**91.411.201: Software Engineering I**

**Software Requirements Specification (SRS) Template**

Items that are intended to stay in as part of your document are in **bold**; explanatory comments are in *italic* text. Plain text is used where you might insert wording about your project.

The document in this file is an annotated outline for specifying software requirements, adapted from the IEEE Guide to Software Requirements Specifications (Std 830-1993).

Tailor this to your needs, removing explanatory comments as you go along. Where you decide to omit a section, you might keep the header, but insert a comment saying why you omit the data.

**91.411.201: Software Engineering I**

**Team #2**

**Mobile App Development**

**Software Requirements Specification**

**Document**

**Version: (n)** **Date: (mm/dd/yyyy)**

**Table of Contents**

**1. Introduction**

*1.1 Purpose*

*1.2 Scope*

*1.3 Definitions, Acronyms, and Abbreviations*

*1.4 References*

*1.5 Overview*

**2. The Overall Description**

*2.1 Product Perspective*

2.1.1 System Interfaces

2.1.2 Interfaces

2.1.3 Hardware Interfaces

2.1.4 Software Interfaces

2.1.5 Communications Interfaces

2.1.6 Memory Constraints

2.1.7 Operations

2.1.8 Site Adaptation Requirements

*2.2 Product Functions*

*2.3 User Characteristics*

*2.4 Constraints*

*2.5 Assumptions and Dependencies*

*2.6 Apportioning of Requirements*

**3. Specific Requirements**

*3.1 External interfaces*

*3.2 Functions*

*3.3 Performance Requirements*

*3.4 Logical Database Requirements*

*3.5 Design Constraints*

3.5.1 Standards Compliance

*3.6 Software System Attributes*

3.6.1 Reliability

3.6.2 Availability

3.6.3 Security

3.6.4 Maintainability

3.6.5 Portability

*3.7 Organizing the Specific Requirements*

3.7.1 System Mode

3.7.2 User Class

3.7.3 Objects

3.7.4 Feature

3.7.5 Stimulus

3.7.6 Response

3.7.7 Functional Hierarchy

*3.8 Additional Comments*

**4. Change Management Process**

**5. Document Approvals**

**6. Supporting Information**

# 1. Introduction

*The following subsections of the Software Requirements Specifications (SRS) document should provide an overview of the entire SRS. The thing to keep in mind as you write this document is that you are telling what the system must do – so that designers can ultimately build it. Do not use this document for design!!!*

## 1.1 Purpose

The purpose of this document is to specify the requirements of the mobile health monitoring android application. This document is intended for developers and users.

## 1.2 Scope

The health monitoring app is an android application that is designed to provide a mobile platform for the main health monitoring application. The app provides a mobile interface for the main health monitoring website. Aside from the mobile website interface it provides two unique services.

**Fall detection:** The app can detect possible falls and contact an emergency number if necessary.

**Mobile health monitoring:** The app displays relevant information for the user such as BMI, activity level, and current position.

This app allows users with disabilities to live independently but with an added security measure that is responsive and easy to use.

## 1.3 Definitions, Acronyms, and Abbreviations.

*Provide the definitions of all terms, acronyms, and abbreviations required to properly interpret the SRS. This information may be provided by reference to one or more appendices in the SRS or by reference to documents. This information may be provided by reference to an Appendix.*

## 1.4 References

Fall Detection App: Fall detection android app that was reused to make the fall detection algorithms in this app. Made by Ariel Keller and Kevin Ringstaff.

## 1.5 Overview

Section two contains general information of the application such as constraints and product perspectives from the point of view of the user. Section three contains more in depth information and specific requirements useful for developers such as function descriptions and system attributes.

# 2. The Overall Description

*Describe the general factors that affect the product and its requirements. This section does not state specific requirements. Instead, it provides a background for those requirements, which are defined in section 3, and makes them easier to understand*. *In a sense, this section tells the requirements in plain English for the consumption of the customer. Section3 will contain a specification written for the developers.*

## 2.1 Product Perspective

*Put the product into perspective with other related products. If the product is independent and totally self-contained, it should be so stated here. If the SRS defines a product that is a component of a larger system, as frequently occurs, then this subsection relates the requirements of the larger system to functionality of the software and identifies interfaces between that system and the software. If you are building a real system,compare its similarity and differences to other systems in the marketplace. If you are doing a research-oriented project, what related research compares to the system you are planning to build.*

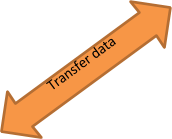
*A block diagram showing the major components of the larger system, interconnections, and external interfaces can be helpful. This is not a design or architecture picture. It is more to provide context, especially if your system will interact with external actors. The system you are building should be shown as a black box. Let the design document present the internals.*

*The following subsections describe how the software operates inside various constraints*.

