

**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).

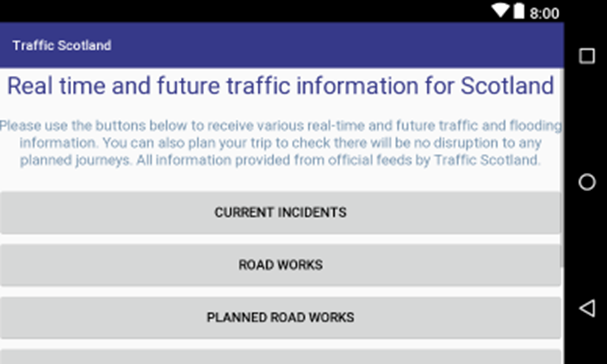
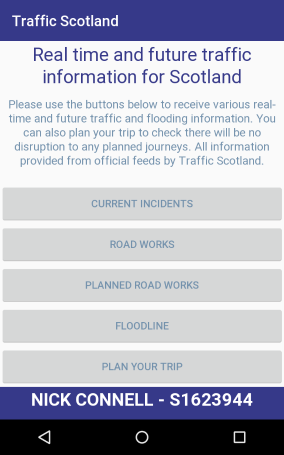
The following subsections will describe what is being tested for, the expected result and the actual result. If there have been any discrepancies found these will also be documented. However, if the result is as expected, there will be no further notes. There will also be some screenshots during development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

