

**Mobile Platform Development**

**MHI322959**

**Session 2017/2018**

Coursework Submission: Design Report

Nick Connell

Matriculation Number: S1623944

## **Introduction**

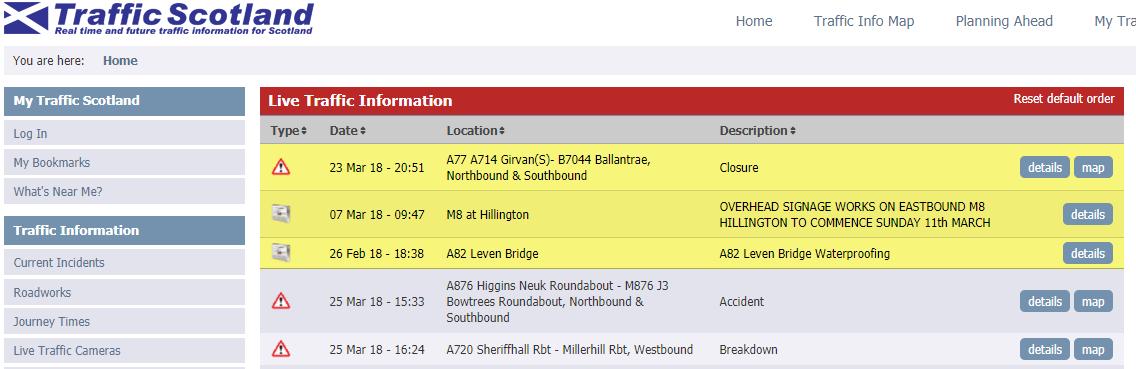
The aim of this design report is to demonstrate the use of certain graphical components, their layout, and the reasoning behind these elements being chosen over other approaches which could have been taken. It will go into detail about what the design tried to achieve in aiding user experience amongst other key decisions which were researched and then later implemented.

## **Aims of the Design**

This section is going to discuss what the design of the application aimed to achieve.

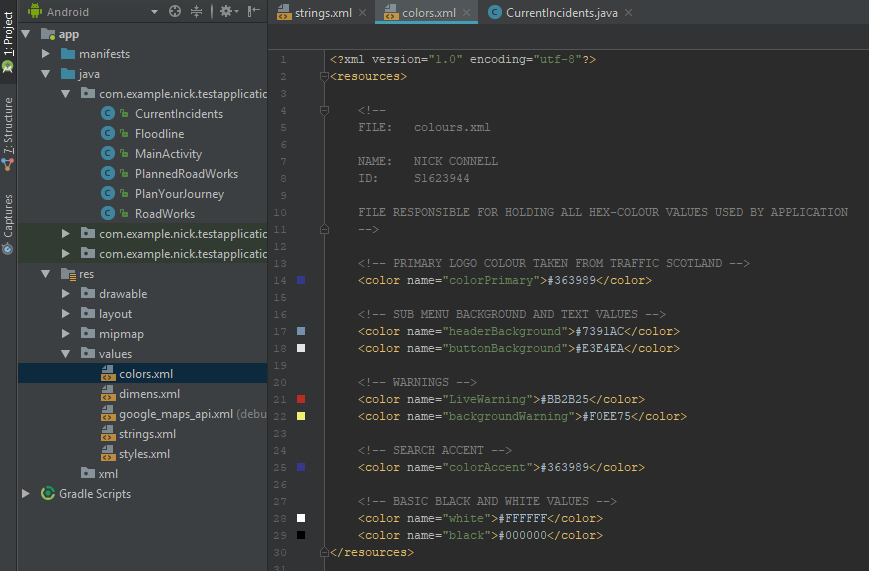
One of the most fundamental goals which must be achieved by any product, software application, program, game, or the actual devices which access such applications, is to make it as simple as possible for the user. This means never allowing for the user to feel lost or unsure as to how they should proceed at any given moment and having a user interface which is friendly and simple to use. This is achieved through clear instructions (if necessary) and appropriately labelled, self-explanatory buttons.