### 2.1.1 System Interfaces

### This system will consist with three parts: Firmware, Mobile Application, and Cloud storage. The mobile application will be used to analyze the data that get from the firmware then send it to the database, see figure 1.







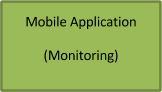


Figure 1 (Block Diagram)

### 2.1.2 Interfaces

Our GUI will be a simple button traversable app that will never go more than two layers deep. This allows for easy navigation by people with disabilities.

### 2.1.3 Hardware Interfaces

This application’s main pieces of hardware to be worked with will be the android phone and the android smart watch.

### 

### 2.1.4 Software Interfaces

The mobile application will need to use the GPS the locatoin of the user for tracking and emergency purposes. It also needs to use the Bluetooth to connect with the firmware to get the heart rate, walking speed and other data of the user.

This application will be using cloud storage. The application will communicate with both the connected firmware and the could program. This will allow for the smoothest transaction between firmware data and cloud storage. The user will get data from the firmware and display it correctly on the android device. while this is happening the adroid will simultaneously update the could wirelessly.

### 2.1.5 Communications Interfaces

### 2.1.6 Memory Constraints

There will be no memory contraints as the only thing this app is doing is temporarily recording data when the user sudddenly moves. This constant monitoring is not memory intensive but will take a lot of power.

### 2.1.7 Operations

This app is not meant to be user intensive and as such will only require a login from the user.

### 2.1.8 Site Adaptation Requirements

## 2.2 Product Functions

With mobile the application, a caretaker will be able to monitor their elderly or disabled. The application has fall detection function that will detect the fall of the user and if something goes wrong, the application will send the warning message to the emergency contact that the user provides, such as family or hospital.

The user can create their profile by adding their personal information and the application will transfer the data to the database. They can manage their profile or view it by clicking at the profile button at the start up page. This profile page will include all the user information, emergency contact and the doctor appointment which the application will remind user about it.

## 2.3 User Characteristics

There are two types of users that interact with the application: patients (disabled) and caretaker (including the third party such as hospital or the patient’s personal doctor). Each of them has a different use for the application

Patients (disabled) can use the application to see their profile and appointment time with the doctor. Also he or she can create their own profile and send it to the cloud system to collect the data.

Caretakers can use this application to monitoring the user and will be notified if and when a fall is detected or the users wanders off somewhere.

## 2.4 Constraints

The mobile application is constrained by the Bluetooth, GPS and internet connection.

BlueTooth: Used to acquire data from smart watch and monitor personal data.

GPS: used to monitor location incase of incident or the user gets lost

Internet: used to send data to and from the cloud when data needs to transferred

## 2.5 Assumptions and Dependencies

One assumption to this application is that it will be used on android mobile phones that have at least 2.2 versions. If the phone doesn’t upgrade to this version, the application will not work and it will cause the application to force close.

The fall detection might not be so accurate since the body of human is not the same size. The big and tall body might fall faster than the smaller one. Also, when the phone drops, the application might think it is the fall of user. (However this issue is being dealt with an I am ok screen and alarm).

Another assumption is that the application is battery draining. The constant monitoring and communication makes the phone run out of batteries very quickly. This is a very hard issue to deal with as this application will naturally be doing a lot of things at once.

## 2.6 Apportioning of Requirements.

The first thing to be created is the fall detection and emergency services. This is the main functionality of the system and therefore will be the first thing finished to allow for maximum testing. With emergency services GPS and data transmitting will all be required. What we will hold back on is the constant stream of data to and from the mobile application as this will be a major power consumer and will not be implemented until needed.

# 3. Specific Requirements

*This section contains all the software requirements at a level of detail sufficient to enable designers to design a system to satisfy those requirements, and testers to test that the system satisfies those requirements. Throughout this section, every stated requirement should be externally perceivable by users, operators, or other external systems. These requirements should include at a minimum a description of every input (stimulus) into the system, every output (response) from the system and all functions performed by the system in response to an input or in support of an output. The following principles apply:*

1. *Specific requirements should be stated with all the characteristics of a good SRS*

* *correct*
* *unambiguous*
* *complete*
* *consistent*
* *ranked for importance and/or stability*
* *verifiable*
* *modifiable*
* *traceable*

1. *Specific requirements should be cross-referenced to earlier documents that relate*
2. *All requirements should be uniquely identifiable (usually via numbering like 3.1.2.3)*
3. *Careful attention should be given to organizing the requirements to maximize readability (Several alternative organizations are given at end of document)*