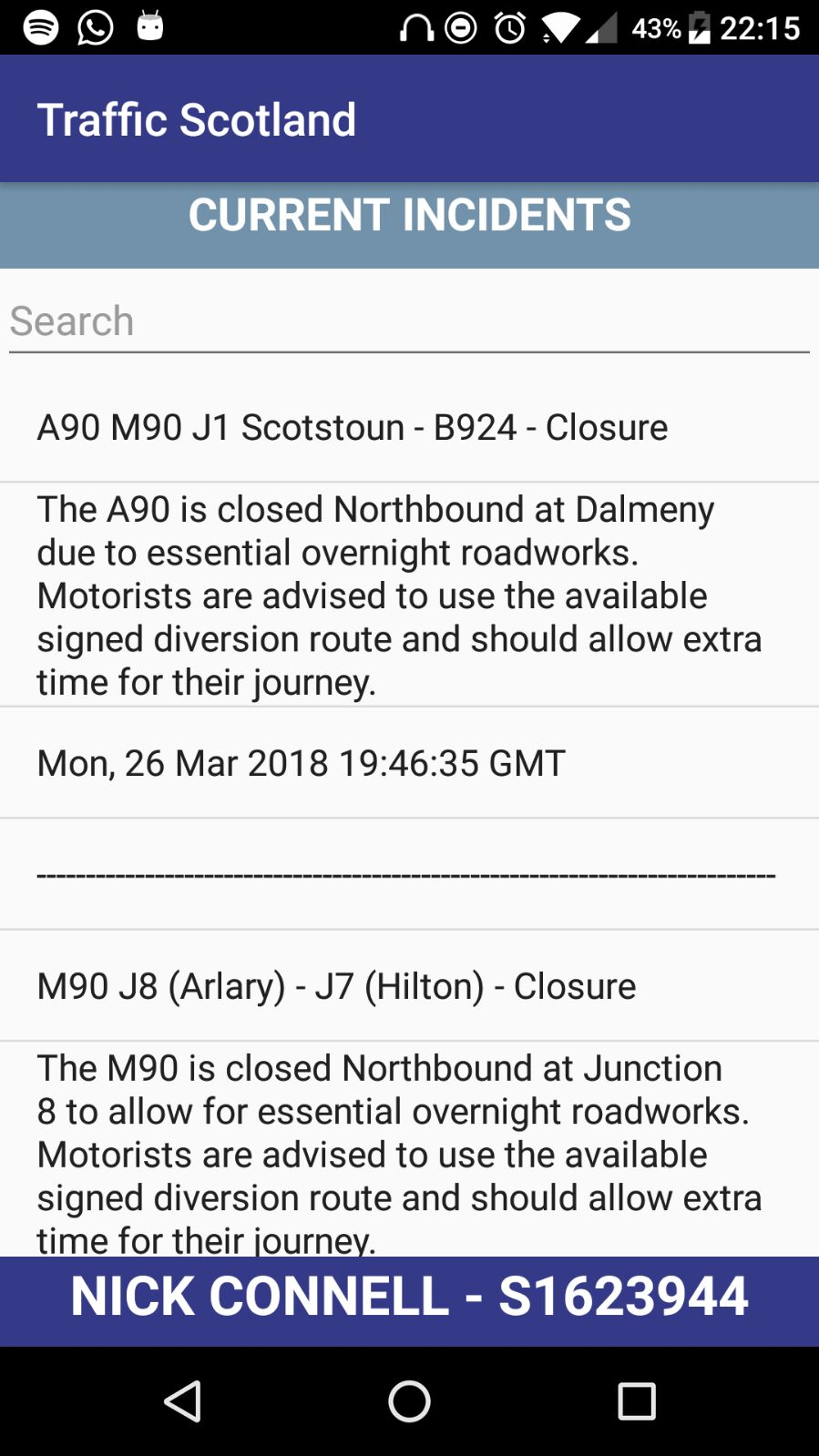


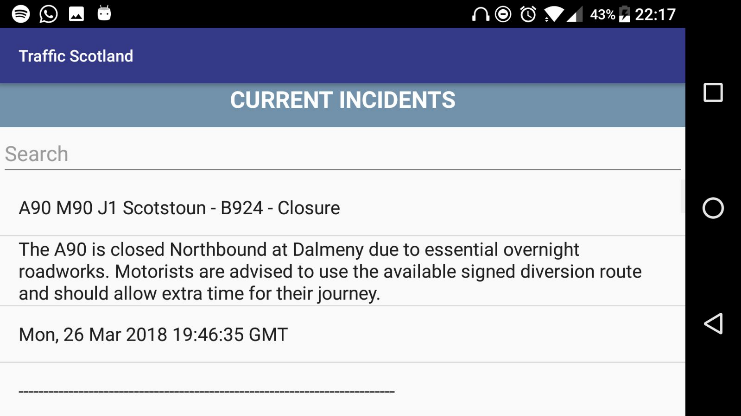
**3.2 Current Incident (Portrait and Landscape)**

