SCHOOL OF ENGINEERING AND BUILT ENVIRONMENT

Introduction to Mobile Device Programming
M3G621212-14-B: 14/15 B

Coursework Assignment

Prototype Report

Zoltan Tompa
S1112414

This prototype was mainly aimed to show the data-processing and –handling aspects of the project.
I've started to develop this aspect of the project first, before I spent a considerable amount of time on developing the graphical displays, because I believe applications in general, first have to have the functionality right and once that's there and working perfectly, developers can take time to make the graphics and visuals more user-friendly. In this project, without accurate, up-to-date carpark information the app loses its purpose and in my opinion no matter how appealing and easy to handle it is, it's still useless for the user.

The current version is fully capable of reading the specified xml-feed, however this functionality is still missing the one minute restriction, which will be implemented in the final iteration in a way, that new readings will also be checked before they get published to the actual objects.

Reading the feed uses the supplied feed-reader code snippet, however, that had been modified to skip the first row, which was causing problems in the parser. (I've applied my own fix, before it got officially patched)

With every update, initially all objects are destroyed and re-created with the data supplied from the data-feed. This decision was made because the data-feed supplied by a third-party organisation, and we have no authorisation to modify/extend the data (to add changing flags for example). Because of this, the app needs to be able to react to changes in the data-feed, like if a new carpark gets built or a current one closes down.
Hardcoding the instances and only updating the numeral values would mean faster processing, but would ignore structural changes and would make the app unable to react for events mentioned above.

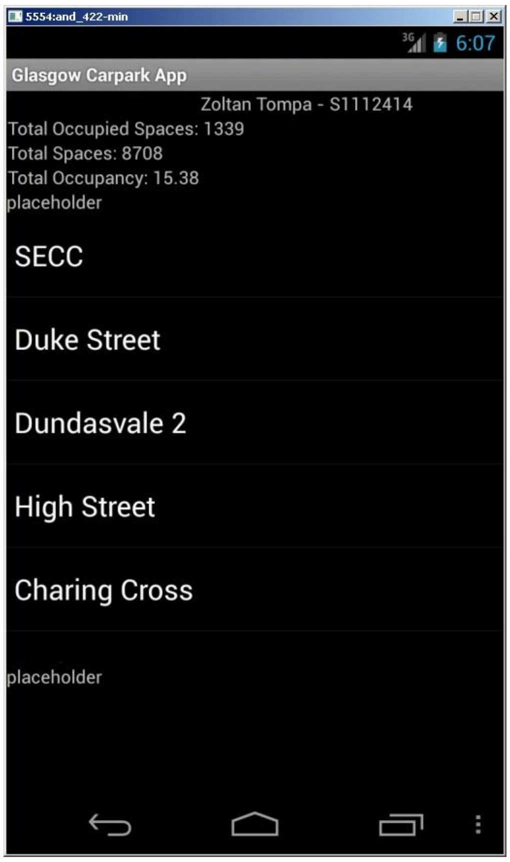
Once the data is stored as a string variable, the parser method goes through each field, and looks for the required data fields, puts their value into temporary variables and when all the information is collected for that carpark instance a new object is created and the data is transferred to that object.
Trimming of the carpark name also takes place here (using a method call in the storage class), where the last seven characters of the carpark name gets cut off as it has no meaning to the user.
After the class instance is created, it's being added to an array-list collection, to keep track of the objects in the storage class.

After the storage class had been updated, the display method takes over, and starts methods to calculate (add up) and display statistical information like the total parking spaces and the total spaces occupied city-wide. Calculating and displaying the percentage of the above is also invoked here using an overloaded method of the storage class (which is also used to calculate the individual occupancy percentages)

Displaying the list of carparks is currently done with a ListView composite element. The rows of the element are dynamically generated from the objects names. They get listed to a string-array and then passed over an adapter, which feeds it, along with other data, to the ListView element.

In portrait mode the prototype implements two LinearLayout view-groups and switch-screen functionality as well. Because elements in this orientation are placed below each other, this choice seemed the best. When the user selects a carpark on the list, the view switches to the secondary screen, where the elements values gets updated with the relevant values of the selected carpark, showing its name, occupancy and other detailed information.

However in landscape orientation, elements are laid out in a way that they fit the current orientation the best, to improve user-experience. This version uses a GridView Viewgroup to organise its members. I've also experimented with TableView, but that quite worked the way I wanted it to. Using GridView in landscape mode, on the top we can see the global statistical information in one row, including the total available spaces and overall total spaces city-wide. Displaying a calculated occupancy percentage also takes place here. Below these on the left hand side there is the same ListView element, but instead of switching the screen to show the detailed information, it gets loaded into the right hand side on the current screen, saving time and complication for the user with avoiding switching between screens and to find their way back to the carpark selector screen.
Building this functionality was probably the trickiest so far in this project.