*Before examining specific ways of organizing the requirements it is helpful to understand the various items that comprise requirements as described in the following subclasses. This section reiterates section 2, but is for developers not the customer. The customer buys in with section 2, the designers use section 3 to design and build the actual application.*

*Remember this is not design. Do not require specific software packages, etc unless the customer specifically requires them. Avoid over-constraining your design. Use proper terminology:*

*The system shall… A required, must have feature*

*The system should… A desired feature, but may be deferred til later*

*The system may… An optional, nice-to-have feature that may never make it to implementation.*

*Each requirement should be uniquely identified for traceability. Usually, they are numbered 3.1, 3.1.1, 3.1.2.1 etc. Each requirement should also be testable. Avoid imprecise statements like, “The system shall be easy to use” Well no kidding, what does that mean? Avoid “motherhood and apple pie” type statements, “The system shall be developed using good software engineering practice”*

*Avoid examples, This is a specification, a designer should be able to read this spec and build the system without bothering the customer again. Don’t say things like, “The system shall accept configuration information such as name and address.” The designer doesn’t know if that is the only two data elements or if there are 200. List every piece of information that is required so the designers can build the right UI and data tables.*

## 

## 3.1 External Interfaces

* Login should be easy and not lag the system. Needed in under 1 second.
* Emergency detection should be called as soon as possible. System warning issued in under 5 seconds
* App should send data to and retrieve data from the cloud and firmware seamlessly. 1 second.

## 3.2 Functions

* System shall be able to detect fall and call emergency services.
* System shall update information routinely.
* System shall receive data through Bluetooth.
* System shall be able to display information for user.
* System shall be able to show user’s location.

## 3.3 Performance Requirements

This app will always need to be using multiple android hardware systems such as accelerometer and gyroscope. The data from these pieces of hardware must be able to coincide with the constant movement of data to and from the firmware and cloud storage.

## 3.4 Logical Database Requirements

## 3.5 Design Constraints

Battery Usage

Bluetooth usage

Accelerometer accuracy

GPS accuracy

Internet Connection

### 3.5.1 Standards Compliance

## 3.6 Software System Attributes

*There are a number of attributes of software that can serve as requirements. It is important that required attributes by specified so that their achievement can be objectively verified. The following items provide a partial list of examples. These are also known as non-functional requirements or quality attributes.*

*These are characteristics the system must possess, but that pervade (or cross-cut) the design. These requirements have to be testable just like the functional requirements. Its easy to start philosophizing here, but keep it specific.*

### 3.6.1 Reliability

This application will run in the background of your android device and will be reliable if your phone is on.

### 3.6.2 Availability

The application will be available to anyone who has an account with medical services.

This application relies on Bluetooth connectivity to receive data. If device is to restart, the application will start back up on restart.

### 3.6.3 Security

Account information and user’s personal information will be secure. No one can access user’s information without account input. Allow maximum number of tries to log into an account.

### 3.6.4 Maintainability

By default Android apps are well organized. It is clear where things are being changed and when. The program has a naming convention that allows for easy recognition of desired tasks.

### 3.6.5 Portability

This code is written in java on the android device. This makes it very easy to transfer to other devices that have the same hardware. With the android code it will be easy to access specific aspects of the app and rewrite wherever they are needed. This makes the code very portable to most any other mobile device

*THE FOLLOWING (3.7) is not really a section, it is talking about how to organize requirements you write in section 3.2. At the end of this template there are a bunch of alternative organizations for section 3.2. Choose the ONE best for the system you are writing the requirements for.*

## 3.7 Organizing the Specific Requirements

*For anything but trivial systems the detailed requirements tend to be extensive. For this reason, it is recommended that careful consideration be given to organizing these in a manner optimal for understanding. There is no one optimal organization for all systems. Different classes of systems lend themselves to different organizations of requirements in section 3. Some of these organizations are described in the following subclasses.*

### 3.7.5 Stimulus

This app is based completely on the fast that if something happens to the user, the app will react. Without stimulus this app will be of no use and thus all the hardware systems must be working corerectly.