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

Expected result: Pages should look similar when orientation changes

Result: Success



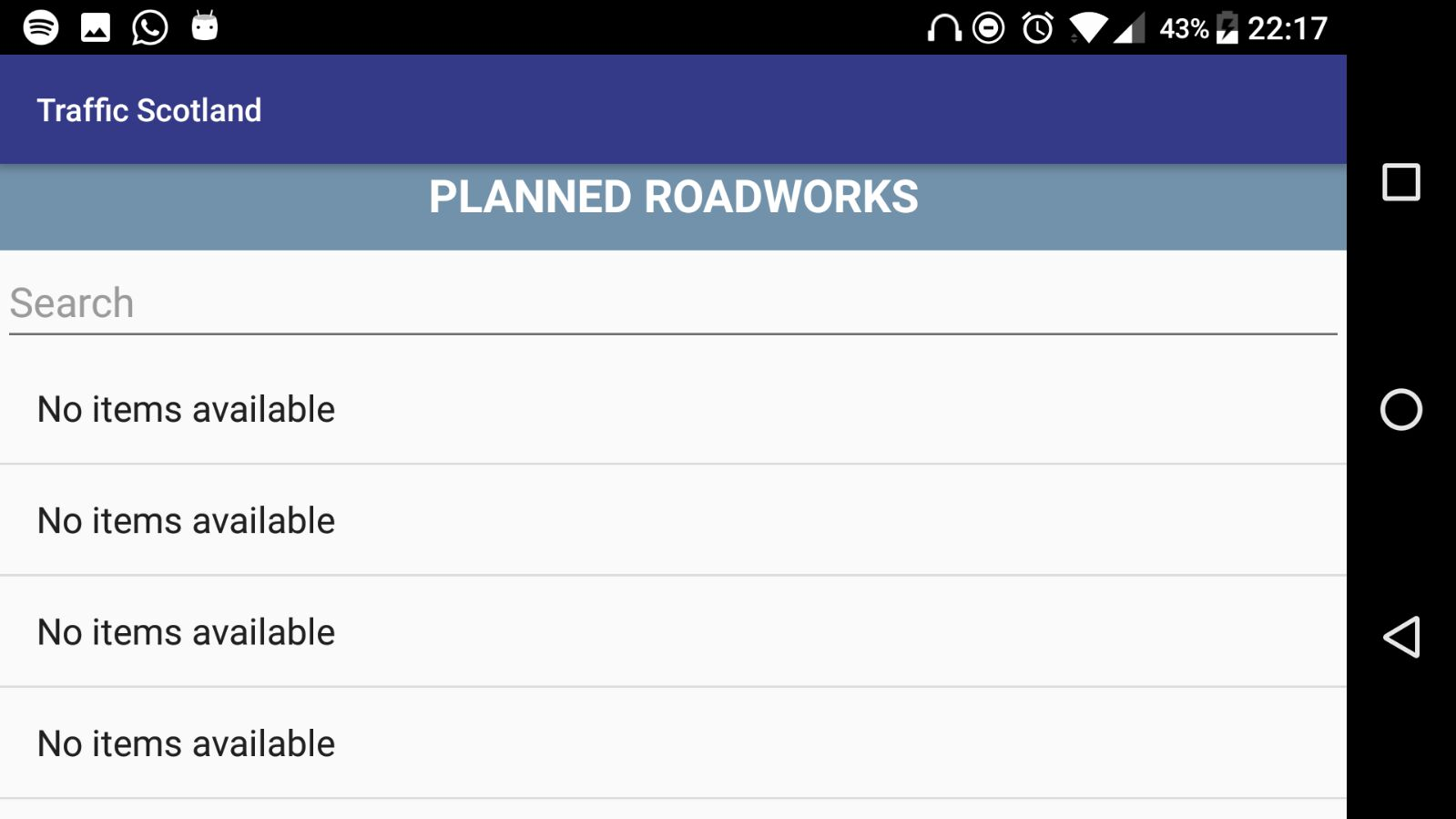
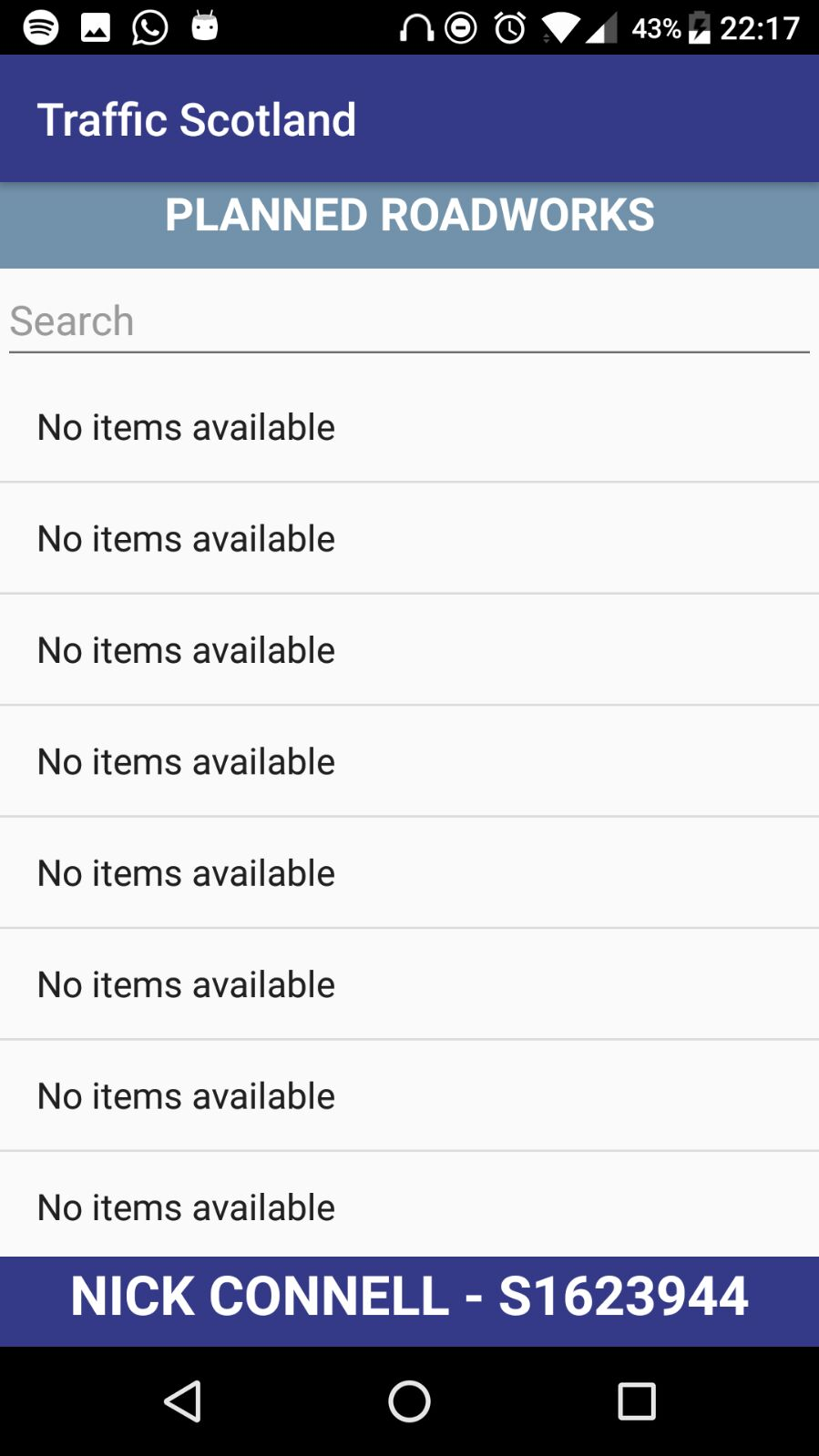


