

# **Mobile Platform Development**

**Assessment Report**

**Andrew Craib**

**S1628364**

**BSc (Hons) Computing**

**GLASGOW CALEDONIAN UNIVERSITY**

**Iain Lambie**

**“Except where explicitly stated all work in this document is my own work.”**

**Signed: \_\_\_\_\_ACraib\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_26/03/18\_\_\_\_**

Contents

[**Mobile Platform Development** 1](#_Toc509928591)

[Design Report 3](#_Toc509928592)

[Introduction 3](#_Toc509928593)

[Main Menu Page 3](#_Toc509928594)

[Current Incidents Page 3](#_Toc509928595)

[Planned Roadworks Page 4](#_Toc509928596)

[Behind the Application 5](#_Toc509928597)

[User Interface 6](#_Toc509928598)

[Future Design 6](#_Toc509928599)

[Human-Computer Interaction 6](#_Toc509928600)

[Testing Report 7](#_Toc509928601)

[Introduction 7](#_Toc509928602)

[Testing Methods 7](#_Toc509928603)

[Black Box Testing 7](#_Toc509928604)

[White Box Testing 7](#_Toc509928605)

[Unit Testing 9](#_Toc509928606)

[Acceptance Test 9](#_Toc509928607)

[Other Testing Techniques 9](#_Toc509928608)

[Test Data 10](#_Toc509928609)

[Screenshots 12](#_Toc509928610)

[Test Log 16](#_Toc509928611)

[References 17](#_Toc509928612)

**GitHub Link:** https://github.com/andycraib/MPD-Coursework.git

# Design Report

## Introduction

The objective of this assignment was to build a Java Application using Android studio, which parses certain RSS (Rich Site Summary) feeds on the Traffic Scotland Website. The RSS feeds that were to be parsed was the Current Incidents and Planned Roadworks. The mobile application was created to use the feeds and display them in an appropriate way. Its purpose is to give the user up to date information on planned roadworks and current incidents using a simple application.

The Android Emulator which was to be used for this assignment was the Android 6.0. (Marsmallow). This emulator played the application on Nexus\_5X\_API\_27\_x86.

## Main Menu Page

The application contains a number of pages that can be navigated through to check different information from the Traffic Scotland website. When the application starts it opens a menu page that has a TextView that reads Traffic Scotland. It also has two buttons: one for Incidents and the other for Roadworks.

## Current Incidents Page

The Incidents button navigates to the current incidents page. This contains a ListView that displays the Current Incidents RSS in a readable way. As there was a lot of information to be displayed from the RSS, there needed to an option to scroll through the different incidents. This is done by using a ScrollView option. The Incidents shows different sections for each incident. In each section contains a number of different details, these details include:

* a warning image
* road where the incident happened
* reason for the incident
* and a link to the webpage.

These sections can then be looked at in further detail, this can be activated by clicking on the specified section. This then pops up a window that shows the previous details mentioned and includes further details such as location co-ordinates, the author of the post, any comments and the date the post was published.

ListView was the best option for the incidents page because it was listing large amounts of data. It also has very effective loading property. This had to be linked with ScrollView, and within the ScrollView a LinearLayout must be placed.

## Planned Roadworks Page

The Roadworks button navigates to the planned roadworks page. This page is slightly more complicated than the incidents page, this is because there are more elements involved on this page. As before this page contains a ListView and this displays the Planned Roadworks RSS in a readable way. The Current Roadworks page contained huge amounts of information, so a ScrollView was needed again. The Roadworks page shows different sections for each roadwork in progress. In each of the sections there are details about the roadwork, these details include:

* a warning image
* road where roadworks are taking place
* what the roadwork has caused and how it is being managed e.g. lane closure, verge works
* start and end date of the roadworks
* and a link to the web page.

The Roadworks sections have been colour co-ordinated depending on how long they will last for. The Green sections mean the roadworks will last less than or equal to one day. The Yellow sections mean the roadworks will last more than one day, but less than or equal to three days. Any sections where roadworks will last for more than three days will show up Red.

The sections can be looked at in further detail, this can be activated by clicking on the specified section. This triggers a window to pop. This window contains the information from the previous section and a couple more details. The extra details include:

* who the work is being done by
* diversion information
* the co-ordinates of the roadworks
* the author of the post
* any comments
* and the date the post was published.