5554:and_422-min

6:07

Glasgow Carpark App

Zoltan Tompa - S1112414
Total Occupied Spaces: 1339
Total Spaces: 8708
Total Occupancy: 15.38
placeholder

SECC

Duke Street

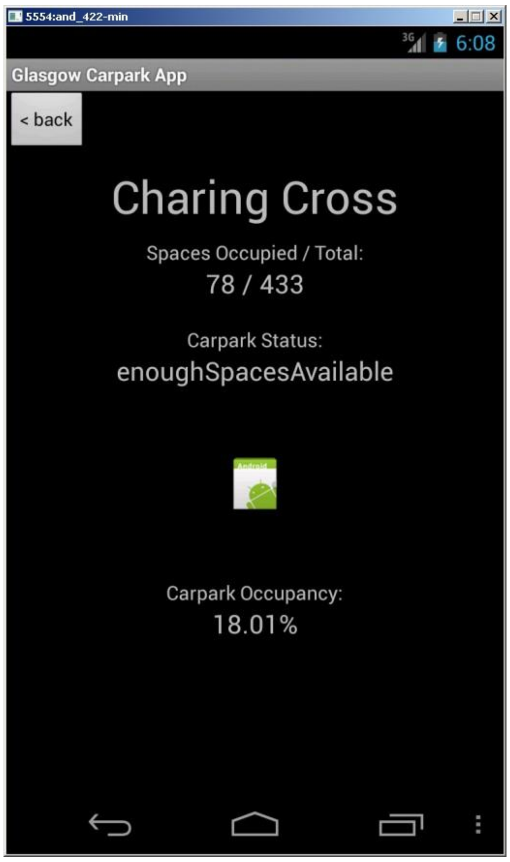
Dundasvale 2

High Street

Charing Cross

placeholder

Portrait View – Main Screen



5554:and_422-min

6:08

Glasgow Carpark App

< back

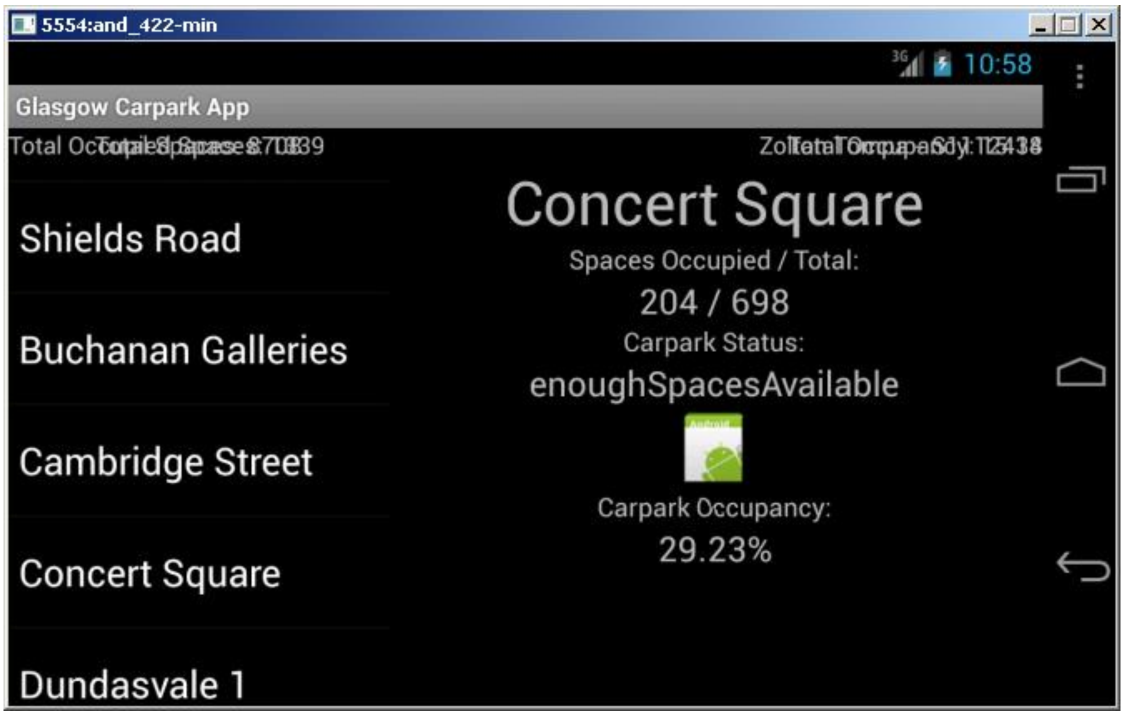
Charing Cross

Spaces Occupied / Total:
78 / 433

Carpark Status:
enoughSpacesAvailable

Carpark Occupancy:
18.01%

Portrait View – Details Screen



5554:and_422-min

10:58

Glasgow Carpark App

Total OcTotal Spaces: 8708339

ZolTotal Occupancy: 15.38

Shields Road

Buchanan Galleries

Cambridge Street

Concert Square

Dundasvale 1

Concert Square

Spaces Occupied / Total:
204 / 698

Carpark Status:
enoughSpacesAvailable

Carpark Occupancy:
29.23%

Landscape View