SE 3XA3: Software Requirements Specification
TouchTime

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

October 6, 2017

Contents

1 Project Drivers **1**
- 1.1 The Purpose of the Project . . . 1
- 1.2 The Stakeholders . . . 1
  - 1.2.1 The Client . . . 1
  - 1.2.2 The Customers . . . 1
  - 1.2.3 Other Stakeholders . . . 1
- 1.3 Mandated Constraints . . . 1
- 1.4 Naming Conventions and Terminology . . . 1
- 1.5 Relevant Facts and Assumptions . . . 2

2 Functional Requirements **2**
- 2.1 The Scope of the Work and the Product . . . 2
  - 2.1.1 The Context of the Work . . . 2
  - 2.1.2 Work Partitioning . . . 2
  - 2.1.3 Individual Product Use Cases . . . 3
- 2.2 Functional Requirements . . . 4

3 Non-functional Requirements **4**
- 3.1 Look and Feel Requirements . . . 4
- 3.2 Usability and Humanity Requirements . . . 4
  - 3.2.1 Ease of Use Requirements . . . 4
  - 3.2.2 Ease of Learning Requirements . . . 5
- 3.3 Performance Requirements . . . 5
  - 3.3.1 Speed Requirements . . . 5
  - 3.3.2 Safety Critical Requirement . . . 5
  - 3.3.3 Precision Requirement . . . 5
  - 3.3.4 Capacity Requirement . . . 5
- 3.4 Operational and Environmental Requirements . . . 5
  - 3.4.1 Expected Physical Environment . . . 5
- 3.5 Maintainability and Support Requirements . . . 5
- 3.6 Security Requirements . . . 6
- 3.7 Cultural Requirements . . . 6
- 3.8 Legal Requirements . . . 6
- 3.9 Health and Safety Requirements . . . 6

4 Project Issues **6**
4.1 Open Issues . . . 6
4.2 Off-the-Shelf Solutions . . . 6
4.3 New Problems . . . 6
4.4 Tasks . . . 7
4.5 Risks . . . 7
4.6 Costs . . . 7
4.7 User Documentation and Training . . . 7

List of Tables

1 **Revision History** . . . ii

List of Figures

Table 1: **Revision History**

Date	Version	Notes
Oct 5, 2017	1.0	Initial Draft
Oct 6, 2017	1.1	Added charts, other requirements, final draft
Nov 10, 2017	1.2	Updated functional requirements

This document describes the requirements for the TouchTime project The template for the Software Requirements Specification (SRS) is a subset of the Volere template.

1 Project Drivers

1.1 The Purpose of the Project

The purpose of the TouchTime project is to create an Android Smartwatch application that is able to communicate the current time through vibration patterns for users that are visually impaired or who would like to be able to tell the time without looking.

1.2 The Stakeholders

1.2.1 The Client

The Client for this project will consist of the SFWR 3XA3 TA's and any other potential project reviewers, mainly for the purposes of the SFWR 3XA3 class.

1.2.2 The Customers

The Customers this product will target for will be the visually-impaired and people who want to tell the time surreptitiously.

1.2.3 Other Stakeholders

The other Stakeholders this product will impact will be those who own an Android smart-watch and wish to use our product. We can accomplish this through continuous improvements alongside customer reviews and feedback.

1.3 Mandated Constraints

1.4 Naming Conventions and Terminology

Term	Definition
ELM	The project team name
The application	The TouchTime application

1.5 Relevant Facts and Assumptions

The project operates under the assumptions that:

1). the user of the product is able to physically touch (or virtually through a VM) an android smart-watch
2). the user of the product is able to touch both the hour and minute hand on the watch-face and be able to distinguish the two (i.e. fingers are not too big that it covers the watch face entirely)
3). the device in which the application operates in must be on Android Wear 2.0 or later

2 Functional Requirements

2.1 The Scope of the Work and the Product

ELM will create the software product **TouchTime** that will allow people to tell the time by using the sense of touch alone. Our software will provide users with the vibration feedback in order to tell time, and it will be usable on Android Wear 2.0 and Android OS 7.1.1 (or later) devices

2.1.1 The Context of the Work

The product is intended to be used by people who are visually impaired or who simply want to be able to tell the time without looking at their watch. As such, simplicity and ease of use should be a priority in order to help our users quickly and accurately tell the time.

2.1.2 Work Partitioning

Work Number	Work Name	Work Requirements
1	Watch Face Design	Graphics and Code
2	Vibration Feedback	Code
3	Time and Vibration Calculation	Code
4	State Transitions	Code

2.1.3 Individual Product Use Cases

Since our product is designed to be used equally by people who are visually impaired and those who are not, there will only be one general user type.