**Diagram 1.0 – Screen from official website of Traffic Scotland**

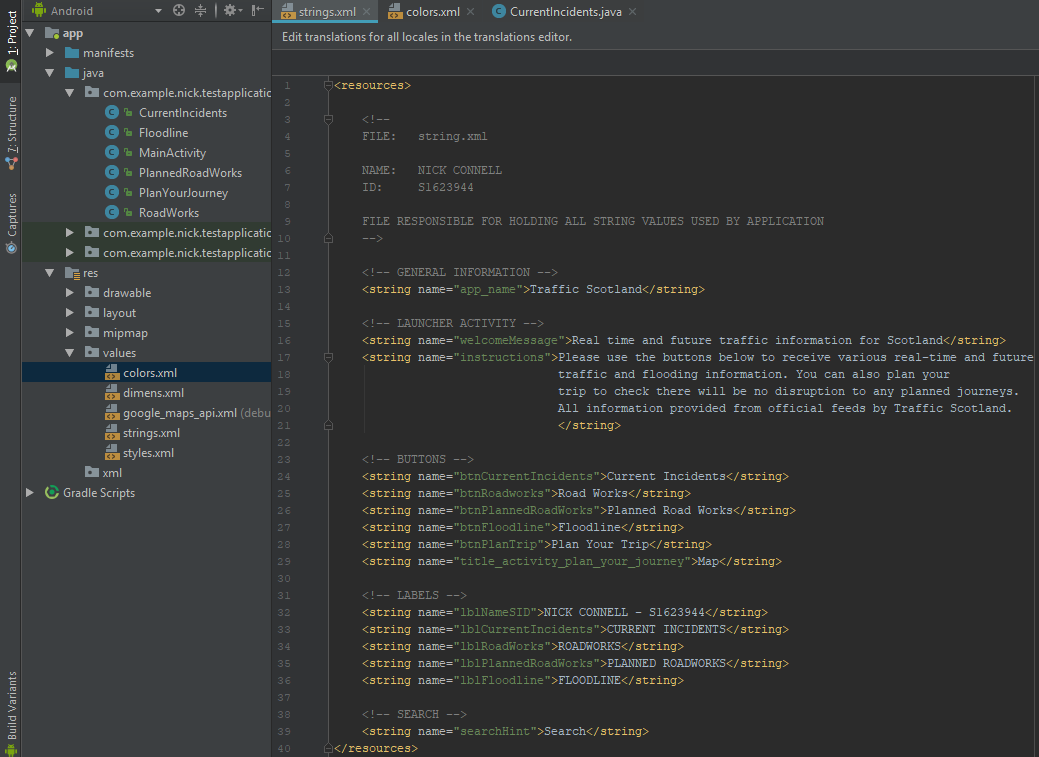


Another key aim during the design process was to make the app look and feel as aesthetically close to the official website for Traffic Scotland taking into consideration both colour scheme and word choice. As can be seen in Diagram 1.0, Traffic Scotland uses the phrase “Real time and future traffic information for Scotland”. This phrase, amongst some other text were taken directly from the official site to help achieve a sense of reliability and consistency should a user change between the android application and the website. Further to this use of text, the colours used are also transferrable and this was achieved by obtaining the hexadecimal values of the assorted colours used by Traffic Scotland. As can be seen in the respective diagrams below, Diagram 1.1 depicts the colours file held in the values directorate, followed by Diagram 1.2 showing the strings file which captures some of the text implementation.

**Diagram 1.1 – Application of Hexadecimal Colour Values**

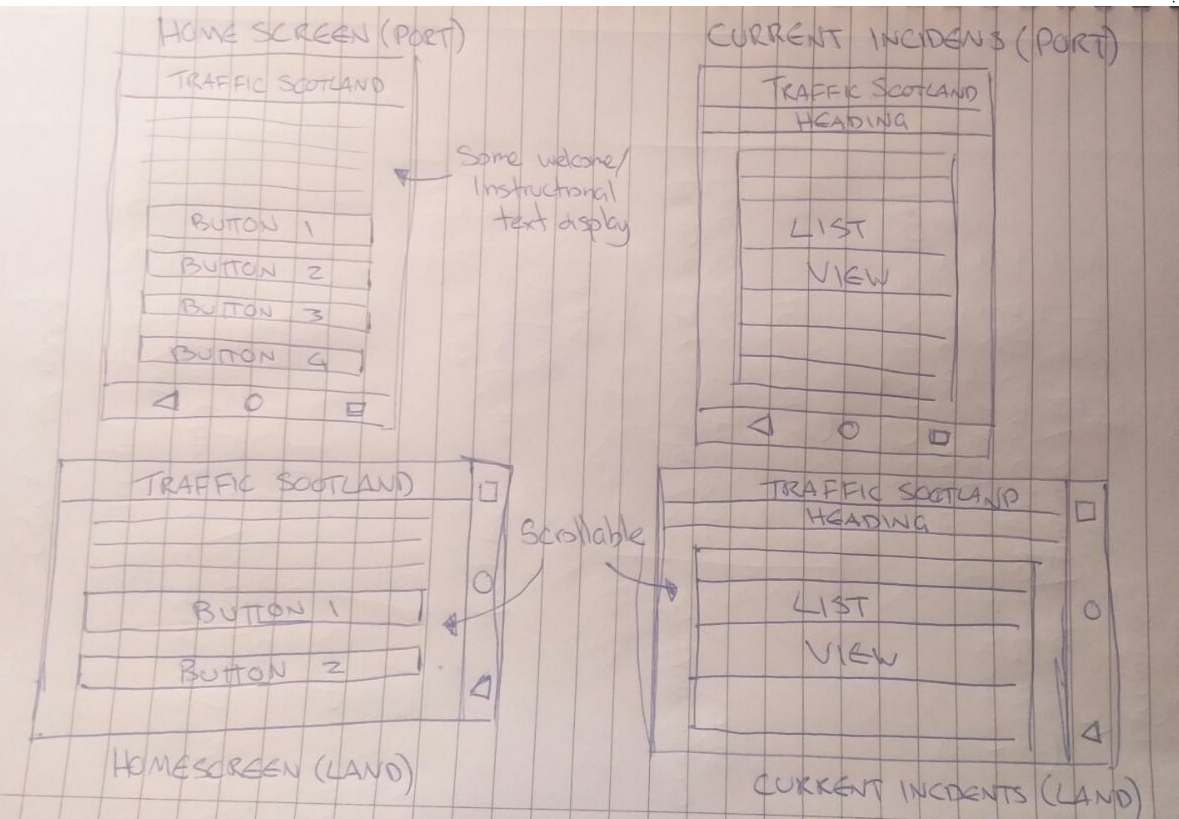


**Diagram 1.2 – Application of String Values**



Initially when planning the user interface, considering the colour schemes, word choice and the fact I planned on making the application as simple to use as possible, I began to draw up some rough sketches of what I set out to achieve. An example of one of these sketches can be seen in Diagram 1.3 below, which shows a rough layout plan for the application in both Portrait and Landscape orientation.

