature the same fragment used for the list item in the register fragment of the main activity but modified to make use to the increased screen space due to only displaying data for a single earthquake. This earthquake detail activity will also change how it displays the map and earthquake summary sections depending on the screen orientation with portrait displaying the map above the summary stats and the summary stats having the location moved to the right to make use of the screen space. The landscape version of this earthquake detail summary will feature the same two sections, but the map section will be moved to the right and the summary stats section will be on the left of the screen and will be identical to the way it is when featured in the register fragment of the main activity due to no longer having the same horizontal space. Whether the device is portrait or landscape the same fragments will be used for the map as it is created from a library and adjusts itself and the stats summary section can be slightly modified to make the best use of the screen space.

### 2.3.3 Other Design Decisions

Along with displaying three fragments the main activity is also in charge of displaying feedback data according to the status of each API request. The application is built to request new earthquake data every fifteen minutes and so if these requests succeed or fail the main activity will display the outcome of this request to the user with a updated message if data is updated a failed to get data message, if the data could not be retrieved or if the data was retrieved but no changes were found from the current data no toast message will be displayed. On start up the app is designed to give an acknowledgement to the earthquake data provider via a toast message although this is not directly built into the main activity but an application start up class. On start up the app will initially ask for permission in order to prevent the user’s usage of the application being interrupted later on.

## 2.4 User Application Interaction

The user’s interaction with the app starts with the main activity and the three different fragments described in the previous section. Using on of these three sections the user can select an earthquake to see the full information from based on some of the information given by one of the three fragments or by filtering to pre-select a type of earth quake to look for. Once an earthquake has been selected the user can view the location and summary data of the earthquake at the same time which they cannot do on the main activity fragments.

## 2.5 Conclusion

Overall, I am pleased with how the overall design of the application turned out and this is due to a few factors I feel were successful in the application. The use of a similar toolbar across both screens makes it obvious to users what functionality is available to them on the current screen. The use of breakpoints in the register allowed me to make the screen the user will most likely spend the most time on work well in multiple screen sizes. Reusing the same component to summarise the earthquake data and including a colour aspect to it allowed me to quickly give users information on each earthquake.

# 3. Testing Report

When testing the application a few different strategies were put in place to test the components of the app. To efficiently test the application, I decided to split the testing into UI testing to test responsiveness across different screen sizes, unit testing was appropriate to test specific functions and integration testing to ensure the app was working correctly as a complete product. Throughout this testing a white box type approach was used to target expected problem areas as I thought this would be the most efficient way to find problems.

## 3.1 Unit Testing

Unit testing allowed me to find issues with specific parts of the app while they were separated out from the rest of the system. This type of testing allowed me to identify issues with these smaller parts of the system before they were integrated with the rest of the system and caused a chain of errors which would have been more difficult to fix. The Unit tests were carried out on the functions contained within the UtilHelper, NetworkHelper and XmlHelper classes. It made sense to perform this type of testing here as the functions within these classes can work without any other parts of the system and all can be run without using the android emulator with the exception of the network status check function.

The unit testing here resulted in large changes taking place within the XmlHelper class was using a different way of extracting information from the description field of the earth quake xml resulting in depths not always being saved correctly. Once finding issues here this was changed to better extract the information found within the tag.

The NetworkHelper was another area aided by testing as issues were found when failed requests occurred which would crash the application. The issues with the network helper class were fixed with the use of try catches to prevent crashes as even if a network connection is found before the request starts it cannot be guaranteed the request will succeed. This improved the app as it no longer crashes upon a failed request but returns null allowing failures to be identified by the app and feedback given to the user based on this.

The UtilHelper class contains a pair of functions used to format the earthquake summary display component used in the register fragment and the earth quake detail activity. These two functions have a large effect on how the earth quake data is displayed to users and so it is important they were tested properly. The location formatting function was difficult to test as there is no standard format for the location string for earthquakes suggesting this location string could change in future to a format that has not been though of making the function break. Unit testing this function involved gathering a list of the location strings at the time of testing and making sure no unexpected feedback were found. As expected some issues were found when testing this and were fixed as they were found during testing with the hope these would prevent any future issues.

