Mobile Platform Development

Coursework

Alan Thomas Moore

S1436102

GITHUB LINK

<https://github.com/althomoore/mpdtrafficscotlandapp.git>

SCREEN SHOT VIDEO LINK – FOLLOW LINK AND CLICK ON:

**S1436102 - Alan Thomas Moore ScreenCast.mp4**

APK file – follow link and click on:

Traffic Scotland.apk

2018

# Design Report

To design an application of this nature it is important to take into consideration the size of the device the information will be displayed on. The application could be run on a small android device or a large tablet. This application will be designed specifically to run on an android phone. In this report there will be several discussions about how the application will operate, why it has been designed the way it is, functionality and style. Part of this will investigate how each function works and why that method was chosen to develop an application of this nature.

## Overall Layout

The standard layout for an android phone is the “Portrait” layout. Most people will hold their phone and view their applications in a Portrait style, unless the application has been specifically designed to be viewed in a Landscape fashion. For an application like this, Portrait will be more suitable due to the type of information the application will be providing the user. The application that has been developed is providing the user with an RSS feed for Traffic Scotland. The RSS feed comes from an XML file accessed through a URL. The information is then parsed and displayed on the screen that is legible for the user to read comfortably. In this instance a Landscape layout would be less suitable, but not impossible to change if it was required.

## ScrollView

The ScrollView in android studio is set up in the activity\_main.xml file in the project. This function does exactly as it describes, it allows the user to scroll through the information on the screen. Essentially it is a window in the application to display Buttons, Checkboxes, toggles, TextView, EditText and ListView. ScrollView parameters are set so that the width and height of the window fills the screen i.e. “fill\_parent”. If anything inside the ScrollView is outside the visible borders, then the user will be able to scroll over to it and make it visible on screen. For this project the ScrollView will contain Buttons and a TextView.

## LinearLayout

LinearLayout sets up the viewing area for the information to be displayed. This type of view will set strict restrictions on where items can be placed on the screen. An example of this is the buttons. There are three buttons on the screen. Because of the LinearLayout the buttons will appear on screen stacked on top of each other. The orientation can be changed from Vertical, which is the way the layout was set up, to horizontal. Horizontal will cause a lot of problems. All the items like the buttons and the TextView will be positioned in a horizontal line next to each other. This means you lose a lot of information off the page immediately. When the user opens the application, they need to get a sense of how it is all working and where things are. There are other ways of setting up the layout, but for the purposes of this project the layout was kept as a LinearLayout. LinearLayout will also keep the buttons and TextView in position whether the device is Portrait or Landscape.

## Position of Buttons

To start with, the application will need some call-to-action buttons to make the RSS feed work. It would be very easy to have the application set up that when it is opened the RSS feed will be there ready to read. That would be a lot of information for the user to read. The buttons provide a way to separate out what the user is looking for. In this instance Current incidents, Current Roadworks and Future Planned Roadworks. Traffic Scotland also provide information to the Flood line, but that has not been included in this project.

Each button when clicked will change the TextView and provide the user with the specific information they were looking for. This will still provide a lot if information for traffic all around Scotland. The user is more likely to be looking for information is a certain area. If the user is given the option of certain dates, areas affected or even specific roads, this is where check boxes, radio buttons and toggle switched would be useful for the user. The user would be able to narrow down what exactly they are looking for. Another way of doing this would be by providing a search input bar so the user could type in a specific road, area or date and get the information provided to them. The decision was made to put the buttons on top of the TextView rather than bellow because if they were put bellow they would get lost on the screen once the XML data was loaded up. Ideally the buttons would stay where they are when the user scrolls for convenience.

## TextView

The TextView is the space where the XML data will be provided. In its raw format the XML data is a series of characters and tags identifying the contents between them. The purpose of these tags is to be used as a reference point for the android java code to locate the information and display it in a more legible manner. So instead of getting:

|  |
| --- |
| <item>  <title>Lothian &amp; Borders Region - Surface water</title>  <description>Drivers are advised to exercise caution due to surface water causing difficult driving conditions.</description>  <link>http://tscot.org/01a8309</link>  <georss:point>55.89605 -3.495042</georss:point>  <author />  <comments />  <pubDate>Tue, 03 Apr 2018 19:29:04 GMT</pubDate>  </item> |

The user will get:

|  |
| --- |
| Lothian & Borders Region - Surface water  Drivers are advised to exercise caution due to surface water causing difficult driving conditions.  http://tscot.org/01a8309    Tue, 03 Apr 2018 19:29:04 GMT |

By parsing, the data can be extracted and displayed in a more suitable way. There can be a lot of information in an XML link or file and a lot of that data will not be relevant to the user. For example, the user does not need to see the GPS coordinates of where the roadworks is, but they might want to see it on a map which is where that data is useful.