**Diagram 1.3 – Rough sketch of Application Interface**



**Screen Orientation**

There are many devices which are likely to be able to access an application such as this, and as such significant forethought must be placed on how you can create a usable interface which is compatible with as many devices. In this application I adopted a simple approach which displayed all items in a vertical manor regardless of the screen orientation. I considered this as the most simple approach to take which would be easy to implement on the many target devices which could be accessing the application.

## **Justification of Components Used**

This section will look at the original interface designs as per the sketch in Diagram 1.3 for the proposed application and discuss why a layout option was chosen to help achieve this based on its characteristics against others. It will then final cover the individual components which have been chosen to make up the user interface.

**Design Layout**

The very first step taken in this project was to draw up a rough sketch of what I expected the application should look like, what components it comprised of and how they would be laid out.

Having researched various layout options available to me, it became quickly apparent which options would be more suited than others. The following is a brief overview of some of these layout options which were considered for the application:

* **Relative Layout**

A view group that displays child views in relative positions. The position of each view can be specified as relative to sibling elements (such as to the left-of or below another view) or in positions relative to the parent RelativeLayout area (such as aligned to the bottom, left or centre).

* **Linear Layout**

I found this layout option to be the most likely to fit the needs of this project as it simply organises children into a single horizontal or vertical row with a built-in scroll bar if the length of the window exceeds that of the screen. I believe this to be particularly appropriate in that it is expected there is going to be multiple rows of traffic information coming in from the RSS feed.

* **Absolute**

“A layout that lets you specify exact locations (x/y coordinates) of its children. Absolute layouts are less flexible and harder to maintain than other types of layouts without absolute positioning.” – Taken straight from Androids official documentation – this layout option was quickly ruled out as it did not fit the ethos this project has tried to follow in keeping things simple as possible. Especially on this instance where there are clearly more appropriate layout options available to meet the needs of the proposed design screens.

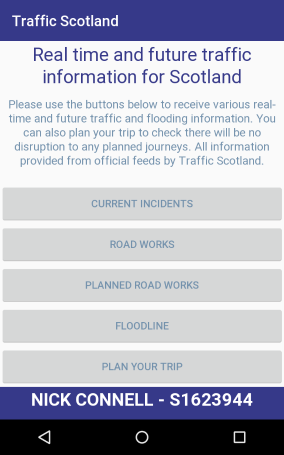
From the above, and with some practice of trying each, I decided to use the Linear Layout as I thought this was most suitable for displaying each of the elements which I wished the application to include. Items such as buttons in a neat, one on top of the next style could easily be implemented with text blocks to accompany them for the main activity scree, for example.

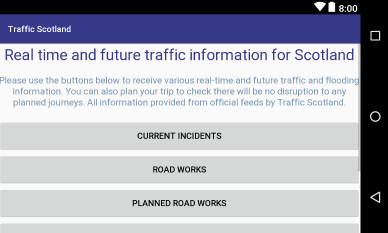
**Linear Layout with Adapter**

After it was determined that the Linear Layout View was likely the best candidate for the job, my next task was to work out which object would be best suited for displaying the individual feed items in a neat and concise manner. For this, I opted for List View, which as the name suggests is a simple list which stores one item on top (or below) of another. I felt this was a consistent option and in-keeping with the reasoning behind the choice of the Linear Layout Another benefit of using the List View is that due to the dynamic natures of this application’s content, this means the layout isn’t populated until runtime. This is achieved by using as subclass of the AdapterView named Adapter which retrieves data from the RSS feed and converts each entry into the List View.

**Buttons and Text Views**

In keeping with a simplistic design principle, I have tried to keep the pages as clean as possible with no clutter as it can detract from the purpose and value of an application if there is too much happening on the screen. As a result, on screen elements have been kept to a minimum. An example of this comes from the launcher activity (activity\_main). The page consists simply of 2 text views and the 5 buttons which will take the user to it’s respective page/activity. The names of each button – and indeed their colour scheme are exactly that of the Traffic Scotland Website to ensure complete consistency in designs. Examples of this can be found below in Figures 2.0 and 2.1 respectively which show both portrait and landscape orientations of the main activity.

**Diagram 2.0 – Main Activity (Landscape) Diagram 2.1 – Main Activity (Landscape)**



## **Implementation of HCI design Principles**

This section is going to discuss the HCI considerations which have been accounted for in development of this application and go on to discuss further options which it could later incorporate following further development.

**Consistent User Interface**

A key factor taken into consideration is the uniform look which this application adheres to. The user will feel confident when navigating between each screen that displays RSS feed information as they all have the exact same layout, colour scheme and formatting. This aides to the comfort and reliability of the application.

Further to this, the placement and sizing of the buttons assists with usage as they have been designed to extend the full width of the screen. This will be more comfortable for the user as they will not need to stretch their hand position to click their desired target.

**Further Options for Accessibility**

One of the biggest considerations which must be accounted for in designing of any product is that everyone can use it. This includes people who may have disabilities and may need some assistance to get the best use possible.

Further development of this application could possibly incorporate extra accessibility options such as audio options for both input and output, various colour schemes with strong contrasts to assist users with visual problems and advanced keyboard options for users requiring input assistance.



**Mobile Platform Development**

**MHI322959**

**Session 2017/2018**

Coursework Submission: Testing Report

Nick Connell

Matriculation Number: S1623944

## **Introduction**

This report sets out to document the testing activities which took place during development of the application. It will also set out plans for future testing and how further efficient and in-depth testing could further improve the overall performance of the application.