The other function within the UtilHelper class was responsible for the colour gradient applied to the magnitude and depth parts of the earth quake component used in multiple places in the app. Testing this function took a long time as its purpose was to provide an RGB hex colour gradient string based on a range of numbers with the option to start from red and go to green or the other way around. Testing revealed the issues with reversing the order the colour transitioned as initially this functionality was forgotten about resulting in the depth colour being redder the deeper it was as opposed to greener for deeper and therefore less dangerous earthquakes. Issues were also found with negative magnitude values which required large modifications to the function to fix. Multiple iterations of this function resulted in many different results with the result being a function which appears to work across all ranges of magnitudes and depths and can reverse the colour gradient transition to go from red to green and vice versa.

## 3.2 UI Testing

The UI of the application employs different methods to display correctly across different devices including a method using breakpoints and a method were an existing element is modified depending on screen orientation. To make sure these two parts of the application are tested properly an android emulator which offers the ability to flip the screen to multiple orientations and set the screen width to custom values was used. When testing the UI each fragment within the main activity along with the filter dialog and the earth quake detail activity were tested along with the specific parts mentioned earlier using the same methods of an android emulator with custom screen widths and orientations and checking for unexpected behaviour. Doing this allowed me to find issues when reusing the earthquake information component from the register list in the earthquake detail activity. This method also allowed me to fix issues with the breakpoints in the register fragment from the main activity. I was also able to identify issues with the filter fragment when on certain screen types as the apply and clear buttons would overlap with other items.

## 3.3 Integration Testing

The integration testing was carried out at the end of the development process and was mostly made up of quickly testing areas that were suspected of having issues since being the developer gave insight to potential problem areas.

The first issue was found when testing the clickable pointers on the map fragment of the main activity. The map pointers used an id given to pointers on creation which was then used as an index to open the detail activity for the clicked earthquake however if the map was reloaded the index would not reset to zero resulting in an out of bounds error crashing the map. This issue made the map screen almost useless when filters or sorts were needed as trying to open an earthquake by clicking the filter would break the app. Identifying this through testing greatly improved the quality of the app as it resulted in the map screen being useable with filters and sorts.

When creating the graph fragment, I decided it was important to implement a way to click the points to open earth quake details activity with these points however the library used for this did not provide a way to do this. To implement this functionality a hack was used where the index of the earthquake within the list was added to the points on the graph through adding a decimal point to the depth of the earthquakes followed by zeros and the index of the earthquake. I expected this would be broken by rounding errors and was correct as some indexes would create rounding errors when added to numbers. This was fixed and tested further mainly by lessening the number of zeros following depth value which seemed to fix the issue.

The filtering of earthquakes is done through the dynamic creation of SQLite statements was suspected to be a large source of errors with the amount of possible SQLite statements that could be created. The dynamic creation of statements did not turn out to cause any issues however the geolocation code and GPS location code used could sometimes fail resulting in filters not including not working as expected. This issue was found to be caused by a context object being passed to the view model for the filter fragment causing issues and so was fixed by creating a new class which gets an application context on start up preventing the need for the filter fragment to pass its own context and fixing the issue

# 4 Testing Documentation

I logged the tests that were carried out in tables with tests which did not go as expected featuring extra comments and what was done to fix issues.

## 4.1 Unit Testing

Unit tests were logged in the table below with commentary given when failures occurred and the code for these can be found within the test folder of the android project if they were needed. For functionailty involving android only libraries running unit tests was not possible as the code can only be run on the android platform so they were tested on a running emulator. Also, the getHexColorFromNoRange function testing involved subjectively judging a colour generated by the function so this involved generating a class and changing the values given to the function to get a result from each input.