In the TextView, the information can be styled to make it easier to read. The fonts can be changed, the background colour manipulated, and the layout of the information can also be formatted. In this project the text has been made white with a black background making it suitable to read with a good contrast. This makes the information clear and the user will easily understand what they are reading. Each piece of information has its own line as well. It is not all bunched together in one paragraph. There is a title, description, hyperlink and date that it was published so the user knows the information is current.

The TextView height has also been set to “wrap\_content”. This is to accommodate the amount of information that will be filled in the TextView. If the height was set to match “match\_parent” or a specific height, it would not display all the information properly. By setting it to “wrap\_content” the height will always match the amount of information in that TextView. This then allows the ScrollView to work properly. This works both in Portrait and Landscape configurations.

## ListView

ListView has a similar function to TextView. The exception is the ListView must use ArrayLists and an AdapterView to process the information. Like it says, it displays the information in a list. The information is assigned to blocks in the ListView and processed. The ListView is probably an easier way of retrieving information than the TextView, but for the purposes of this project the TextView was used. Each block of information would act like a button and when clicked on it will take the user to the location of the roadworks on a map and provide the user with further information.

## Display of information

In Portrait the information is very clear, the buttons are visible and there are clear instructions on how to use the application. The XML data repeats down the TextView and the user is able to scroll to find what they are looking for. Moving forward with this project, additional features would be introduced to make the display more palatable. All the coding for this project was a culmination of practising on the Mobile Platform Development Labs, Lecture and Tutorials. Solutions that were specifically used were XmlPullParserFactory as demonstrated in lecture 2. Plus, the AsyncTask from lecture week 5. Other solutions that would have been useful for this project was the colours example. With that solution a drop-down menu would have been developed so the buttons would not cover up most of the screen. The options for the project are limitless. It was good experience developing the application. It is a good project to make a person think about the design and the functionality working together.

# Testing Report

There were several steps involved in testing this program as it was being developed. A develop and test approach was undertaken throughout the project. Code was tested in increments to make sure it was working before moving on to the next block of code. In some cases when a block of code was working after a test, then it was just a case of repeating the code for the rest of the program to make it work with small changes to variables to relate to what it was supposed to be doing. If code was not working, many ways were used to find out what the problem was.

## Using the Android Monitor

The android monitor in the IDE was very useful in identifying problems with the code. In the early stages of developing the program there was a problem getting the raw XML data to display in the TextView.

### The Problem

The code provided for this project ran with no problems and displayed one set of data from the RSS feed in its raw format. Logic dictates if the process is repeated twice more but the variable is changed for Roadworks and Planned Roadworks, then the TextView will display raw data for those buttons when clicked.

In practise this did not occur. The TextView would only display the raw data for the first button pushed. So, when the other buttons were pressed after that all you would get is the first set of XML data.

### The Solution

To fix this problem, measures were put in place to identify if the correct information was being accessed and if the application was trying to display the information. To do this additional information would need to be displayed on the screen as the code was run. The following code was added to the public void startProgress1() to identify if the code was processing in that method:

Toast.makeText: This brings up a little dialogue box on the screen to tell the user what has been pressed.

Toast.*makeText*(getApplicationContext(), **"Alan Thomas Moore\nS1436102\nCurrent Incidents"**, Toast.***LENGTH\_LONG***).show();

urlInput.setText: This displays its own page as the information is processed and the message tells the user it is accessing the information they want.

**urlInput**.setText(**"Alan Thomas Moore\n\nS1436102\n\nGetting Current Incidents Information..."**);

The Log.e will be found in the Android Monitor

Log.*e*(**"INFO"**, **"urlInput: cleared"** );

Log.*e*(**"INFO"**, **"Currently fetching: "** + **url1**);

And finally, setTitle changes the title on the page to tell the user they are on the right page.

setTitle(**"Current Incidents - S1436102"**);

After all this information was added to the code, the program was run again. This time, the Toast.makeText appeared on the screen telling the user they pressed, for example, Roadworks. The urlIput text appeared on the screen informing the user again they pressed Roadworks and the program was running the code to fetch the relevant information. The logs were found in the Android Monitor indicating the code was running correctly. The setTitle also changed the title on the page. All this demonstrated the code was running properly and the problem might be somewhere else.

It turned out the most minor of things. At the start of the code the variable “result” was declared as [String result = “ “;]. Even though this is technically correct, it caused problems with the code when additional functions were added. The solution was to change it from [String result = “ “;] to [String result;] and later assign [result = “ “;]. By doing this it cleared the parameters every time a button was pressed and refreshed the data.

## Other testing Methods

Other ways the code was tested was to use break points and the debugger. The debugger runs through the code line by line and will keep going until it comes to a problem. By identifying a problem, the developer can fix the code and run it again, hopefully successfully next time. It could be the smallest thing like a ‘;’ not in the correct place.

The IDE was also useful for identifying problems in the code. The IDE can tell if the wrong methods or variables have been used. The predictive coding is also useful because it tells the developer if what they want to put in belongs in the code or not.