## **Testing Types & Techniques**

Testing primarily comes falls under two technique categories: Black Box and White Box.

• **Black Box Testing:**

This form of testing is carried out with no prior knowledge of the systems and its workings. This is typically carried out by users who have been invited to do so who will have no bias whilst carrying out their tests.

• **White-box Testing:**

Contrary to this the Black Box technique, this approach sees the users with some previous knowledge or at least an understanding of how certain functions and interfaces should react and respond. This project primarily follows this technique as the single developer of the application (in its current state) will carry out all initial testing.

In a professional environment, both techniques would be utilized extensively having a range of users carry out tests in various forms of testing types, including Unit testing, interface testing and system testing amongst many others.

Two testing strategies which may also be utilized are Top-Down and Bottom-Up testing. As their names suggest, top down involves testing the application at a very low level, and testing the bottom from the very smallest, individual components, in isolation to see if they work independently of each other. In the instance of Bottom-Up, these tests are usually carried out during Unit testing. The testing at this stage of development will incorporate both techniques.

## **Interface Testing**

This section of testing will include several types of testing of the interface to check the application is displaying as expected in both portrait and landscape orientation of the device. This is an important phase of testing because it ensures all on screen elements are positioned as per the specifications and their respective reason for being placed where they have been chosen to (such as full width buttons as outlined in the HCI section of the design document).

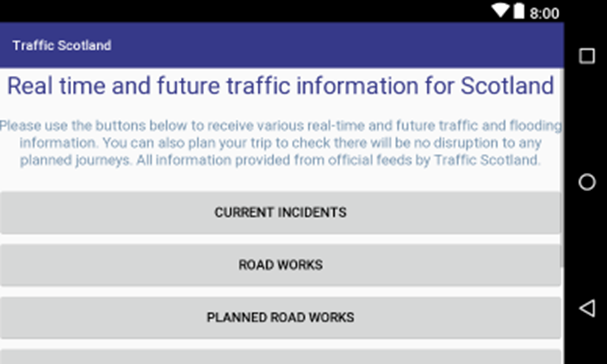
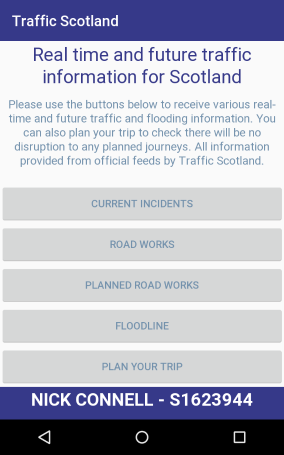
The following subsections will describe what is being tested for, the expected result and the actual result. If there have been any discrepancies found these will also be documented. However, if the result is as expected, there will be no further notes. There will also be some screenshots during development to accompany these results.

* 1. **Main Activity (Portrait and Landscape)**

Description: Check main activity displays both orientations in a similar, consistent manner.