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Expected Result | Actual Result | comments |
| getUrl function tested with xml url | Xml will be returned | As expected |  |
| getUrl function tested with url which will fail | Null will be returned | Error occurred and stopped execution | Try catch surrounded to catch this error and return null |
| getUrl function tested with url which will give 404 | Null will be returned | As expected |  |
| getUrl function tested with url which will give empty | Empty string will be returned | As expected |  |
| Android emulator used and app started with no wifi | User receives no connection feedback | As expected |  |
| Android emulator used and app started with wifi | User receives new data | As expected |  |
| parseXml run with xml data | No errors will occur and xml will parse correctly and return earthquake list | As expected |  |
| parseXml run with badly formed xml | Null returned | Code stopped with error similarly to original network issue | Try catch surround added to code to handle this situation and return null |
| parseXml run with one earthquake of xml | No errors will occur and xml will parse correctly and return earthquake list of length one | As expected |  |
| parseXml run with ten earthquakes of xml | No errors will occur and xml will parse correctly and return earthquake list of length one | As expected |  |
| parseXml run with earthquake with 0 magnitude | No errors will occur and xml will parse correctly and return earthquake list of length one | As expected |  |
| parseXml run with earthquake with 4 magnitude | No errors will occur and xml will parse correctly and return earthquake list of length one | As expected |  |
| parseXml run with earthquake with -1 magnitude | No errors will occur and xml will parse correctly and return earthquake list of length one | As expected |  |
| parseXml run with empty string | Null returned | As expected |  |
| getFormattedLocationString run with location without sub location | String array of length 1 that is correctly formatted returned | Length of two returned | String in position two was empty so would not cause display issues anyways but modified code to prevent this |
| getFormattedLocationString run with location with sub location | String array of length 2 that is correctly formatted returned | As expected |  |
| getFormattedLocationString run with location that is empty string | String array of length 0 that is correctly formatted returned | One length list with empty string inside is returned | This should make no difference to UI since string is blank and would take more work than is worth to fix so this shall remain the same unless it is found to cause issues with UI on device |
| getFormattedLocationString run with location with multiple sub locations | String array of length 3 that is correctly formatted returned | As expected |  |
| getHexColorFromNoRange run with value of 50, min of 0, max of 100 and greenFirst set to true | Colour function returns yellow colour hex code | As expected |  |
| getHexColorFromNoRange run with value of 0, min of 0, max of 100 and greenFirst set to true | Colour function returns green colour hex code | As expected |  |
| getHexColorFromNoRange run with value of 100, min of 0, max of 100 and greenFirst set to true | Colour function returns red colour hex code | As expected |  |
| getHexColorFromNoRange run with value of -50, min of -50, max of 50 and greenFirst set to true | Colour function returns green colour hex code | getHexColorFromNoRange returned invalid hex code by giving string with 10 characters | Testing the getHexColorFromNoRange revealed it has large issues when dealing with ranges which do not start from 0 and issues with ranges that dip into negative numbers. The function was heavily modified to deal with these types of inputs, previous tests were redone and found to be successful and testing continued after verifying modifications had fixed the issue. |
| getFormattedLocationString run with location with multiple sub locations |  | As expected |  |
| getHexColorFromNoRange run with value of 50, min of 0, max of 100 and greenFirst set to true |  | As expected |  |
| getHexColorFromNoRange run with value of 0, min of 0, max of 100 and greenFirst set to true |  | As expected |  |
| getHexColorFromNoRange run with value of 100, min of 0, max of 100 and greenFirst set to true |  | As expected |  |

## 4.2 UI Testing

UI testing mostly involved testing issues I suspected would occur involving UI components within the app