To begin to tell the time, a user will touch the face of their android watch with the TouchTime application open. To stop at any time the user simply removes their finger from the face.

Use: **TellHour**

User touches quadrant where the hour hand lies: Device will vibrate once if a.m., twice if p.m. and then pause. Device will then vibrate once for each hour it is past the beginning of the quadrant (so if it is 7pm, it would vibrate once).

Use: **TellMinutes**

Following telling the hour, user touches the 10-minute-section where the minute hand lies: Device will vibrate once then pause. Device will then vibrate once for each minute it is past the beginning of the 10-minute-section (so if it is 7:13, it would vibrate 3 times).

2.2 Functional Requirements

1. The Software shall display a standard analogue-style watch face.
2. The software shall communicate the time through vibrations when the user touches the watch face anywhere
3. The software shall indicate the hours by causing the Android smartwatch to vibrate a corresponding number of short pulses
4. The software shall pause all vibrations for 3 seconds after indicating the hours of the current time before indicating the hours
5. The software shall indicate the minutes by vibrating a series of long, medium and short pulses.
   (a) The tens digit of the current minutes will be indicated by a corresponding number of long pulses
   (b) If the ones digit of the current minutes is 5 or greater, this is indicated by a single medium vibration
   (c) The ones digit is indicated by a series of short pulses. If a medium pulse occurred, the number of short pulses is added to 5 to indicate the minutes.

3 Non-functional Requirements

3.1 Look and Feel Requirements

The appearance of the applications should be that of a clock face displaying the current time.

3.2 Usability and Humanity Requirements

3.2.1 Ease of Use Requirements

This applications shall usable by anyone who is capable of touching a smartwatch and shall be usable by persons with impaired visions.

3.2.2 Ease of Learning Requirements

The user should be capable of opening an application on an Android smart-watch and should be able to touch a screen.

3.3 Performance Requirements

3.3.1 Speed Requirements

The applications should respond within one second of being touched to indicate the time

3.3.2 Safety Critical Requirement

There are no critical safety hazards associated with this project. This is left to the hardware specifications of the physical machine in which the application operates on.

3.3.3 Precision Requirement

The application shall tell the time within the precision of one minute at which the watch face was first tapped

3.3.4 Capacity Requirement

The application shall not exceed the load for the Android smart-watch

3.4 Operational and Environmental Requirements

3.4.1 Expected Physical Environment

The application shall be used on an Android smart-watch that is at least version 8.

3.5 Maintainability and Support Requirements

The application shall be maintained with every Android OS update to ensure compatibility

3.6 Security Requirements

The application should not compromise the security of Smart-watch and should keep all user information confidential.

3.7 Cultural Requirements

The application shall be usable by any culture familiar with a 12-hour clock. The application shall not be offensive to any cultural group.

3.8 Legal Requirements

The application shall comply with all Canadian laws.

3.9 Health and Safety Requirements

The application should not cause the Android smart-watch to dangerously overheat or vibrate in a way that may be damaging.

4 Project Issues

4.1 Open Issues

- Android watch hardware dependencies for Android Studio
- Vibration manipulation dependencies for Android Studio

4.2 Off-the-Shelf Solutions

ELM can make references to previous implementation of project; format code similar to basic software components / backbone of project

4.3 New Problems

- Navigating Android Studio environment
- Keeping up with AOSP programming conventions
- UI integration with back-end code

4.4 Tasks

- familiarize with AOSP programming convention
- familiarize with Android studio environment for Android Wear Application development
- implementation
    - add vibration manipulation dependencies
    - UI
    - integration of back-end with front-end
- proof-of-concept demonstration preparation
- document changes

4.5 Risks

- not completing project on time due to scope; in terms of being able to manipulate how the watch vibrates in response to a screen touch
- hardware lag in terms of vibrational response on Android watch
(There are no major risks to the project development at this point in time)

4.6 Costs

- Cost to develop the software is free (Android Studio, Android development learning tools)
- Cost to test the software is free; Android studio provides us with an emulator for testing applications on a virtual machine
- Cost to hard-test the software on a physical machine depends on the type of machine the user wants to test on (Watch). This can range anywhere from $99 to $499
- ELM recognizes the cost to purchase a physical machine is feasible but inefficient. However, the team is already equipped with a personal smart-watch by one of the group members, allowing us to test on the hardware in real-time.

4.7 User Documentation and Training

Training for android development and Android studio environment in general can be found here