At the top of the roadworks page there is a TextView that informs the user that they can select the Calendar Button. This will then open up a calendar and the user can then select a date to see if the roadworks are either; on that selected date, or if the duration of the roadworks runs over that date. This calendar allows the user to swipe through the months of the year, select a date, cancel their selection or confirm their selection. After confirming the date for selection, all the roadworks that are relevant to that date will not appear in the previous ListView. The calendar is generated using the DatePickerDialog method. This method comes into play when the Calendar Button is selected because it uses an setOnClickListener method. If the date that is selected is in the past or there are no roadworks in relation to that date, a notification will pop up informing the user that no entries have been found. This was done by creating a method that used a Toast object. Toast was used because the pop up only uses the amount of space required to display the message. It also means that the ongoing activity remains visible and can still be interacted with. Toasts will automatically disappear after a small period of time.

The final element in the Roadworks page is the search bar. The search bar was created using EditText. This was used to allow the user to input their own text to find specific roadworks. The EditText previously had android:text=”Enter roadwork name”, but this means that the user has delete the current text to input their own which is bad practice. To change this the EditText element now adopts an android:hint. This still shows the text “Enter roadwork name”, but when it is selected that text disappears and the user can input there own text. When the user clicks the select button after they have entered the desired roadworks name, the ListView will only show the roadworks related to the specific search. If there is no roadworks related to the search, then a notification pop up, again using the Toast Object, informing the user that no entries could be found. For both the calendar and search bar functions the roadworks can go into further detail like before.

## Behind the Application

The details of both the incidents and roadworks were parsed from their respective RSS. There were several more information details in the XML files that were excluded because they were not useful to the creation of the application. The chosen pieces of information included in the application were important details to have alongside the Current Incidents and Planned Roadworks.

For the majority of the layouts of the mobile application, the Relative Layout was used. The reason for this was because the position of a view can be specified relative to it sibling elements, meaning it is a lot simpler to dictate where buttons, textviews etc. go.

When using the ScrollView in the application an ID had to be assigned to it, so that a ListView could be implemented in it. The ListView that was created held an independent layout file because each of the separate items had to be listed as its own item. If the separate layout file had not been included in the application all of the items in the ListView would have been presented as one entity. This would have made it very difficult when trying to select separate items.

In order to do this, a unique ID was given to the ListView and this connected it to a different XML file which held two TextViews that have their own ID’s. Within the Activity file the ListView was called, and then to link to this to the Property XML file the inflate command was used. The two new TextViews were contained in the Property XML file. To initialise the two TextViews in the activity, FindViewById is used. Then the text was fixed to the proper Title and Date.

ImageView was used to pull the warning images that were in the drawable directory. Having the images made the incidents and roadworks items stand out and draw the eye of the user. The images used are the signs that are used by The Highway Code for these kind of road occurrences.

The application needed certain permissions to run certain elements of the app. One of the permissions was the Internet. This was needed because the application uses a number of URLs from the Traffic Scotland website. The other permission was the Access Network State is needed to access ConnectivityManager (which is mainly for monitoring network connections). These permissions were implemented by altering the Android Manifest.

## User Interface

The UI was kept plain but easy to navigate, this helps users to use the app with ease. Getting the functions of the application up and running was the most important part of the assessment and graphic details were not particularly important. The colour co-ordinated roadworks items where done so in a way that it catches the eye of the user. Red, which is usually associated with bad or warning was given to the longest roadworks. Yellow was given to the medium length of roadworks. And Green which is usually associated with good or complete was given to the roadworks that are almost finished.

## Future Design

When nearing the end of the development of the application, most of the required specifications were achieved in the project. However, when the user selects to look at the incidents or roadworks in more detail, there was planning to have a map of the location of the incident or roadworks. Unfortunately, the emulator did not have Google Play Services, so this meant that the method chosen did not work. With more time the application could have implemented the map, but the assignment deadline was fast approaching.