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Expected Result | Actual Result | comments |
| Main activity, with register fragment active tested with screen size of 650px | Register will display 1 item to a row | As expected |  |
| Main activity, with register fragment active tested with screen size of 750px | Register will display 2 items to a row | As expected |  |
| Main activity, with register fragment active tested with screen size of 850px | Register will display 2 items to a row | As expected |  |
| Main activity, with register fragment active tested with screen size of 1150px | Register will display 3 items to a row | As expected |  |
| Main activity, with register fragment active tested with screen size of 1250px | Register will display 4 items to a row | Register shows 1 item per row | This was due to the grid layout always showing one if screen width was not in range of 700-1200 and to fix this code was added to consider screens greater than 1200px |
| Main activity, with register fragment active tested with filter active to 1 result | Register will display a single item on the left side of the screen | As expected |  |
| Main activity, with map fragment active tested in landscape mode | Toolbar displays at top of screen with map below and markers oriented correctly | As expected |  |
| Main activity, with map fragment active tested in portrait mode | Toolbar displays at top of screen with map below and markers oriented correctly | As expected |  |
| Main activity, with map fragment active tested with emulator locations et to point in UK and sorting by distance from me in descending order | Points close to location will be green with further points getting slowly redder | Close points appearing slightly bluer than green and further points appearing more yellow than would be expected | Colour is dependant on the offset of a hue wheel, numbers in the code to deciding the offset were updated to ensure blue was not reached |
| Main activity, with graph fragment active tested in landscape mode | Three graphs will appear one above the other | As expected |  |
| Main activity, with graph fragment active tested in portrait mode | Three graphs will appear one above the other | As expected |  |
| Main activity, with register fragment active tested with filter fragment opened over top | The filter fragment will appear on top of the active fragment and should not cause any changes to below fragment by being open | As expected |  |
| Main activity, with map fragment active tested with filter fragment opened over top | The filter fragment will appear on top of the active fragment and should not cause any changes to below fragment by being open | As expected |  |
| Main activity, with graph fragment active tested with filter fragment opened over top | The filter fragment will appear on top of the active fragment and should not cause any changes to below fragment by being open | As expected |  |
| Main activity, with filter fragment opened over top active fragment with device in portrait mode | Each section of the filter fragment should be visible after the previous in vertical layout | As expected |  |
| Main activity, with filter fragment opened over top active fragment with device in landscape mode | Each section of the filter fragment should be visible after the previous in vertical layout | Apply and clear buttons overlap over the top of sorting section | This was an issue with where the button linear layout had been placed and was fixed by placing buttons within same linear layout as the sorting section |
| Detail activity opened in portrait mode | Toolbar will appear at top of screen with map below and earthquake summary component displayed at bottom | As expected |  |
| Detail activity opened in landscape mode | Toolbar will appear at top of screen and map and earthquake summary displayed below with the map on the right side | Modifications to the earthquake summary display when in portrait mode stay applied in landscape causing the text to take up more space than desired | Code added to modify the text location when landscape mode active |
| Main activity, with register fragment active tested with a single earthquake with less than 0 magnitude value | Magnitude colour indicator of earthquake item will be green | As expected |  |
| Main activity, with register fragment active tested with a single earthquake with 0 magnitude value | Magnitude colour indicator of earthquake item will be green | As expected |  |
| Main activity, with register fragment active tested with a single earthquake with 6 magnitude value | Magnitude colour indicator of earthquake item will be red | As expected |  |
| Detail activity, with register fragment active tested with a single earthquake with less than 0 magnitude value | Magnitude colour indicator of earthquake item will be green | As expected |  |
| Detail activity, with register fragment active tested with a single earthquake with 0 magnitude value | Magnitude colour indicator of earthquake item will be green | As expected |  |
| Detail activity, with register fragment active tested with a single earthquake with 6 magnitude value | Magnitude colour indicator of earthquake item will be red | As expected |  |

## 4.3 Integration Testing

During these tests the application was tested whilst all components were put together as a whole product. Unless mentioned in the test case these tests were performed after the application had started and loaded the API data.

