TEST REPORT – SCHMAPS

Table of content

1. INTRODUCTION

1.1 PURPOSE OF APPLICATION

The purpose of our application is to help students of Chalmers to find their way around campus. The application allows the user to find locations for lecture halls and getting directions to these by searching for the name of the lecture halls. The user can also list locations of restaurants, ATMs and microwave ovens. Other functions the application provides are ability to check when the buses that between the campuses depart and also to check in on the map to tag your current position for your friends to know where you are.

1.2 GENERAL CHARACTERISTICS OF APPLICATION

The application uses an internal database to store the locations of the different positions. From the query of a user the database returns the location of the query on a map view. An external database handles the check-ins of the people using the application, saving locations and erasing them after a fix time interval.

2. TEST ENVIRONMENT

To appropriately test this software you need the source code by cloning it from the git hub repository. Instructions on how this is done are written in the developer manual. To get the latest testing environment you should get the tagged release with the tag “Testing version”. After you’ve downloaded the right version you can follow the “Build and test procedure” document that is found in the “doc” directory of the git repository to set up the project properly and start testing. The tests are written for testing with a real phone, not the emulator. To have the tests complete properly you need to have a 3G internet connection or some kind of mobile data connection on your phone, since some of the tests require this type of internet connection. The reason to why is because the group felt that the 3G network is the internet connection most users will use when using the application. Ergo, the tests are testing how the application reacts whilst using a 3G internet connection.

2.1 HARDWARE ENVIRONMENT

The computer used to test was a regular ThinkPad laptop with technical specs of:
OS: Windows 7 Enterprise, 64 bit.
Processor: Intel Core 2 Duo T8100 2,1 GHz.
RAM: 2,00 GB.
Graphics adapter: Mobile Intel 965 Express.

The tests were also tested on a variety of mobile phones, the emulator was generally not used. The phones used in the tests are as following:
Sony Ericsson X10.
Samsung Galaxy S.
LG Optimus 2X.

2.2 Software environment

2.2.1 Software

The software used for the testing are as following:
Eclipse Juno version 4.2.0.
Android SDK API version 15.
Ant version 3.2.4.
Emma version 2.0.5312

2.2.2 Software settings

All the tests were run with default settings for the software used.

3. System information

3.1 System version

The version tested when writing this document is the one tagged at the git repository as "v0.11" which is the official release testing version. This one has tests covering the first release of the application.

4. Known bugs and limitations

- The telephone must be unlocked and lit up for the tests to run
- When Västtrafik has their nightly fees they stop returning one of the values used to display the bus departures so you get an error where you can't retrieve the data.
- Sometimes when entering landscape mode and going forth and back through the menus, going back to regular mode then landscape again causes the application to crash.
- An aesthetic bug of making the splash screen show a light darker grey line at the bottom of the screen

5. Test specification

The document that contains the test specification is found in the "doc" directory and is called "Test cases". The tests in this document cover the latest test release of version 0,11.

6. Automatic testing

6.1 Code coverage

We used Emma as a code coverage tool in this project and aimed for a total coverage of above 90% since we felt that having a high percentage of the code covered means hopefully less refactoring and bug fixes in the future. To have as much code covered as possible means that we can easily test the whole system once a new function has been implemented, we feel it is good practice to have a high coverage since it means less work in long-term.

6.2 Unit test

All the tests are written using the ActivityInstrumentationTestCase2 since it's the smoothest test case for testing activities. We also used something called Robotium which is a test class to assert the coupling between the activities.

7. Test results

The results of the latest testing can be found in the "Test results".pdf found in the doc directory of the git repository.