Ross Hendry

S1227760

Coursework Report

Mobile platform development

Table of Contents

[1. Design Report 2](#_Toc509966597)

[1.1 Graphical Components implemented with Justification 2](#_Toc509966598)

[1.1.1 MainActivity Class 2](#_Toc509966599)

[1.1.2 Current Roadworks Class 3](#_Toc509966600)

[1.1.3 Planned Roadworks Class 4](#_Toc509966601)

[1.2 Layout and XMLPullParser 5](#_Toc509966602)

[1.3 HCI Design Principles implemented 6](#_Toc509966603)

[2. Testing 7](#_Toc509966604)

[2.1 Types of Testing 7](#_Toc509966605)

[2.2 Test Cases 8](#_Toc509966606)

# 1. Design Report

## Graphical Components implemented with Justification

There are very many useful graphical components readily available to be implemented in Android Studio for building applications. During the development of the Traffic Scotland App, an effort was made to capture as many of these components as possible to ensure of its compliance to the major design principles, which will be discussed further in section 1.3 of the report. This report will discuss the components used in chronological order as they were created in each class. With any java project, the variables will be declared at the top of each class. These variables will then be used to link in with the XML pages and their components using findviewbyid().

### 1.1.1 MainActivity Class

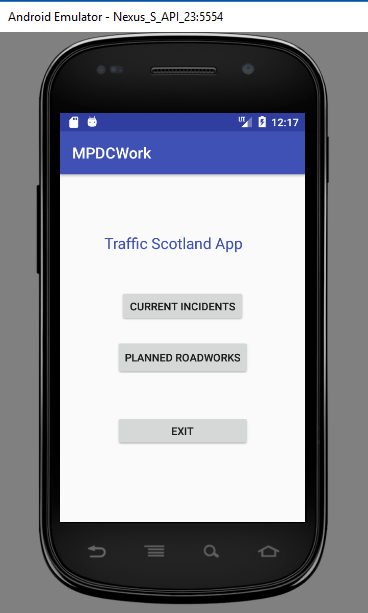
The first class to be created was MainActivity and its purpose was solely to allow the user to select one of two options on the screen and in essence, it is a menu. The menu consisted of three buttons in the centre of the screen labelled “Current Incidents”, “Planned Roadworks” and “Exit” with the latter being solely to exit the application. When the user clicks on any of the three buttons they will be taken to the new screen of their choice. This was implemented in the code by using an OnClickListener which will act upon the user’s interaction, in this case a click. The code required to carry out this task is through the use of intent to switch activities on the click of the button, with adjustments made to the androidmanifest file to allow the switching of activities. Using the button component seemed like the most logical option as it allows the user to clearly select one of the two options on screen. A button component usually only has one singular action to carry out and for the purpose of this application, a single action was required to move to the next screen(activity). This provided a simple and easy to use menu for the user’s interaction. The other graphical component used in MainActivity was a TextView which was simply used to display “Traffic Scotland App” at the top of the screen.

Figure 1 - MainActivtiy(HomePage)

### 1.1.2 Current Roadworks Class

The currentRoadworks class was one of the two main activities of the application for showcasing the implementation of the Traffic Scotland RSS feed. It incorporated many graphical components to create a simplistic but effective user interface.

**Home Button**

The top of the page starts with a simple “Home” button to allow the user to return to the home page for extra interaction with the app, although this could have been achieved by pressing the up button on the android device. Finding the right balance of user interaction is key for applications as is reinforced by the key design principles so simple navigation tools seemed adequate.

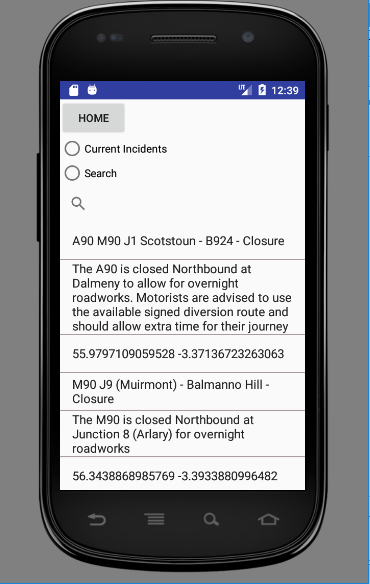
**Radio Group (Consisting of Radio Buttons)**

Figure 2 - Current Incidents screen

Then follows a group of radio buttons which allow the user to toggle between two options, “Current Incidents” and “Search”. Having a graphical component such as radio buttons implements an easy to use interface by providing two or more options in which only one can be selected. This was fit for purpose as the user could select to view all the current incidents with one radio button and then search with the other, as it was decided the user would only need one option at a time and not both. There could have perhaps been a use for buttons here instead of radio buttons, but with the aim to create an application that was appealing aesthetically, it was decided to use radio buttons. Unfortunately, when selecting search, it was unsuccessfully implemented to show the search bar when on that option, so SearchView had to exist outside of the RadioGroup, but this has been left in the code to sample how it was intended to function. Due to time constraints and perhaps in future with further knowledge and iterations, this can be implemented to perform as intended.

