WatchMe: Test Report

1 Introduction

WatchMe is an app for remembering movies you want to watch when they are released.

1.1 Purpose of application

The main idea is to enable simple and fast movie insertions into an accessible and functional list. The user adds the movie's release date (if not possible to acquire automatically) and the app uses Android notifications to inform when it is time to visit the cinema. The app connects to IMDB and downloads related data, such as images, plot summary, etc.

1.2 General characteristics of application

The app is easy to expand upon. There are many features that can be added easily but that are not necessary for the app to work. We expect small but fast iterations in order to learn more about the Android API step by step.

2 Test environment

Tests are performed manually.

2.1 Hardware environment

We are performing the manual tests in the Android emulator installed on OS X systems.

2.2 Software environment

The tests are performed on the following OS X system versions:

- 10.6.8 (Snow Leopard)
- 10.7.4 (Lion)
- 10.8.1 (Mountain Lion)

The Eclipse editor (Indigo) is used along with Android 4.1 (emulated Google API platform level 16).

2.2.3 Software settings

No particular system settings. Internet connection is required to get movie suggestions from IMDb.

3 System information

3.1 System version

WatchMe 1.0 alpha.

4 Known bugs and limitations

(list all bugs you knew of before the test)

- A passed date can be set on a movie.
- Even though a movie cannot be added twice, a notification is still set.

5 Test specification

Tests are specified in Requirements document (watchme/doc).

6 Automatic Tests

Automatic tests are included in a separate project embedded within the application project, and is called `WatchMeTest`. To run the included tests, head to the command line and use `ant` from the `WatchMeTest` directory:

```
$ ant test
```

Make sure to test the newest application package by running `ant clean debug install` before running any tests.

6.1 Code coverage

We will use *eclEmma* as our code coverage tool.

We aim to test more than 40% of the code. We test only the core functionality, that which would have a major impact on the apps usability when not functioning. Code coverage should ideally be at least 90%, but testing is not a big focus in this course.

6.2 Nightly builds

Jenkins is used to perform nightly builds on a dedicated server.

6.3 Unit Tests

Regular Android unit test cases are used to test the domain model (for now).

7 Test Report

7.1 Functional tests

Green	Passed test
Red	Did not pass test
Yellow	Not implemented yet

Test IDs refer to the Requirements document.

Test ID	Result	Comment
1	**Success**	
2	**Success**	
3	**Failed**	*Not implemented*
4	**Success**	
5	**Failed**	*Not implemented*

6	**Failed**	*Not implemented*
7	**Failed**	*Not implemented*
8	**Failed**	*Not implemented*
9	**Failed**	*Not implemented*
10	**Failed**	Suggestions in auto complete box are implemented and functional.

Result: 3 **Success**, 7 **Not implemented**, 0 **Failed**

7.2 Unit Test Report

```
Buildfile: watchme/WatchMeTest/build.xml

-check-env:
 [checkenv] Android SDK Tools Revision 20.0.3
 [checkenv] Installed at /usr/local/Cellar/android-sdk/r20.0.1

-setup:
     [echo] Project Name: WatchMeTest
  [gettype] Project Type: Test Application

-test-project-check:

test:
     [echo] Running tests ...
     [exec]
     [exec] se.chalmers.watchmetest.httptest.HttpRetrieverTest:..
     [exec] se.chalmers.watchmetest.modeltest.MovieTest:..........
     [exec] se.chalmers.watchmetest.modeltest.TagTest:.....
     [exec] se.chalmers.watchmetest.utiltest.MovieHelperTest:..
     [exec] Test results for InstrumentationTestRunner=..................
     [exec] Time: 0.845
     [exec]
     [exec] OK (20 tests)
     [exec]
     [exec]

BUILD SUCCESSFUL
```