SE 3XA3: Test Report
TouchTime

Team #13, ELM
Livia Kelle and kellel
Evan Ansell and ansella
Matthew Po and pom

December 6, 2017

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	Version	Notes
December 6	1.0	Livia added sections 2, 3, 5, 7
December 6	1.0.1	Evan added sections 1, 4, 6, 9

1 Functional Requirements Evaluation

test-f1:
Initial State: Application installed and launched on Android smartwatch.
Input: User touches watch face twice at a given time, on the hour - 11:00 used.
Output: Distinct vibration pattern for current time - 11 long vibrations. Logcat log to indicate the screen has responded to the touch and vibration has occurred. Result: PASS

test-f2:
Initial State: Application installed and launched on Android smartwatch. Input: User touches watch face twice at a given time, at between 1-9 minutes past the hour - 11:06 used. Output: Distinct vibration pattern for current time - 11 long vibrations, 6 short vibrations. Logcat log to indicate the screen has responded to the touch and vibration has occurred. Result: PASS

test-f3:
Initial State: Application installed and launched on Android smartwatch.
Input: User touches watch face twice at a given time, at a multiple of 10 past the hour - 11:20 used.
Output: Distinct vibration pattern for current time - 11 long vibrations, 2 long vibrations. Logcat log to indicate the screen has responded to the touch and vibration has occurred.
Result: PASS

test-f4:
Initial State: Application installed and launched on Android smartwatch
Input: User touches watch face twice at a given time, at a time where the digit in the tens and ones place for the minutes is not 0 - 11:23 used.
Output: Distinct vibration pattern for current time - 11 long, 2 long, 3 short. Logcat log to indicate the screen has responded to the touch and vibration has occurred
Result: PASS

test-fa1:

Initial State: Application installed and launched on Android smartwatch.

Input: User very quickly double taps watch face.
Output: Distinct vibration pattern for current time. Logcat log to indicate the screen has responded to the touch and vibration has occurred
Result: PASS - lockout funtion succeeded in ignoring the 2nd tap.

test-fa2:

Initial State: Application installed and launched on Android smartwatch.
Input: User touches the watch during daylight savings time when the time skips an hour - simulated by changing time zone on phone.
Output: Distinct vibration pattern for current time. Logcat log to indicate the screen has responded to the touch and vibration has occurred.
Result: PASS/FAIL? - lockout prevents the daylight savings time from affecting the vibration pattern which follows our design for the lockout, but may not be a desired feature.

2 Nonfunctional Requirements Evaluation

2.1 Usability

Input/Condition: Users must tap the watch face twice, with a brief pause in between.

Output/Result: Percentage of users capable of correctly identifying the time based off of the vibrations of the smartwatch.

How test will be performed: Users are taught how to use the application with a 2 minute tutorial by the tester. Users must tap the watch face once to get out of ambient mode. Users must tap watch again to receive vibration feedback. They are allowed to practice this for 1 minute. The users are then asked to perform the functions again and are asked what time was indicated by the watch 5 times. The percentage of users that answer this correctly is recorded.

Test Subjects: 11 engineering students between the ages of 17 and 21.

From the results below, the users had approximately a 93% success rate

in identifying the times. These results were promising as users were able to quickly learn how to use the application with a high success rate.

Table 2: **Usability Test Results**

Accuracy	Frequency
100%	8
80%	2
60%	1
40%	0
20%	0
0%	0

2.2 Performance

Input: Users must tap the watch face twice, with a brief pause in between.

Output: The watch indicates the time through vibration.

How test will be performed: The user will tap the watch face. The watch must respond to the user's tap and exit ambient mode within one second. The watch must respond to the user's second tap and begin vibrating within one second.

This test was performed multiple times while testing the functionality of the smartwatch. Each time, the watch responded immediately to the taps.

3 Comparison to Existing Implementation

The original TouchTime project required users to trace their finger around the watch face until their finger came in contact with either the minute or hour hand. When one of these hands was reached, the smartwatch application had distinct vibration patterns to indicate if the hour hand or minute hand was hit. Additionally, there was a third vibration pattern to indicate that if the hour and minute hands were overlapping. This implementation allowed for a very small number of vibrations to indicate the time, but was

imprecise as it is difficult to tell the time to the exact minute just from knowing the rough position of the hour and minute hands. In addition to this, not everyone is familiar with a traditional watch face.

Our implementation of TouchTime focused on delivering a product that could deliver the time precisely and is easy to use. It consists of long vibrations equal to the number of hours, followed by long vibrations corresponding to the tens digit of the minutes and short vibrations corresponding to the ones digit of the minutes. Each of these vibration patterns has a pause in between to ensure each pattern can be counted distinctly. This allows for a more precise and accessible method of delivering the time through vibration than the original implementation.

4 Unit Testing

Unit testing was performed with JUnit to test the functions responsible for calculating and creating the vibration patterns that will be output by the device. Html output of the test results is available in the TestReport folder. Below is a table summarizing the tests.

Table 3: **Unit Test Results**

	Description	vibCountTest	vibPatternTest
minTime	1:00 (least vibrations)	PASS	PASS
maxTime	12:59 (most vibrations)	PASS	PASS
hour12Test	12:00 (special case)	PASS	PASS

5 Changes Due to Testing

One major change that resulted directly from testing usability was simplifying the vibration pattern that TouchTime used to indicate the time. Initially, we believed the easiest to use solution would be the vibration pattern that minimized the number of vibrations. Our vibration pattern consisted of long, medium, and short vibrations and only required a maximum of 16 vibrations to indicate the worst case time.

The problem with this implementation was that the patterns were too complicated for the user. Our team thought it was intuitive, but that turned out that it was only intuitive because we came up with it. Without user testing, we would not have discovered that this was an issue. Our new vibration pattern has a maximum of 26 vibrations, but it is much easier for the average user to use.

6 Automated Testing

Aside from unit testing, automated testing for Android generally is based around testing the GUI and how the application interacts with a user. Due to the simplicity of our input and output (input: only tap, output: only vibrations), and that the Android Wear emulator cannot emulate vibrations yet, we decided that there wasn't a need to implement an automated testing suite and that our manual testing, particularly with a physical device, would be enough.

7 Trace to Requirements

The functional tests can all be traced back to the functional requirements that define the vibration pattern. These include all the requirements from 2.3 to 2.6, as these outline what vibrations should occur to correspond to each time. Requirement 2.2 outlines how the system responds to touch and is also tested though these tests.

For the non-functional requirements, the usability test can be traced back to the original usability requirements indicated in the SRS. It can be traced back to 3.2.1, which is the ease of use requirement, and to 3.2.2, the ease of learning requirement. This is because the test directly tests how well users were able to learn and use the software.

The performance test can be traced back to 3.3.1, which states that the system must respond to touch within one second. This is exactly what the test case is testing.

8 Trace to Modules

The functional tests can all be traced back to the modules that are responsible for creating the vibration pattern. This includes M2, M3, and M5.

The non-functional tests can be traced to M3 and M5. Usability can be traced to both of these as these modules are what enable the software to be easy to use. Performance can also be traced to M5, as the control module performs the calculations quickly to enhance performance.

9 Code Coverage Metrics

Code coverage metrics were not used in the testing of our application, but the idea behind it was kept in mind when creating our test suite. From what we understood, while code coverage can be useful to see if you're missing large sections of your code in your test, 100% coverage does not guarantee the effectiveness of your tests. Additionally, for our purposes there are many parts of our code that are just designed to create and run the watch face which would not be tested with unit testing and would therefore not be contributing towards code coverage.