**Search View**

The SearchView provides a simplistic but easy to use searching interface. This allows the user to make an input to the search bar to narrow down the list of results from the parsed XML data. There is room for improvement in this app’s search bar, but the basic functionality of a search bar has been implemented. Adding extra functionality such as search bars allow the user to pinpoint a specific road or junction within the current incidents which can dramatically save time as opposed to sifting through every item on the list.

**List View**

This is an area of the application which was the vehicle in which the parsed XML data would be displayed. The XML data was parsed in a class called Retrieve Feed and contained the list of current incidents which would then be displayed in the ListView. It simply displays the data down the page in a listed fashion, which seemed appropriate for this project’s desires. There could have been scope for using other types of views, such as TextView or EditView. Although these options could have been used, it was decided that the list view was more suitable for building an RSS feed and displaying it in a list.

### 1.1.3 Planned Roadworks Class

The currentRoadworks class was one of the two main activities of the application for showcasing the implementation of the Traffic Scotland RSS feed. It incorporated many graphical components to create a simplistic but effective user interface.

As this class is almost identical to the Current Incidents class apart from showing different RSS feeds, it would seem a waste of the allocated word count to repeat the exact same information. This will mean the report can focus on giving more critical appraisal of the application as opposed to repetition. A visual of the layout has been added to show the large similarities and slight differences.

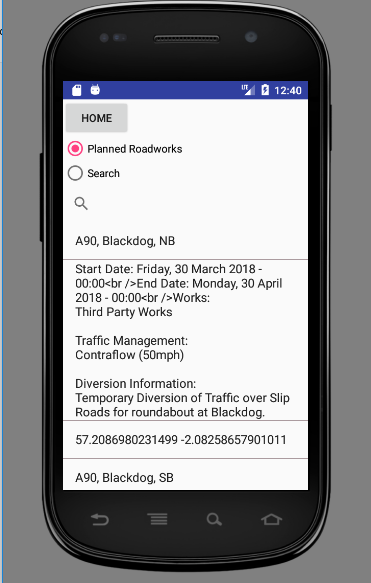


Figure 3 - Planned Roadworks Screen

## Layout and XMLPullParser

**Layout**

The applications pages had to be designed in a logical way that had positional sense for each component. This must be carried out to ensure that the interface is not confusing to the less tech savvy user. For the main activity page, a constraint layout was applied, which enables all the buttons being used to remain in a fixed position without the chance of misaligning. For the Current Incidents and Planned Roadworks class, a Linear Layout was applied as this allows for easy reading down the page in a sequential manner.

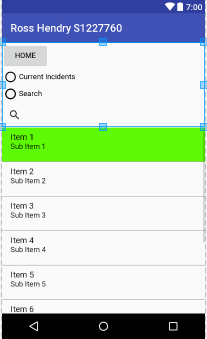
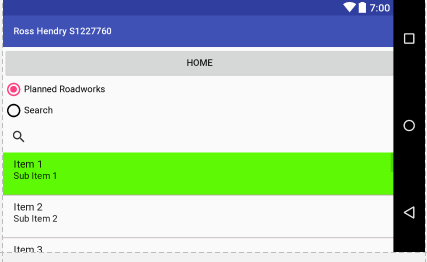
As with many smart devices now you can tilt the screen to view it in landscape as opposed to android. This is important to take into consideration when building applications so that the screen doesn’t crash or recalibrate to a messy screen when in landscape. This is avoidable, and a separate layout folder was created in “res” called “layout-land”. This folder allows you to create the landscape view for the application and automatically adopts the landscape view created in the “layout-land” folder when the device is tilted. This enables the user to either view the application in portrait or landscape.

Figure 4 - Landscape view

Figure 5 - Portrait view

**XMLPullParser**

For the purpose of parsing XML data using java, there were a few options that could have been considered but it was the XMLPullParser method that was most appropriate for this project as it is more suited to android development. This project carried out the parsing in two separate classes, “RetrieveFeed” and “RetrieveFeed2”, with the first being for the Current Incidents feed and the latter being the Planned Roadworks feed. Each class obtained the XML data supplied through the Traffic Scotland website. The data was parsed and retrieved the “Title”, “Description” and the “Georss” from the items, which were then stored in a String ArrayList, ready to be called and used by the respective classes. The class returns the variable which contains the list of ParsedXML which can then be called upon and displayed within the ListView. The project attempted to use the parsed “Georss” coordinates in order to display a visual representation of the location. It was believed that it would be incorporated using a Google API but due to time constraints this was an extra functionality which was unsuccessful in being implemented but could be a future development for the program in further iterations.