Expected result: Pages should look similar when orientation changes

Result: Success

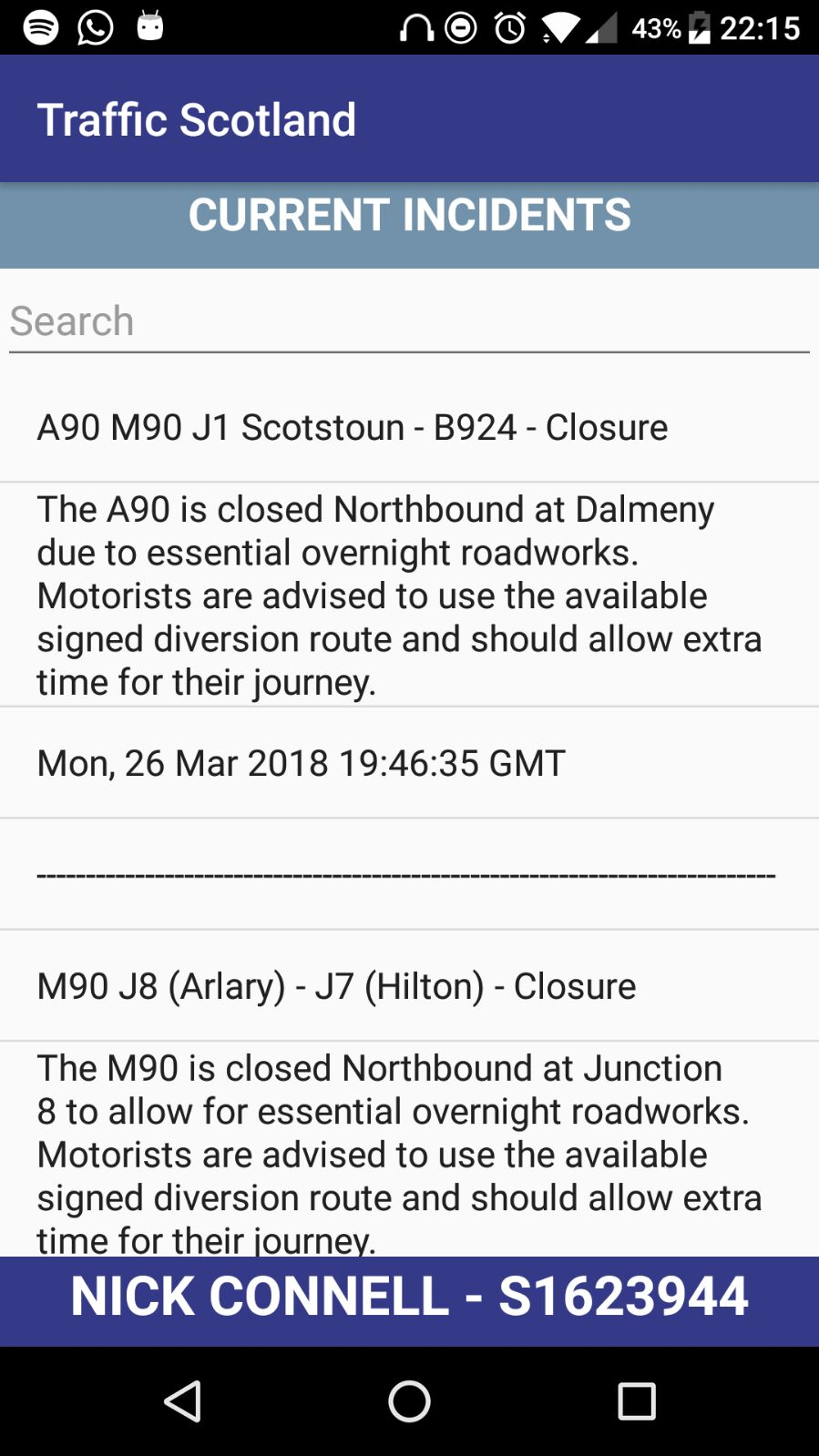


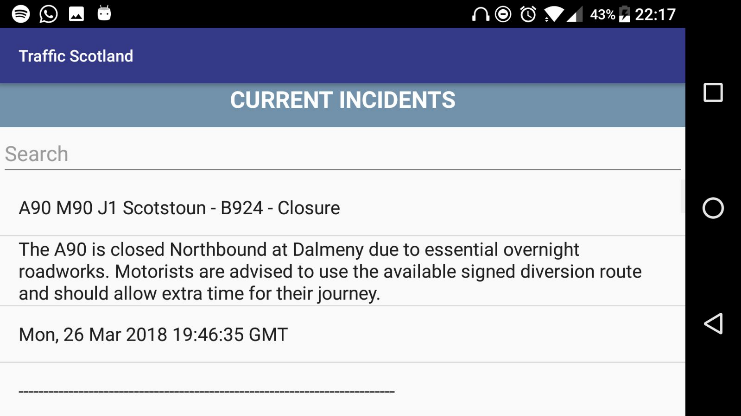
**3.2 Current Incident (Portrait and Landscape)**

Description: Check main activity displays both orientations in a similar, consistent manner.