## 3.8 Additional Comments

*Whenever a new SRS is contemplated, more than one of the organizational techniques given in 3.7 may be appropriate. In such cases, organize the specific requirements for multiple hierarchies tailored to the specific needs of the system under specification.*

*Three are many notations, methods, and automated support tools available to aid in the documentation of requirements. For the most part, their usefulness is a function of organization. For example, when organizing by mode, finite state machines or state charts may prove helpful; when organizing by object, object-oriented analysis may prove helpful; when organizing by feature, stimulus-response sequences may prove helpful; when organizing by functional hierarchy, data flow diagrams and data dictionaries may prove helpful.*

*In any of the outlines below, those sections called “Functional Requirement i” may be described in native language, in pseudocode, in a system definition language, or in four subsections titled: Introduction, Inputs, Processing, Outputs.*

# Change Management Process

*Identify the change management process to be used to identify, log, evaluate, and update the SRS to reflect changes in project scope and requirements. How are you going to control changes to the requirements. Can the customer just call up and ask for something new? Does your team have to reach consensus? How do changes to requirements get submitted to the team? Formally in writing, email or phone call?*

# Document Approvals

*Identify the approvers of the SRS document. Approver name, signature, and date should be used.*

# Supporting Information

*The supporting information makes the SRS easier to use. It includes:*

* *Table of Contents*
* *Index*
* *Appendices*

*The Appendices are not always considered part of the actual requirements specification and are not always necessary. They may include:*

*(a) Sample I/O formats, descriptions of cost analysis studies, results of user surveys*

*(b) Supporting or background information that can help the readers of the SRS*

*(c) A description of the problems to be solved by the software*

*(d) Special packaging instructions for the code and the media to meet security, export, initial loading, or other requirements*

*When Appendices are included, the SRS should explicitly state whether or not the Appendices are to be considered part of the requirements.*

Tables on the following pages provide alternate ways to structure section 3 on the specific requirements. You should pick the best one of these to organize section 3 requirements.

**Outline for SRS Section 3**

**Organized by mode: Version 1**

3. Specific Requirements

3.1 External interface requirements

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. Functional requirements

3.2.1 Mode 1

3.2.1.1 Functional requirement 1.1

.....

3.2.1.*n* Functional requirement 1.*n*

1. Mode 2

.....

3.2.*m* Mode *m*

3.2.*m*.1 Functional requirement *m*.1

.....

3.2.*m.n* Functional requirement *m.n*

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

**Outline for SRS Section 3**

**Organized by mode: Version 2**

3. Specific Requirements

3.1 Functional Requirements

1. Mode 1

3.1.1.1 External interfaces

3.1.1.1 User interfaces

3.1.1.2 Hardware interfaces

3.1.1.3 Software interfaces

3.1.1.4 Communications interfaces

3.1.1.2 Functional Requirement

3.1.1.2.1 Functional requirement 1

.....

3.1.1.2.*n* Functional requirement *n*

3.1.1.3 Performance

3.1.2 Mode 2

.....

3.1.*m* Mode *m*

1. Design constraints
2. Software system attributes
3. Other requirements

**Outline for SRS Section 3**

**Organized by user class (i.e. different types of users ->System Adminstrators, Managers, Clerks, etc.)**

3. Specific Requirements

3.1 External interface requirements

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. Functional requirements

3.2.1 User class 1

3.2.1.1 Functional requirement 1.1

.....

3.2.1.*n* Functional requirement 1.*n*

1. User class 2

.....

3.2.*m* User class *m*

3.2.*m*.1 Functional requirement *m*.1

.....

3.2.*m.n* Functional requirement *m.n*

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

**Outline for SRS Section 3**

**Organized by object (Good if you did an object-oriented analysis as part of your requirements)**

3 Specific Requirements

3.1 External interface requirements

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. Classes/Objects

3.2.1 Class/Object 1

3.2.1.1 Attributes (direct or inherited)

1. Attribute 1

.....

3.2.1.1.*n* Attribute *n*

1. Functions (services, methods, direct or inherited)

3.2.1.2.1 Functional requirement 1.1

.....

3.2.1.2.*m* Functional requirement 1.*m*

3.2.1.3 Messages (communications received or sent)

3.2.2 Class/Object 2

.....

3.2.*p* Class/Object *p*

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

**Outline for SRS Section 3**

**Organized by feature (Good when there are clearly delimited feature sets.**

3 Specific Requirements

3.1 External interface requirements

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. System features

3.2.1 System Feature 1

3.2.1.1 Introduction/Purpose of feature

3.2.1.2 Stimulus/Response sequence

3.2.1.3 Associated functional requirements

3.2.1.3.1 Functional requirement 1

.....

3.2.1.3.*n* Functional requirement *n*

3.2.2 System Feature 2

.....

3.2.*m* System Feature *m*

.....