## 1.3 HCI Design Principles implemented

Many HCI Design Principled were implemented during the development of the application, as it is striving to make it an enjoyable and streamline experience for the user. One of the principles implemented was having consistency across the applications interface. Using the same types of graphical components across all of the app’s screens allows the user to adjust to the layout quickly and feel comfortable navigating around the app to fulfil their purpose. Both the Current Incidents and Planned Roadworks layout have almost identical layouts, so the user will feel comfortable switching between both.

The application was also built with shortcuts for frequent users to allow them to easily navigate between the pages. The introduction of a Home button means the user can hop to one of the pages and then back to the main menu at the click of a button. Including an Exit Button in the menu allows the user to close the program knowing fully that it has been closed, as opposed to just pressing the home button on the android device. The application has incorporated some aspects of error handling which should be sufficient to prevent the user carrying out any task which could inhibit an error and crash. This enables the user to comfortably navigate and use the features without any disruption.

# 2. Testing

## 2.1 Types of Testing

The main types of testing carried out on the Traffic Scotland RSS feed application were Functional, Smoke Testing and Black Box Testing.

**Functional**

Functional Testing is when the application is tested to ensure that each of the functional requirements set out beforehand have been included within the app itself. This allows the app developer to ensure they have achieved what has been asked of them from the start of the project. For the basis of this project, the application will be tested to ensure that it has complied with at least the basic requirements set out by the module leader Iain Lambie. Any functionality added over and above the basics would be viewed as optional or extra to what is required. The basic functionality required was an application that parsed XML data and displayed it in a clear and meaningful way to the user which this application has achieved. There was also scope for added functionality such as a search feature and using a map system to pinpoint locations with a visual representation. As the developer has managed to achieve basic functionality whilst also including extra functionality, the application passes the Functional Testing stage.

**Smoke Testing**

Smoke testing comprises mainly of non-exhaustive tests which aim to ensure that the programs main features perform as expected. This is suitable for this project as there is no need to carry out exhaustive testing as there is limited functionality for user input, rendering exhaustive testing inappropriate. By creating small tests to ensure the important functions of the program are working, this provides useful information to decide whether further testing can be carried out. The smoke testing carried out for this program was successful as each important function within the application ran smoothly, this is mainly due to having error handling such as try/catch to ensure the program doesn’t encounter any bugs which may cause it to crash. Smoke testing is used on conjunction with Functional Testing but should not be instead of.

**Blackbox Testing**

This form of testing allows the application to be tested by a user who has no understanding of the code behind the application and is able to solely test the app whilst being impartial to how the application gets to the end goal. This is good for the testing stage as the application can be tested in relation to how a normal user would go about using the app if fully deployed on a larger scale. It gives the developer feedback on aspects such as ease of use and how the user felt overall whilst trialling the app. The application was tested by the developer’s flatmate who has no previous experience in the development of android applications and was impartial to the ins and outs of the code. The feedback from the user was positive and conveyed that the application was easy to use with a straightforward layout for navigating its various features. In this case the application successfully passes the black box testing as the program ran as expected and the user was able to carry out the main features of the application without running into any errors.

## 2.2 Test Cases

|  |  |  |  |
| --- | --- | --- | --- |
| MainActivity(Menu) | | | |
| Test Case | Expected Output | Actual Output | Comments |
| Running the application | The program should run | The program runs successfully. | Successful |
| Selecting the current incidents button | The application moves to the Current Incidents activity | The application successfully moves to the Current Incidents activity | Successful |
| Selecting the planned roadworks button | The application moves to the planned roadworks successfully | The application successfully moves to the planned roadworks activity | Successful |
| Selecting the exit button | The application should exit the application | The application exits the program | Successful |

|  |  |  |  |
| --- | --- | --- | --- |
| Current Incidents class | | | |
| Test Case | Expected Output | Actual Output | Comments |
| Selecting the home button | The application should return to the MainActivity containing the menu | The program returns to the home page | Successful |
| Selecting the current incidents radio button | Display the current incidents data in the list view | The current incidents data is displayed in the ListView. | Successful |
| Selecting the search radio button | This feature will not fully work as the code is incorrect | The search button only removes the current incidents from the ListView | Unsuccessful as expected. The search had to be included outside of the radio group in order to work. |
| Searching an item in the current incidents feed | The search view should display the item containing the input | The search view displays the items containing the search input | Successful |

|  |  |  |  |
| --- | --- | --- | --- |
| Planned roadworks class | | | |
| Test Case | Expected Output | Actual Output | Comments |
| Selecting the home button | The application should return to the MainActivity containing the menu | The program returns to the home page | Successful |
| Selecting the planned roadworks radio button | Display the planned roadworks data in the list view | The current incidents data is displayed in the ListView. | Successful |
| Selecting the search radio button | This feature will not fully work as the code is incorrect | The search button only removes the current incidents from the ListView | Unsuccessful as expected. The search had to be included outside of the radio group in order to work. |
| Searching an item in the planned roadworks feed | The search view should display the item containing the input | The search view displays the items containing the search input | Successful |