**Mobile Platform Development Coursework**

<https://github.com/StevieMoss/EarthquakeApp.git>

Design report

The task for the mobile platform development coursework was to create an android app that can parse the XML feed from the British Geographical Survey that records information about earthquakes that have occurred in the previous 100 days and display the earthquakes with the most largest magnitude, the deepest and most shallow earthquakes and the most northernly/southernly/westerly and easterly earthquakes.  
  
I began creating the android using a plain white background but, after experimenting with several different background colours, I decided upon the hexadecimal colour code #7ea6a7, which is a shade of cyan and it is a greenish-blue colour, which are considered to be a cool colour.  
  
I chose this cool colour because they are considered the hues of serenity, peace and security and cool hues bring about a more passive reaction in the brain and can create a sense of calm and friendliness.  
  
A key principle of HCI when developing any app is that the screens that the user interface can be viewed in both portrait and landscape modes. Since both Android devices (phones and tablets) can be used in either orientation screen support for both is helpful to the user.  
  
Handling any orientation changes for an activity is an important feature and if orientation changes is not handled properly then it can result in unexpected and unwanted behaviours of the application, since, when configurations changed during run time, Android restarts the running activity when the change occurs, meaning the it destroys the activity and is created again.  
  
Initially I had wished to use a Recyclerview to display the parsed earthquake data, however, after several unsuccessful attempts to achieve this, I. ultimately chose to implement the design using a ListView, which is a view that groups several items and display them in vertical scrollable list. The list items are automatically inserted to the list using an Adapter that will parse the content from the XML feed.  
  
Nowadays, maps are an essential component of modern apps and especially with an app that tracks earthquakes and I chose to at least try to implement this feature since it is was an essential component for this app.

Android integrates Google Maps in the application and Google provides a library via Google Play Services be able to use maps. In order to use the Google Maps API, the application must be registered on the Google Developer Console and enable the API.

The spinner component provides a quick way to select one value from a set of values and the spinners are just a drop down-list. In a default state, a spinner shows its currently selected value. It provides an easy way to select a value from a list of values.  
  
The choice of a search view in Android was one of the specifications of the app and the SearchView provides a user interface to search queries submitted in the app. The SearchView widget can be implemented over ToolBar/ActionBar or inside a layout.  
  
The last component I implemented was the Parceable component, which is the Android implementation of the Java Serializable. Parcelable can be processed relatively fast, compared to the standard Java serialization.

When interacting with an Android app, there is usually consist of more than one activity that must pass data between the other. The intent class is used to connect them together.  
  
The user runs the app and some data is displayed in the view and the user clicks on each row by implementing the Parcelable interface, which allows an object to be parsed to another component, in order to ascertain some further details regarding each earthquake.  
  
In order to successfully send data between activities is using the ptExtra and getExtra methods of intent object.

Unfortunately, despite attempting to select many more advanced components into the app, the approach was not successful.  
  
The listview managed to present the data in rows, when attempts to use a recyclerview were unsuccessful.

The function to handle the configuration change was successful as was the but the spinner, which has now had stopped displaying the items in the dropdown list, the search function doesn’t work and the maps don’t bind with the data collected either.

Testing Strategy

The approach I took to testing was similar to the approach taken in development, an iterative evolutionary approach.

As each requirement was designed and coded, it was integrated into the software and tested.  
  
A list of test cases was developed which gives an outline of the testing process and for each test case a test data form is completed, showing the input test data and expected results.  
  
Once the test case has been completed, the actual results are compared against the expected results and compared against the expected results. If the results match they are recorded as a success, otherwise it may be necessary to amend the code.  
  
All changes must be documented.

White box testing is implemented and it depends upon the tester having knowledge of the detailed design inside the modules and involves producing test data to thoroughly test every line of code and all conditions and loops in the code.  
  
Black box testing was also implemented at times and it focuses on the functional requirements of the software, using the specifications as the basis of carrying out black box testing.  
  
Black box testing should be used in addition to white box testing

Testing Document

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test No | Test Case | Input | Expected Output | | Actual Output | Screenshot |
| 1 | Testing that xml feed is parsed | Load app | | Screen with data appears | As expected | Pic 1 |
| 2 | Testing that individual data has info | Click on individual earthquake | | Screen with data appears | As expected | Pic 2 |
| 3 | Testing that map works | Press map button | | Map appears | Error | Pic 3 |
| 4 | Testing that search works | Type a word in search view | | Searched item appears on screen | Error | Pic 4 |
| 5 | Testing that spinner works | Click on item1 | | Item appears and user taken to item | Spinner not present | Pic 5 |
| 6 | Testing that spinner works | Click on item2 | | Item appears and user taken to item | Spinner not present | Pic 6 |
| 7 | Testing that spinner works | Click on item3 | | Item appears and user taken to item | Spinner not present | Pic 7 |
| 8 | Testing that landscape orientation works | Click on landscape button | | Phone changes to landscape mode | As expected | Pic 8 |
| 9 | Testing that individual data has info in landscape orientation | Click on landscape button | | Phone changes to landscape mode | As expected | Pic 9 |

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

|  |
| --- |
|  |