**3.3 Planned Roadworks (Portrait and Landscape)**

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**

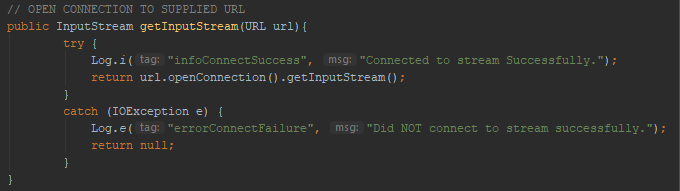
The following test is taken from the Floodline activity which, at the time of writing has no incidents to show. A simple exception has been written to display a message on screen should this occur.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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 |

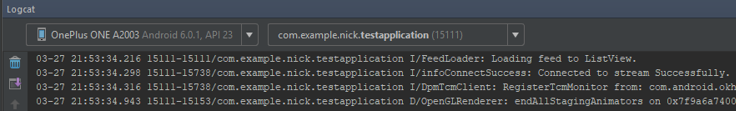
## **Proof of Testing**

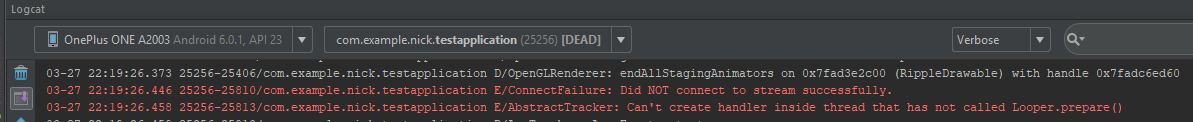
The following images are taken from testing sessions.

**Diagram 4.1.1 – Connecting to Stream Function**

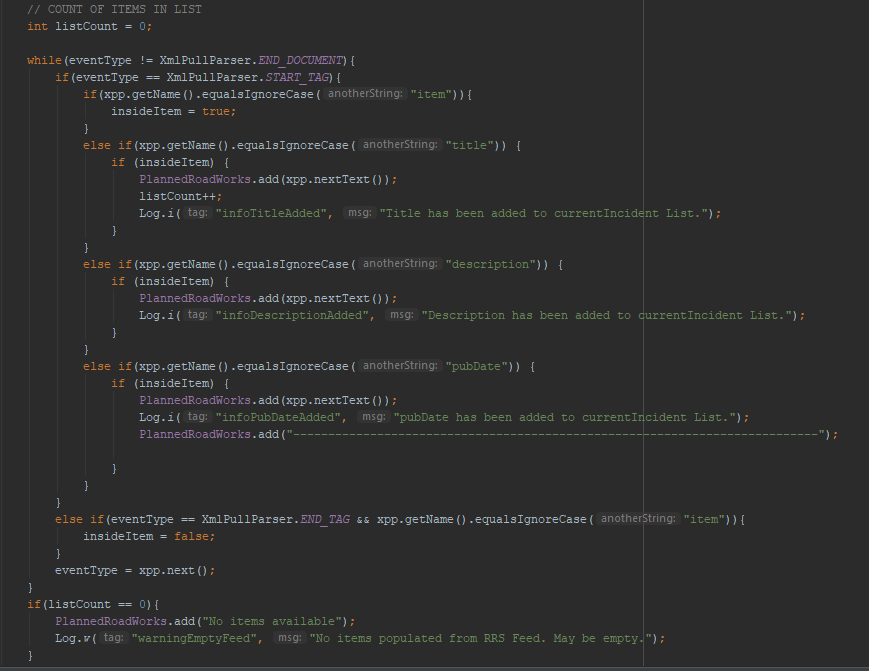


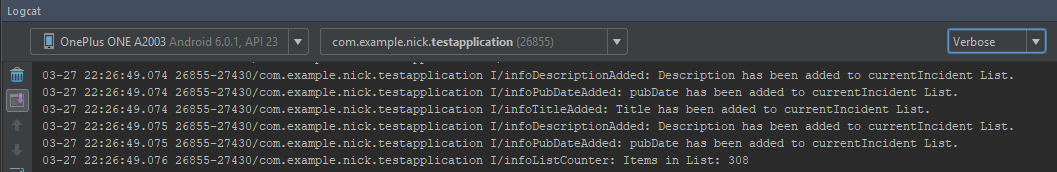
**Diagram 4.1.2 – Successful Logcat Message**