Expected result: Pages should look similar when orientation changes

Result: Success



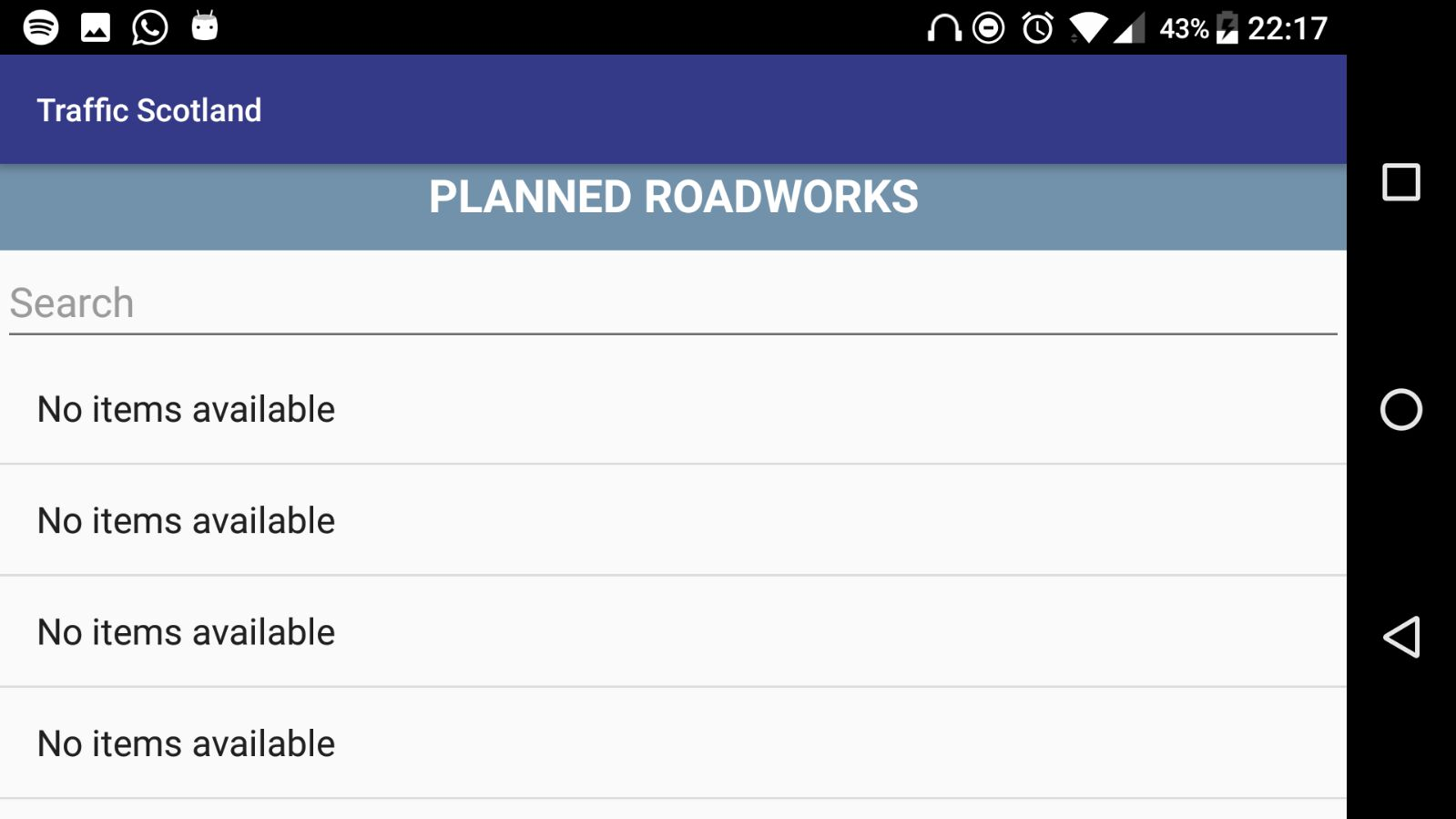
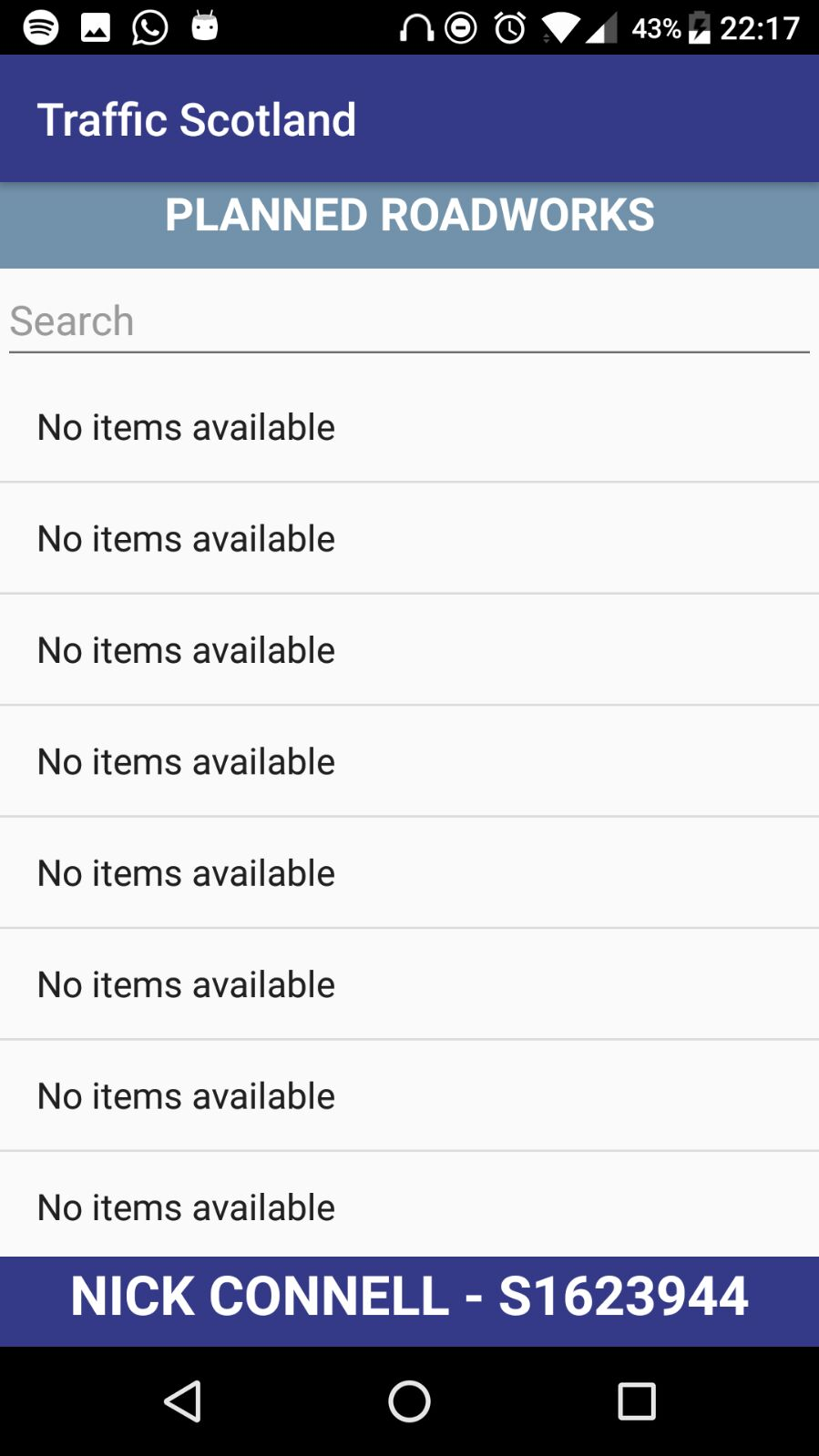


**3.3 Planned Roadworks (Portrait and Landscape)**

Description: Check main activity displays both orientations in a similar, consistent manner.

Expected result: Pages should look similar when orientation changes

Result: Success



## **Functionality Testing**

This section is going to focus on the testing of the various functions and other relevant code which makes up the functionality of the application. The code being tested will be taken from two files: CurrentIncidents.java and PlannedRoadWorks.java, as per the specification, to demonstrate various testing activities and their results.