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

**Outline for SRS Section 3**

**Organized by stimulus (Good for event driven systems where the events form logical groupings)**

3 Specific Requirements

3.1 External interface requirements

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. Functional requirements

3.2.1 Stimulus 1

3.2.1.1 Functional requirement 1.1

.....

3.2.1.*n* Functional requirement 1.*n*

3.2.2 Stimulus 2

.....

3.2.*m* Stimulus *m*

3.2.*m*.1 Functional requirement *m*.1

.....

3.2.*m.n* Functional requirement *m.n*

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

**Outline for SRS Section 3**

**Organized by response (Good for event driven systems where the responses form logical groupings)**

3 Specific Requirements

3.1 External interface requirements

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. Functional requirements

3.2.1 Response 1

3.2.1.1 Functional requirement 1.1

.....

3.2.1.*n* Functional requirement 1.*n*

3.2.2 Response 2

.....

3.2.*m* Response *m*

3.2.*m*.1 Functional requirement *m*.1

.....

3.2.*m.n* Functional requirement *m.n*

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

**Outline for SRS Section 3**

**Organized by functional hierarchy (Good if you have done structured analysis as part of your design.)**

3 Specific Requirements

3.1 External interface requirements

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. Functional requirements

3.2.1 Information flows

3.2.1.1 Data flow diagram 1

1. Data entities
2. Pertinent processes
3. Topology

3.2.1.2 Data flow diagram 2

1. Data entities
2. Pertinent processes
3. Topology

.....

3.2.1.*n* Data flow diagram *n*

3.2.1.*n*.1 Data entities

3.2.1.*n*.2 Pertinent processes

3.2.1.*n*.3 Topology

3.2.2 Process descriptions

1. Process 1
2. Input data entities
3. Algorithm or formula of process
4. Affected data entities

3.2.2.2 Process 2

3.2.2.2.1 Input data entities

3.2.2.2.2 Algorithm or formula of process

3.2.2.2.3 Affected data entities

.….

3.2.2.*m* Process *m*

3.2.2.*m*.1 Input data entities

3.2.2.*m*.2 Algorithm or formula of process

3.2.2.*m*.3 Affected data entities

3.2.3 Data construct specifications

3.2.3.1 Construct 1

3.2.3.1.1 Record type

3.2.3.1.2 Constituent fields

3.2.3.2 Construct 2

3.2.3.2.1 Record type

3.2.3.2.2 Constituent fields

…..

3.2.3.*p* Construct *p*

3.2.3.*p*.1 Record type

3.2.3.*p*.2 Constituent fields

3.2.4 Data dictionary

3.2.4.1 Data element 1

3.2.4.1.1 Name

3.2.4.1.2 Representation

3.2.4.1.3 Units/Format

3.2.4.1.4 Precision/Accuracy

3.2.4.1.5 Range

3.2.4.2 Data element 2

3.2.4.2.1 Name

3.2.4.2.2 Representation

3.2.4.2.3 Units/Format

3.2.4.2.4 Precision/Accuracy

3.2.4.2.5 Range

…..

3.2.4.*q* Data element *q*

3.2.4.*q*.1 Name

3.2.4.*q*.2 Representation

3.2.4.*q*.3 Units/Format

3.2.4.*q*.4 Precision/Accuracy

3.2.4.*q*.5 Range

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

**Outline for SRS Section 3**

**Showing multiple organizations (Can’t decide? Then glob it all together)**

3 Specific Requirements

3.1 External interface requirements

1. User interfaces
2. Hardware interfaces
3. Software interfaces
4. Communications interfaces
5. Functional requirements

3.2.1 User class 1

3.2.1.1 Feature 1.1

3.2.1.1.1 Introduction/Purpose of feature

3.2.1.1.2 Stimulus/Response sequence

3.2.1.1.3 Associated functional requirements

3.2.1.2 Feature 1.2

3.2.1.2.1 Introduction/Purpose of feature

3.2.1.2.2 Stimulus/Response sequence

3.2.1.2.3 Associated functional requirements

…..

3.2.1.*m* Feature 1.*m*

3.2.1.*m*.1 Introduction/Purpose of feature

3.2.1.*m*.2 Stimulus/Response sequence

3.2.1.*m*.3 Associated functional requirements

3.2.2 User class 2

.....

3.2.*n* User class *n*

.....

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

**Outline for SRS Section 3**

**Organized by Use Case (Good when following UML development)**

3. Specific Requirements

3.1 External Actor Descriptions

3.1.1 Human Actors

3.1.2 Hardware Actors

3.1.3 Software System Actors

3.2 Use Case Descriptions

3.2.1 Use Case 1

3.2.2 Use Case 2

3.2.n Use Case n

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements