SE 3XA3: Test Plan
TouchTime

Team 13, ELM
Matthew Po - 40007877 - pom
Evan Ansell - 1415992 - ansellea
Livia Kelle - 1419709 - kellel

October 28, 2017

Contents

List of Tables

Table 1: **Revision History**

Date	Version	Notes
October 23	1.0	Initial Draft
October 26	1.0.1	Livia created non-functional tests and Gantt chart
October 26	1.0.2	Matt created functional tests
October 27	1.0.4	Evan created PoC tests and unit tests and parts of section 1
October 27	1.0.5	Livia edited parts of section 1 and 2
October 27	1.0.6	Final revision - everything else by everyone

1 General Information

1.1 Purpose

This document will serve as the basis for the verification and validation of our program through testing. During and after the implementation of our software we will use this plan as a reference for which tests we will perform, how the tests will be performed, and the results that we expect for our tests.

1.2 Scope

Tests of the functional and non-functional requirements in the SRS will be completed in this project and outlined in this document. The aim is to perform a variety of tests, from functional vs structural, static vs dynamic, and automatic vs manual. On a smaller level, unit testing will be completed to test the functionality of individual methods. Utilizing many different testing styles has the advantage of testing different aspects of the application.

1.3 Acronyms, Abbreviations, and Symbols

Table 2: **Table of Abbreviations**

Abbreviation	**Definition**
PoC	Proof of Concept
SRS	Software Requirements Specification

Table 3: **Table of Definitions**

Term	**Definition**
TouchTime	Name of the Android smartwatch application.
Functional Test (Black Box)	Analyzes program by probing with various inputs; not concerned with how processing occurs, only results
Structural Test (White Box)	Analyzes program with test data with knowledge of implementation; concerned with how processing occurs
Static Test	Testing that does not involving code execution
Dynamic Test	Testing involving execution of code.
Automatic Test	Tests completed by testing tool such as Junit
Manual Test	Tests conducted by people, by hand

1.4 Overview of Document

This document will be outlining our plans for testing of the TouchTime software. Described below are our approaches to testing, tools to be used, and plans for system and unit tests. TouchTime is not an indredibly complex software so there is a limited amount of functions that can be tested, however a large portion of our testing will involve non-functional testing to make sure that it meets the requirements and expectations of our users.

2 Plan

2.1 Software Description

TouchTime is an Android smart-watch application that enables users to know the time through vibration patterns, for the visually impaired or anyone that would like to be able to tell the time without having to look at the screen. Users interact with the application through tapping the watch face. Tapping on the left side of the screen will result in the vibration patterns corresponding to the hour; the right side of the screen will indicate the minutes.

2.2 Test Team

The software will be tested by Matthew Po, Evan Ansell, and Livia Kelle.

2.3 Automated Testing Approach

The software will be tested using the JUnit testing framework against an emulator that is running the TouchTime software. Each vibration sent by the code to the watch emulator will be logged into logcat in Android Studio. We can use JUnit to assert that the logs are being sent, which is an implication that the code has successfully executed and the watch is vibrating. Note that the test cases are being used to test against the software telling the watch to vibrate, and not used to test the emulators response (i.e. the physical vibration).

2.4 Testing Tools

Laptop running Android Studio with an Android Watch Emulator

2.5 Testing Schedule

See Gantt Chart in the Project Schedule folder. The bright green indicates the tasks related to testing.

3 System Test Description

3.1 Tests for Functional Requirements

3.1.1 Vibration Test on Screen Touch

Vibrate Pattern Left Screen

1. test-fl1

   Type: Functional, Dynamic, Manual

   Initial State: Application installed and launched on Android smartwatch.

   Input: User touches left half of watch emulator.

   Output: Distinct vibration pattern for hour hand. Logcat log to indicate the screen has responded to the touch and vibration has occurred.

How test will be performed: Compare Logcat log pattern to pattern that should have occurred at the given hour. Repeat at various hour times, such as 12, 1, 2, 3, 9, 10.

2. test-fl2

   Type: Functional, Dynamic, Automatic

   Initial State: Application installed and launched on Android smartwatch.

   Input: Simulated touch on left half of watch emulator.

   Output: Distinct vibration pattern for hour hand. Logcat log to indicate the screen has responded to the touch and vibration has occurred.

   How test will be performed: Automatically compare Logcat log pattern to pattern that should have occurred at the given hour. Repeat at every hour for 12 hours.

Vibrate Pattern Right Screen

1. test-fr1

   Type: Functional, Dynamic, Manual

   Initial State: Application installed and launched on Android smartwatch.

   Input: User touches right half of watch emulator.

   Output: Distinct vibration pattern for minute hand. Logcat log to indicate the screen has responded to the touch and vibration has occurred.

   How test will be performed: Compare Logcat log pattern to pattern that should have occurred at the given hour. Repeat at various minutes, such as 00, 01, 09, 15, 20, 29, 45, 55, 59.

2. test-fr2

   Type: Functional, Dynamic, Automatic

Initial State: Application installed and launched on Android smartwatch.

Input: Simulated touch on right half of watch emulator.

Output: Distinct vibration pattern for minute hand. Logcat log to indicate the screen has responded to the touch and vibration has occurred.

How test will be performed: Automatically compare Logcat log pattern to pattern that should have occurred at the given hour. Repeat every minute for one hour.

3.1.2 Abnormal Situations

1. test-fa1

   Type: Functional, Dynamic, Manual

   Initial State: Application installed and launched on Android smartwatch.

   Input: User touches left half of watch emulator and then right half before hours are fully indicated through vibrations.

   Output: Distinct vibration pattern for hour and minute hand. Logcat log to indicate the screen has responded to the left touch and vibration has occurred, but stops upon right touch.

   How test will be performed: Compare Logcat log pattern to pattern that should have occurred at the hour and minute. See where hour pattern was cut off and determine if minutes were correctly indicated at that point.

2. test-fr2

   Type: Functional, Dynamic, Manual

   Initial State: Application installed and launched on Android smartwatch.

   Input: User touches left half of watch emulator at 11:59 and hours are indicated. User then touches the right half after the time changes to 12:00.

Output: Distinct vibration pattern for hour and minute hand. Logcat log to indicate the screen has responded to the left and right touch and vibration has occurred.

How test will be performed: Compare Logcat log pattern to pattern that should have occurred at the hour and minute. The time indicated should be 11:00 due to this timing error, as the user will receive the hour vibration pattern for 11:59 and the minute vibration pattern for 12:00. This is an incorrect but expected behavior that will be corrected in future revision.

3. test-fr3

   Type: Structural, Dynamic, Automatic

   Initial State: Application installed and launched on Android smartwatch.

   Input: Simulated simultaneous touch on both left and right halves of watch emulator.

   Output: Expected: Exception.

   How test will be performed: A computer-driven test. The watch is not able to read out both hour and minute simultaneously, and so an exception should be produced that will be handled. No output should be produced for the user.

3.2 Tests for Nonfunctional Requirements

3.2.1 Usability

1. test-nf-u1

   Type: Functional, Dynamic, Manual

   Initial State: Application installed on Android smartwatch but not launched.

   Input/Condition: Users must tap icon to launch TouchTime.

   Output/Result: Percentage of users capable of launching TouchTime watch face.

How test will be performed: A group of users will tap the smartwatch application to launch it and the percentage that succeed is recorded. This will ensure that an acceptable amount of users are capable of launching the application.

2. test-nf-u2

   Type: Functional, Dynamic, Manual

   Initial State: Application installed and launched on Android smartwatch.

   Input/Condition: Users must tap the left side of the watch face and after the watch finishes vibrating, the user taps the right side.

   Output/Result: Percentage of users capable of correctly tapping left and right sides of the screen at appropriate times.