## Human-Computer Interaction

There were several design principles that were implemented in this application. The design principles used were:

* Balance – The design of the application is similar throughout the whole project. There are no radical changes from page to page.
* Emphasis – The design in the application has colours that stand out in the roadworks section. It also has pictures to make the incident or roadwork distinct.
* Movement – The design of the application is simple to move through and easy to work.
* Repetition – The application carries the same design throughout all of its pages.
* Proportion – The elements of the application all relate well with each other in terms of size.

# Testing Report

## Introduction

The purpose of the testing report is to show that the developer has made good use of testing strategies to ensure that the application works properly. This will also demonstrate that the developer has tested the application running. The Traffic Scotland App that was to be developed for this assessment, contains two main functions that run operate on it. These functions are to show the Parsed RSS Feeds for Current Incidents and Planned Roadworks. These feeds are also available on the Traffic Scotland Website.

The software development life cycle used for the development of the application was the Iterative Life Cycle. This ensured after the completion of each stage there was a working prototype, that had been tested for bugs or errors. These different iterations all worked towards the working application and met the assessment requirements.

## Testing Methods

The application was mainly tested using the Android Studios Android Emulator - Android 6.0. (Marsmallow). Running the Android Emulator usually ensured the program was without syntax and logic errors, but sometimes the application would not open if there were connection issues between the java classes and the XML layouts.

### Black Box Testing

Using the Android Emulator showed how the application runs as if it was in a completed state. This is an example of black box testing, because the underlying structure was not visible to the tester and they could only test what was displayed on the emulator. This is also known as Behavioural Testing, this was used to give an indication of how the application would behave when it was being interacted with. This was usually used to test the functional side of the application. Black Box Testing was used to test the final version of the application. This allowed to developer to know if the final version worked properly and was presentable in the final report.

### White Box Testing

The Android Emulator could also be used for White Box Testing. This is the opposite of Black Box Testing in that the tester knows the internal structure of the application and should be able to predict the behaviour of the app when different options are selected. This was the most used method of testing for creating this app. It helped with a number of different aspects of the apps which will looked into in more detail. The aspects it assisted in creating and testing were:

* Navigation – The developer would create buttons that would move to certain pages. The developer knew the expected output of clicking certain buttons e.g. if the Incident button moved to the incident page. If this output was not given, then this could be easily identified as a problem.
* Design - The design of the application was to be simple and easy to use. Through testing in this manner, the developer could see if certain elements of the app were in the correct place. So ListViews for Incidents were in the Current Incidents related page and ListViews for Roadworks were in the Planned Roadworks related page. There was an issue with the images on the two elements. The image displayed was a roadworks image, so this was fine for the imageViews of the current roadworks item. However, this image was also displaying on the incidents items, which was incorrect. The developer tried different methods so that the program would use both the images in the drawable file, but every method stopped the application moving further than the main menu page. White Box Testing was used to test each method quickly and see if a solution could be found. Another part of the design requirements was to have both portrait and landscape layouts. Portrait layouts were already created when building the application, but Android Studio had to be used to make sure that when the emulator switched to a landscape layout that the app could adapt. Some of the activity pages adapted properly, but ones with ListViews only adapted partially. For example, when a ListView was displayed in a landscape layout the ListView would not fill the width of the screen.
* ListViews & ScrollViews – White Box Testing was very useful when trying to ensure the correct information was being parsed from the Traffic Scotland RSS feeds. There were two listviews and scrollviews in this application, so data of the both elements could easily get mixed up. This type of testing again could be used to quickly to see if the correct information was being parsed to the Incidents ListView and the same to the Roadworks ListView.
* Calendar/DatePickerDialog – White Box Testing was used to see if the calendar would pop up from the onClick method it was related to. An issue arose with this when selecting a date to see planned roadworks. Any date selected would bring up every single roadworks rather than a refined search. A selectedDates method had to be incorporated into the app so it would refine the search to only the selected dates. Also, an error message was created so that users could not select a date in the past. This type of testing allowed the developer to create different methods that would show the expected output of the calendar function.
* Search Bar/EditText – This method of testing allowed the developer to create a working search bar, but like the calendar, issues arose when trying to refine the search. The search bar would get all the items related to the text input and put them into the ListView, but when going into further detail by selecting the item it would show the wrong item. To solve this problem a method was created like that of the calendar refined search detailed view.
* Item Detail – When the items are in the ListView they can be selected in further detail. This shows the parsed information from the corresponding RSS feeds. To begin with the developer created a method that would only parse the information about the road title of the incident or roadworks. White Box Testing allowed the developer to ensure one part of the details was working then work could begin to bring in more information for the items.