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Expected Result | Actual Result | comments |
| App started for first time | Main activity loads with API acknowledgement and successfully got earthquake data feedback | As expected |  |
| From main activity with register fragment active user press menu button in toolbar | Menu should open showing 3 fragments available to select | As expected |  |
| From main activity with register fragment active and filter fragment open, filter of depth 7 applied | Earthquakes with depth of 7 displayed in list | As expected |  |
| From main activity with register fragment active and filter fragment open, filter of depth -7 applied | No earthquakes will appear | As expected |  |
| From main activity with register fragment active and filter fragment open, filter of depth 0 applied | No earthquakes will appear | As expected |  |
| From main activity with register fragment active and filter fragment open, filter of magnitude 1.4 applied | Earthquakes with magnitude 1.4 will be displayed | As expected |  |
| From main activity with register fragment active and filter fragment open, filter of magnitude -0.1 applied | Earthquakes with magnitude -0.1 will be displayed | As expected |  |
| From main activity with register fragment active and filter fragment open, filter of magnitude 0 applied | Earthquakes with magnitude 0 will be displayed | As expected |  |
| From main activity with register fragment active and filter fragment open, filter of depth between 3 and 6 applied | Earthquakes within range will be displayed | As expected |  |
| From main activity with register fragment active and filter fragment open, filter of magnitude between 3 and 6 applied | Earthquakes within range will be displayed | As expected |  |
| From main activity with register fragment active and filter fragment open, filter of date in future applied | No earthquakes displayed | As expected |  |
| From main activity with register fragment active and filter fragment open, filter of date before last year applied | No earthquakes displayed as earthquake data only from last hundred days | As expected |  |
| From main activity with register fragment active and filter fragment open, filter of date in data range applied | Earthquakes between selected dates displayed on list | As expected |  |
| From main activity with register fragment active and filter fragment open, filter of depth greater than 7 applied | Earthquakes with depth greater than 7 shown on list | As expected |  |
| From main activity with register fragment active and filter fragment open, filter of magnitude greater than 0 applied | Earthquakes with magnitude greater than 0 shown on list | As expected |  |
| From main activity with register fragment active and filter fragment open, filter of depth less than 7 applied | Earthquakes with depth less than 7 shown on list | As expected |  |
| From main activity with register fragment active and filter fragment open, filter of magnitude less than 0 applied | Earthquakes with magnitude less than 0 shown on list | As expected |  |
| From main activity with register fragment active and filter fragment open, filter not applied and sorting by date in ascending order | list will display oldest earthquakes first | As expected |  |
| From main activity with register fragment active and filter fragment open, filter not applied and sorting by date in descending order | list will display newest earthquakes first | As expected |  |
|  |  | As expected |  |
| From main activity with register fragment active and filter fragment open, filter not applied and sorting by most northerly in descending order | list will display most northerly earthquakes first | As expected |  |
| From main activity with register fragment active and filter fragment open, filter not applied and sorting by most easterly in descending order | list will display most easterly earthquakes first | As expected |  |
| From main activity with register fragment active and filter fragment open, filter not applied and sorting by magnitude in descending order | list will display earthquakes with highest magnitude coming first | As expected |  |
| From main activity with register fragment active and filter fragment open, filter not applied and sorting by depth in descending order | list will display earthquakes with highest depth value coming first | As expected |  |
| From main activity with register fragment active and filter fragment open, filter not applied and sorting by distance from me in descending order with device location disabled | User will receive error feedback and sort will not change | As expected |  |
| From main activity with register fragment active and filter fragment open, filter not applied and sorting by distance from me in descending order with device location enabled | List will display earthquakes closest to user first | User receives error feedback and sort did not apply | This issue is due to location getting code which worked when tried on a physical device but appears to not work on the emulator due to the version of android and permission getting. To fix this the code was rewritten to focus on supporting the emulator and android easy permissions were implemented to try and make getting permission more reliable |
| From main activity with register fragment active and filter fragment open, filter not applied and sorting by distance from in descending order with Glasgow input to sort by text box which appears | List will update and display earth quakes closest to Glasgow first | As expected |  |
| From main activity with register fragment active and filter fragment open, filter not applied and sorting by distance from in descending order with blank input to sort by text box which appears | User will receive error feedback and sort will not change | As expected |  |
| From main activity with map fragment active and filter fragment open, filter not applied and sorting by distance from me in descending order with device location enabled | Map markers will appear green if they are closer to user and redder the further away they are from user position | As expected |  |
| Earthquake item pressed on main activity with register fragment active |  | As expected |  |
| Earthquake location marker pressed on main activity with map fragment active |  | Error occurred and application closes | This issue was caused by the id given to map pointers not resetting when markers are wiped and new markers are placed on the map. During the test run the map fragment was reloaded causing this issue to occur and this was fixed by checking if the index was out of range of the earthquakes list and adjusting the index of each marker if needed. |
| Earthquake point pressed on main activity with graph fragment active |  | Error occurred and application closes | This was caused by the hack used to tie an index to each earthquake point generating rounding errors. The solution to this was modifying |