   How test will be performed: Users must tap the left side of the watch face to get TouchTime to vibrate the hours of the current time. After the hours are indicated, users will tap the right side of the watch face for the minutes. The percentage of users capable of performing these steps is recorded.

3. test-nf-u3

   Type: Functional, Dynamic, Manual

   Initial State: Application installed and launched on Android smartwatch. Users have been taught how the vibration patterns correspond to the time.

   Input/Condition: Users must tap the left side of the watch face and after the watch finishes vibrating, the user taps the right side.

   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 must tap the left side of the watch face to get TouchTime to vibrate the hours of the current time. After the hours are indicated, users will tap the right side of the watch face for the minutes. They are allowed to practice this for 10 minutes. The users are then asked to perform the functions again and are asked what

time was indicated by the watch. The percentage of users that answer this correctly is recorded.

4. test-nf-u4

   Type: Functional, Dynamic, Manual

   Initial State: Application installed and launched on Android smartwatch.

   Input/Condition: Users must tap the left side of the watch face but before the watch finishes vibrating to indicate hours, the user taps the right side.

   Output/Result: Smart watch vibrating to indicate hours until user taps right side at which point, the watch begins indicating the minutes.

   How test will be performed: Users must tap the left side of the watch face to get TouchTime to vibrate the hours of the current time. Users will the prematurely tap the right side of the watch face for the minutes before the hours are indicated. The users are then asked how many minutes were indicated by the watch. The percentage of users that answer this correctly is recorded.

3.2.2 Performance

1. test-nf-p1

   Type: Functional, Dynamic, Manual

   Initial State: Application installed on Android smartwatch but not launched.

   Input/Condition: Users must tap icon to launch TouchTime.

   Output/Result: TouchTime watch face is displayed within 5 seconds.

   How test will be performed: A group of users will tap the smartwatch application to launch it. The time between the application being tapped and the watch face being displayed is recorded and checked if it meets the time requirements.

2. test-nf-p2

   Type: Functional, Dynamic, Manual

   Initial State: Application installed and launched on Android smart watch.

   Input: Users must tap the left side of the watch face.

   Output: The hours of the current time are indicated through vibration.

   How test will be performed: The user will tap the left side of the watch face and identify the hours of the current time indicated by TouchTime. The watch must respond to the user's tap and begin vibrating within one second.

3. test-nf-p3

   Type: Functional, Dynamic, Manual

   Initial State: Application installed and launched on Android smart watch.

   Input: Users must tap the right side of the watch face.

   Output: The minutes of the current time are indicated through vibration.

   How test will be performed: The user will tap the right side of the watch face and identify the minutes of the current time indicated by TouchTime. The watch must respond to the user's tap and begin vibrating within one second.

4. test-nf-p4

   Type: Functional, Dynamic, Manual

   Initial State: Application installed and launched on Android smart watch.

   Input: Users must tap the left side of the watch face.

   Output: The hours of the current time are indicated through vibration.

   How test will be performed: The user will tap the left side of the watch face and identify the hours of the current time indicated by TouchTime.

This must be accurate to the current hour within one second in real-time.

5. test-nf-p5

   Type: Functional, Dynamic, Manual

   Initial State: Application installed and launched on Android smart watch.

   Input: Users must tap the right side of the watch face.

   Output: The minutes of the current time are indicated through vibration.

   How test will be performed: The user will tap the right side of the watch face and identify the minutes of the current time indicated by TouchTime. This must be accurate to the current minute within one second in real-time.

3.3 Traceability Between Test Cases and Requirements

The test cases for the functional requirements relating to vibration patterns can be traced directly to SRS 2.2.4 and 2.2.5, which indicate the vibration patterns for the hours and minutes.