One of the main tools which I utilized during this process was to use the Android Studio Logcat to keep track of notable events through the applications usage. These are mainly categorised through simple lettering denotations specifying the significance of a certain action taking place (or not taking place). Some of these include:

* e = Error
* w = Warning
* i = Information
* d = debug
* v = verbose

The following tables will be used to test each of the functions which are used in the main java files. The tables will give a brief description of the intent of the function, display the name of the function, the test data being used, the expected result, the actual result and any further comments as necessary. Each test will also include a reference to a diagram found at the bottom of this document for proof of testing.

**4.1 Connection Testing**

The following test is taken from the Planned Road Works activity/java file. The specified URL used was: <https://trafficscotland.org/rss/feeds/plannedroadworks.aspx>

Incorrect url: https://trafficscotland.org/rss/feeds/plannedroadworks.aspx.WRONG

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **Purpose** | **Data** | **Expected Result** | **Actual Result** | **Comments** | **Diagram Number** |
| getInputStream | Open a connection to XML feed which will supply the application with data | Expected: URL as above. | Returned XML data and successful logcat message | Success | Success | 4.1.1  4.1.2 |
| getInputStream | Open a connection to XML feed which will supply the application with data | Unexpected: Broken/incorrect URL | App to crash, error message in logcat | Success | Could add toast message and some sort of catching mechanism to state the URL is incorrect and take user back to home page. | 4.1.1  4.1.3 |

**4.2 Parsing and Displaying XML feed**

The following documents the function and subsequent processes taken to retrieve specific elements from the feed and display these in the application.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **Purpose** | **Data** | **Expected Result** | **Actual Result** | **Comments** | **Diagram Number** |
| doInBackground | Parse XML feed and pick out appropriate information into a list and displaying it in the application | Encoding:  iso-8859-1 | Display XML data in application and increment counter | Success | Success | 4.2.1  4.2.2  4.2.3 |

**4.3 Empty List/No XML Data to Display**

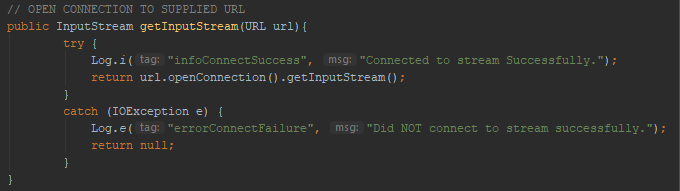
The following test is taken from the Floodline activity which, at the time of writing has no incidents to show. A simple exception has been written to display a message on screen should this occur.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **Purpose** | **Data** | **Expected Result** | **Actual Result** | **Comments** | **Diagram Number** |
| doInBackground | To handle the exception of no data being able to display. | Empty XML feed. | On screen message | Success |  | 4.3.1 |

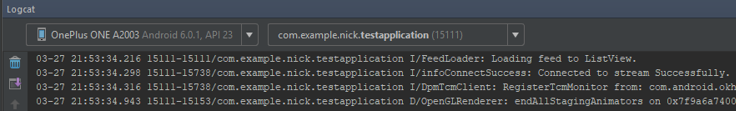
## **Proof of Testing**

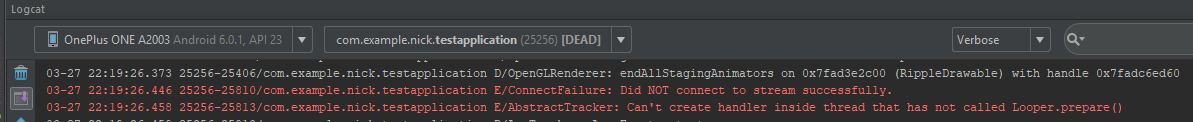
The following images are taken from testing sessions.