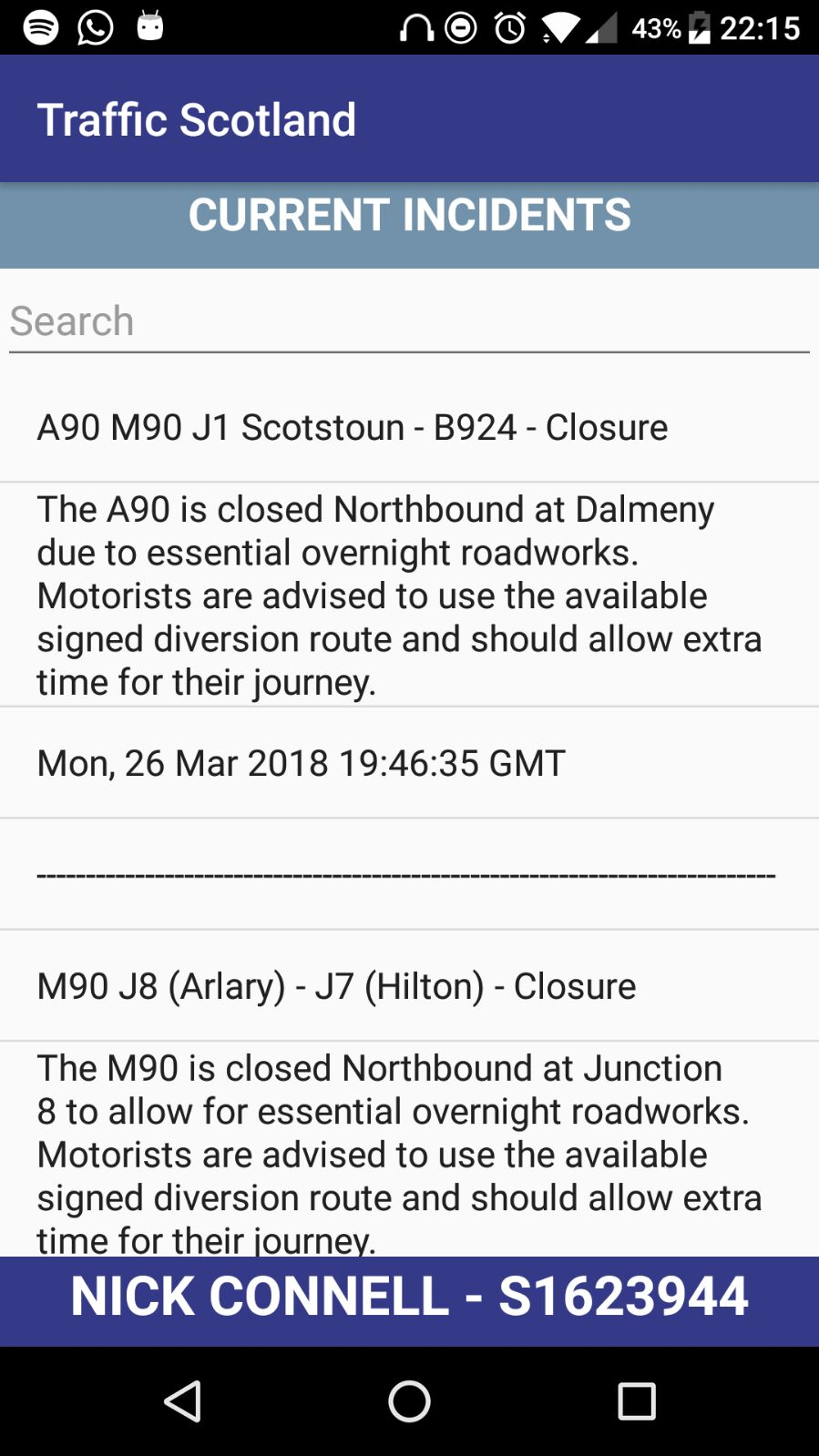
 **Diagram 4.1.3 – Error in Logcat Message**

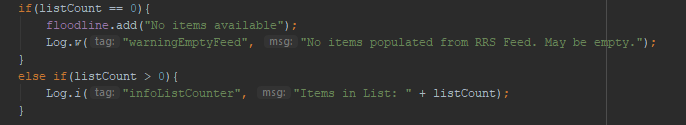
**Diagram 4.2.1 – Parsing Data**



 **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**