### Unit Testing

The different sections of the application were all completed in an iterative lifecycle, meaning that one section was in working order before the next step was undertaken. So, to begin with the developer created the Main Menu layout of the app with the required buttons and created the Current Incidents layout. Then these two layouts were linked through an onClick method to navigate to the Current Incidents page. This had a short test to ensure that it navigated properly. The same was done for the Planned roadworks page and the respective button for navigation.

The ListView with a scroll view had to be created for the Current Incidents activity, this was done using an ArrayList. The information related to current incidents was parsed from the relevant RSS feed to the ListView to be displayed in the Current Incidents Activity. The onClick method from the Main Menu had to start this activity, so the incidents would be displayed. This was another Unit Test that was undertaken to ensure this section was working. The same was done for the Planned Roadworks page and another Unit Test was undertaken to ensure this section was working properly.

The next part was to investigate each incident into further detail, this was done so by creating a Detailed Activity that ran from the onClick of a certain item. Once this was completed a Unit Test was commenced to check that the detailed activity was working properly. The same was again done for the Planned Roadworks.

This was the completion of the Current Incidents page and this testing was carried out to ensure there was not problems with the activity.

The next section to work on was the calendar/DatePickerDialog. This had to be activated through an onClick method and take the correct roadworks for the selected dates and put it in a ListView. This was another Unit Test to ensure that Calendar function was working properly.

The final section to work on was the search bar/EditText element. This had to read in the text given by the user and then use an onClick method to bring up all the roadworks related to the text given. This was Unit Tested to confirm that the search bar was working.

### Acceptance Test

The final test conducted on the application was the Acceptance Test. This was used to guarantee that the application worked as a whole and that all elements were working. This was conducted using both a portrait and a landscape layout.

### Other Testing Techniques

Break points were used throughout the construction of the application. This was used to debug any problems and it allowed for a greater understanding of the flow of the program and how certain methods should work.

All testing was documented over the course of the application building process.

# Test Data

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Data** | **System: Traffic Scotland Application** | **Author: Andrew Craib** | **Date: 26/03/18** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Number** | **Test Case** | **Input** | **Test Data** | **Expected Output** | **Actual Output** | **Screenshot** |
| 1 | The user will run the application and the homepage will appear | Run ‘app’ will be selected on Android Studio | Selection of Run ‘app’ option | The homepage will appear in the Android Emulator | The homepage appears in the Android Emulator | [1](#Screenshot1) |
| 2.1 | On the homepage the user will navigate to the Incidents page | The user will click the Incidents button | OnClick | The Current Incidents Page will appear | The Current incidents page appears | [2.1](#Screenshot2a1) |
| 2.2 | On the Current Incidents page, an incidents item will be looked at in further detail | The user will click an incident item | OnClick | The chosen incident will pop in a new window with extra detail | The chosen incident pops in a new window with extra detail | [2.2](#Screenshot2a2) |
| 3 | On the homepage the user will navigate to the Roadworks page | The user will click the Roadworks button | OnClick | The Planned Roadworks Page will appear | The Planned Roadworks Page appears | [3](#Screenshot3) |
| 4 | On the Planned Roadworks page, a roadworks item will be looked at in further detail | The user will click a roadworks item | OnClick | The chosen roadworks will pop in a new window with extra detail | The chosen roadworks pops in a new window with extra detail | [4](#Screenshot4) |
| 5.1 | On the Planned Roadworks page, the calendar button will be selected | The user will click the calendar button | OnClick | The calendar will pop up in a new window | The calendar pops up in a new window | [5.1](#Screenshot5a1) |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Number** | **Test Case** | **Input** | **Test Data** | **Expected Output** | **Actual Output** | **Screenshot** |
| 5.2 | On the calendar window a date in the past will be selected | The user will click a date in the past and click ok | Select 25/03/18  OnClick | A warning message will notify the user that no entries are found | A warning message notifies the user that no entries are found | [5.2](#Screenshot5a2) |
| 5.3 | On the calendar window the cancel button will be selected | The user will click cancel on the calendar | OnClick | The calendar will disappear and show the roadworks page | The calendar disappears and shows the roadworks page | [5.3](#Screenshot5a3) |
| 5.4 | On the calendar window a valid date will be selected | The user will click a valid date and click ok | Select the 27/03/18  OnClick | The roadworks on or running on 27/03/18 will appear | The roadworks on or running on 27/03/18 appears | [5.4](#Screenshot5a4) |
| 6.1 | On the search bar invalid text will be entered | The user will search for an invalid road and click select | Test Road  OnClick | A warning message will notify the user that no entries are found | A warning message notifies the user that no entries are found | [6.1](#Screenshot6a1) |
| 6.2 | On the search bar a valid road will be entered | The user will search for a valid road and click select | A77  OnClick | In the ListView all the Roadworks related to the search will show | In the ListView all the Roadworks related to the search shows | [6.2](#Screenshot6a2) |
| 7 | Landscape View of the application | The user will change the view to landscape | Turn Emulator | The application will display in a landscape view | The application will display in a landscape view | [7](#Screenshot7) |