**Diagram 4.1.1 – Connecting to Stream Function**

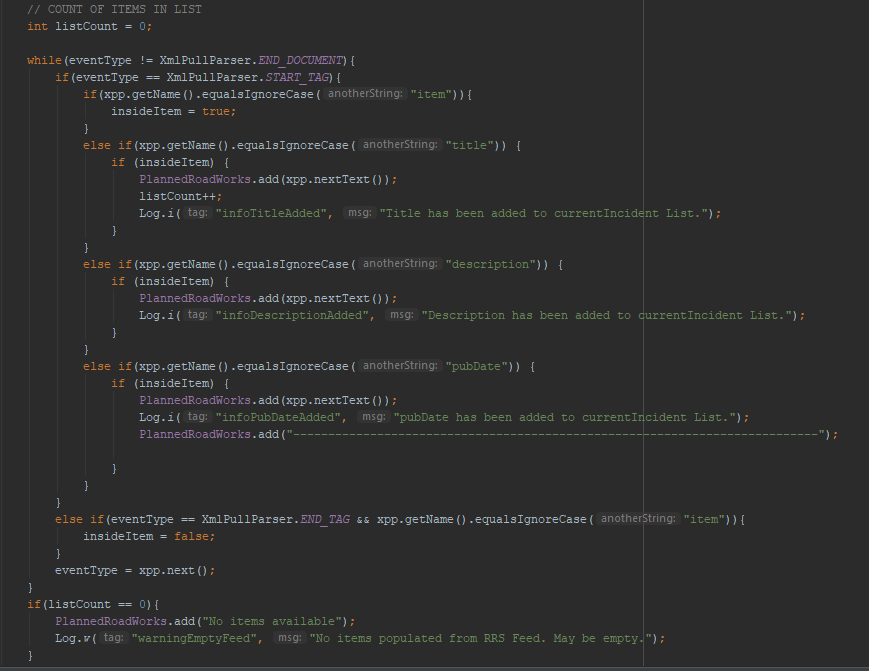


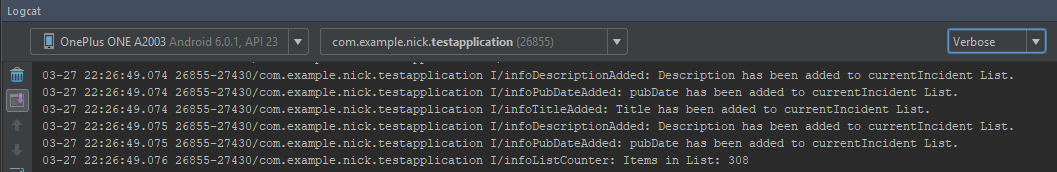
**Diagram 4.1.2 – Successful Logcat Message**



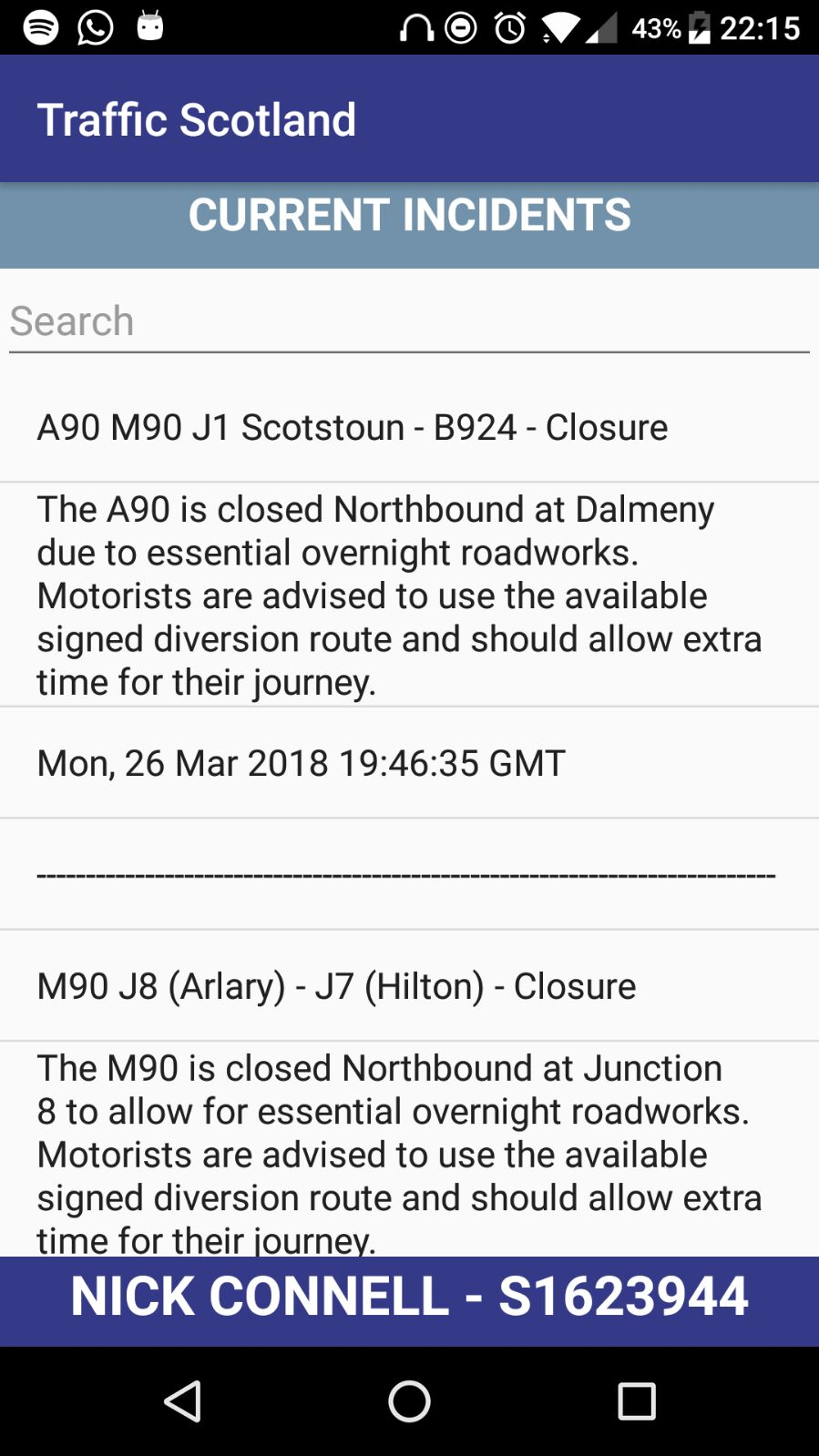
 **Diagram 4.1.3 – Error in Logcat Message**

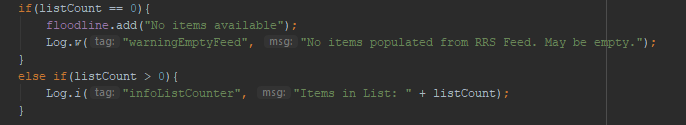
**Diagram 4.2.1 – Parsing Data**



 **Diagram 4.2.2 – Logcat Message**

**Diagram 4.2.3 – Displaying of XML**



 **Diagram 4.3.1 – Exception Handler for Empty Feed**

**Diagram 4.3.2 – Displayed Message**