For the non-functional requirements, the usability test cases can be traced back to the original usability requirements indicated in the SRS. Tests test-nf-u1 and test-nf-u2 can be traced back to the ease of use requirement 3.2.1, stating that the application shall be usable by anyone capable of touching the screen of a smart watch. The test case test-nf-u3 can be traced back to the ease of learning requirement 3.2.2, stating that the application shall be easy for anyone capable of using a smart watch to use.

In the case of performance, tests test-nf-p1,test-nf-p2, and test-nf-p3 can be traced to the speed requirements 3.3.1. Test-nf-p4 and test-nf-p5 can be traced to the precision requirement 3.3.4, stating the time must be accurate to within one second.

4 Tests for Proof of Concept

For our Proof of Concept, our objectives are to create an Android Wear application that displays a clock face and responds to distinct touches on the left and right side of the face with a unique vibration pattern for each side. Since it is only a basic application of which we are mostly concerned with the functionality, the early proof of concept tests to be performed are simple manual functional tests.

4.1 Vibration Test on Screen Touch

Vibration Pattern

1. test-pc-v1

   Type: Functional, Dynamic, Manual

   Initial State: Application installed and launched on Android smartwatch. Smartwatch connected to laptop running Android Studio.

   Input: Tap on left side of face ($x < 150$).

   Output: Long vibration pattern from watch. Long vibration pattern output to Logcat log.

   How test will be performed: We will tap the left side of the watch face on either the emulator or physical android watch. The expected result is a vibration pattern: long vibration, pause, long vibration, pause. We will be able to verify that this was the expected vibration pattern by looking at the Logcat logs in our Android Studio window.

2. test-pc-v2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State: Application installed and launched on Android smartwatch. Smartwatch connected to laptop running Android Studio.

   Input: Tap on right side of face ($x \geq 150$).

   Output: Short vibration pattern from watch. Short vibration pattern output to Logcat log.

How test will be performed: We will tap the right side of the watch face on either the emulator or physical android watch. The expected result is a vibration pattern: short vibration, pause, short vibration, pause. We will be able to verify that this was the expected vibration pattern by looking at the Logcat logs in our Android Studio window.

5 Comparison to Existing Implementation

Our TouchTime application will have a different functional implementation than the existing software. As such, direct functional comparisons will be difficult. Despite the different functionalities, our software has the same goal as the existing software, and the nonfunctional requirements (particularly usability and performance) that effect the user's experience will help us compare between the programs.

6 Unit Testing Plan

The Junit framework supplied with Android Studio will be used to complete unit testing. We will use Junit testing in order to quickly and efficiently perform structural tests, particularly on boundary conditions which may be hard to test manually such as boundary pixels for the left and right sides of the watch face.

6.1 Unit testing of internal functions

Unit tests for internal functions will be capable of sending input to the functions through simulated gestures. It may be beneficial for us to create a driver that will perform these gestures. The internal functions that we will focus on will be the functions responsible for storing the time which will be read to the user, along with some simple tests such as making sure our pixel boundaries are set up correctly for registering left/right taps, and tests that make sure the application is correctly determining between a tap, short press, and a long press based on our time boundaries.

6.2 Unit testing of output files

Unit tests for external functions will be capable of reading the output from the Logcat log files. For our methods, we can set certain messages to be sent to the log (such as listing out the vibration pattern sent and the time at which it was recorded). Our unit tests here will be helpful by covering the code for many scenarios and reading the logs to match each output to its expected result. If testing for any time, especially on the boundaries of maximum and minumum time allowed between presses, the vibration patterns should always match with the time being tested.

7 Appendix

7.1 Usability Survey Questions

Some of our tests will be usability tests. Due to the nature of the application, we will have to provide users with instructions on how to tell the time based on the vibration patterns. These usability tests will be helpful to determine if changes need to be made in order to make our application more user-friendly. Some such questions that would follow system tests with groups of users could be:

- Were you able to launch the application on the smartwatch without issue?
- What was the hour that the watch indicated?
- What was the minute that the watch indicated?
- Was the application responsive enough for your needs?