# Screenshots

|  |  |  |  |
| --- | --- | --- | --- |
| **Screenshots** | | | |
| **Number** | **Screenshot** | **Number** | **Screenshot** |
| 1  [Return](#Screenshot1Return) |  | 2.1  [Return](#Screenshot2a1Return) |  |
| **Number** | **Screenshot** | **Number** | **Screenshot** |
| 2.2  [Return](#Screenshot2a2Return) |  | 3  [Return](#Screenshot3Return) |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Screenshots** | | | |
| **Number** | **Screenshot** | **Number** | **Screenshot** |
| 4  [Return](#Screenshot4Return) |  | 5.1  [Return](#Screenshot5a1Return) |  |
| **Number** | **Screenshot** | **Number** | **Screenshot** |
| 5.2  [Return](#Screenshot5a2Return) |  | 5.3  [Return](#Screenshot5a3Return) |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Screenshots** | | | |
| **Number** | **Screenshot** | **Number** | **Screenshot** |
| 5.4  [Return](#Screenshot5a4Return) |  | 6.1  [Return](#Screenshot6a1Return) |  |
| **Number** | **Screenshot** |
| 6.2  [Return](#Screenshot6a2Return) |  |

|  |  |
| --- | --- |
| **Screenshots** | |
| **Number** |  |
| 7  [Return](#Screenshot7Return) |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Log** | | | | |
| **Test Number** | **Date** | **Success** | **Failure** | **Reason for Failure/Solution** |
| 1 | 20/3/18 | True |  |  |
| 2.1 | 20/3/18 | True |  |  |
| 2.2 | 22/3/18 |  | True | Images only showed the roadwork image and not the incident image |
| 2.2 | 22/3/18 | True |  |  |
| 3 | 20/3/18 | True |  | Had to make do with the problem |
| 4 | 22/3/18 |  | True | <br /> in the start date and end date of roadworks would not disappear |
| 4 | 22/3/18 | True |  | Had to make do with the problem |
| 5.1 | 24/3/18 | True |  |  |
| 5.2 | 24/3/18 |  | True | The error message did not show |
| 5.2 | 24/3/18 | True |  | The error message was working using Toast |
| 5.3 | 24/3/18 | True |  |  |
| 5.4 | 24/3/18 |  | True | Calendar date selection would bring up every planned roadworks |
| 5.4 | 24/3/18 | True |  | Calendar date was now showing related roadworks for date in ListView |
| 6.1 | 25/3/18 | True |  |  |
| 6.2 | 25/3/18 |  | True | Search bar gets the refined search, but detailed view shows the wrong item selected. |
| 6.2 | 25/3/18 | True |  | Method was created like that of the calendar refined search detailed view. |
| 7 | 26/3/18 |  | True | The app in landscape view was not adapting the ListView to the width of the mobile page. |
| 7 | 26/3/18 | True |  | Had to make do with the problem |

# Test Log

# References

https://www.getty.edu/education/teachers/building\_lessons/principles\_design.pdf

https://www.techopedia.com/definition/3821/iteration