SE3352a: Software Requirements and Analysis

BRiGADE

Modern Software Requirements Specification

For Self Start System

Version 1.0



Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 02/11/17 | 1.0 | Started shared group document | Everyone |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 4

1.1 Purpose 4

1.2 Scope 4

1.3 Definitions, Acronyms and Abbreviations 4

1.4 References 4

1.5 Overview 4

2. Overall Description 4

2.1 Use-Case Model Survey 4

2.1.1 Introduction 4

2.1.2 Survey Description 4

2.1.3 Use-Case Model Hierarchy 5

2.1.4 Diagrams of the Use-Case Model 5

2.2 Assumptions and Dependencies 5

3. Requirements 5

3.1 Use-Case Specifications 5

3.2 Functionality 5

3.2.1 <Functional Requirement One> 5

3.3 Usability 6

3.3.1 <Usability Requirement One> 6

3.4 Reliability 6

3.5 Performance 6

3.5.1 <Performance Requirement One> 6

3.6 Supportability 6

3.6.1 <Supportability Requirement One> 7

3.7 Design Constraints 7

3.7.1 <Design Constraint One> 7

3.8 Online User Documentation and Help System Requirements 7

3.9 Purchased Components 7

3.10 Interfaces 7

3.10.1 User Interfaces 7

3.10.2 Hardware Interfaces 7

3.10.3 Software Interfaces 7

3.10.4 Communications Interfaces 7

3.11 Licensing Requirements 7

3.12 Legal, Copyright and Other Notices 7

3.13 Applicable Standards 7

Modern Software Requirements Specification

# Introduction

## Purpose

The purpose of this Modern SRS is to help in the creation and design of a software system for Marcotte Physiotherapy. This system, titled Self Start, will bring the treatment of the physiotherapist into a patient’s home through an online application, and it will provide clinicians tools to improve the traditional approaches of the patient’s progression assessment. The software must be a web-based application that is accessible 24/7. It must have security features such as authentication to protect the privacy of patients and maintain confidentiality. In addition, the software should be designed in English, and allow new content, such as multimedia, to be uploaded through administrator portals. Finally, the interface must be user-friendly, and a secure database will be kept that contains patient information.

## Scope

[A brief description of the software application that the Modern SRS applies to; the feature or other subsystem grouping; what Use Case model(s) it is associated with, and anything else that is affected or influenced by this document.]

## Definitions, Acronyms and Abbreviations

|  |  |
| --- | --- |
| **Term** | **Definition** |
|  |  |

## References

[This subsection should provide a complete list of all documents referenced elsewhere in the Modern SRS. Each document should be identified by title, report number (if applicable), date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]

## Overview

Section two, known as Overall Description, conveys the required functionality that the final software must have. This section will provide the foundation for understanding the in-depth descriptions that section three, which is Requirements, will contain. Section two will contain use-case descriptions and the use-case model, while section three will provide use-case specifications along with system functions and requirements.

# Overall Description

[This section of the Modern SRS should 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 detail in section 3, and makes them easier to understand. Include such items as product perspective, product functions, user characteristics, constraints, assumptions and dependencies, and requirements subsets.]

## Use-Case Model Survey

[This section contains an overview of the use-case model or the subset of the use-case model that is applicable for this subsystem or feature.  This includes a list of names and brief descriptions of all use cases and actors, along with applicable diagrams and relationships. This section describes the use-case model comprehensively, in terms of how the model is structured into packages and what use cases and actors there are in the model. If you are using packages, the document shows the model structure hierarchically.]

### Introduction

[Introduction to the use-case model.]

### Survey Description

[Survey description of the use-case model.]

### Use-Case Model Hierarchy

[This section presents the use-case packages hierarchically, explains the dependencies among them, and shows the content of each package recursively. If the model has several levels of packages, those at the top-level are presented first. The packages within these are presented next, and so on, all the way down to the packages at the bottom of the hierarchy. For each package include:

* The Name.
* A Brief Description explaining the package's function and role in the system. The description must be understandable to any developer who wants to use the package.
* A list of the use cases owned by the package, including the name and brief description of each use case.
* A list of actors owned by the package, including the name and brief description of each actor.
* A list of relationships owned by the package, including the name and brief description of each relationship.
* A list of the packages directly owned by the package, with each package presented in the same hierarchical manner as above]

### Diagrams of the Use-Case Model

[Diagrams, primarily use-case diagrams, of the entire use-case model are included here.]

## Assumptions and Dependencies

[This section describes any key technical feasibility, subsystem or component availability, or other project related assumptions on which the viability of the software described by this Modern SRS may be based.]

# Requirements

[This section of the Modern SRS should contain all the software requirements to 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.   When using use-case modeling, the majority of these requirements are captured in the use cases.]

## Use-Case Specifications

[In use-case modeling, the use cases often define the majority of the functional requirements of the system, along with some non-functional requirements. For each use case in the above use-case model, or subset thereof, enclose the use-case specification here. If you have documented use cases in an separate document, cross reference to all applicable external use-case specifications in this section. Make sure that each requirement is clearly labeled.]

## Functionality

[This section describes the functional requirements of the system for those requirements that are expressed in the natural language style. For many applications, this may constitute the bulk of the Modern SRS Package and thought should be given to the organization of this section. This section is typically organized by feature, but alternative organization methods, for example organization by user, or organization by subsystem may also be appropriate. Functional requirements may include: **feature sets, capabilities and security**.

Where application development tools (requirements tools, modeling tools, etc) are employed to capture the functionality, this section document will refer to the availability of that data and indicate the location and name of the tool which is used to capture the data.]

### <Functional Requirement One>

[The requirement description.]

## Usability

[This section should include all of those requirements that affect usability. Examples:

1. Specify the required training time for a normal users and power users to become productive at particular operations.
2. Specify measurable task times for typical tasks, or
3. Base usability requirements of the new system on other systems that the users know and like.
4. Specify requirements to conform to common usability standards – e.g., IBM’s CUA standards, or the GUI standards published by Microsoft for Windows 95.]

### <Usability Requirement One>

The requirement description.

## Reliability

[Requirements for reliability of the system should be specified here. Suggestions:

1. Availability – specify % of time available ( xx.xx%), hours of use, maintenance access, degraded mode operations etc.
2. Mean Time Between Failures (MTBF) – this is usually specified in hours, but it could also be specified in terms of days, months, or years.
3. Mean Time To Repair (MTTR) – how long is the system allowed to be out of operation after it has failed?
4. Accuracy – specify precision (resolution) and accuracy (by some known standard) that is required in the systems output.
5. Maximum bugs or defect rate – usually expressed in terms of bugs/KLOC (thousands of lines of code), or bugs per function-point.
6. Bugs or defect rate – categorized in terms of minor, significant, and critical bugs: the requirement(s) must define what is meant by a “critical” bug (e.g., complete loss of data, complete inability to use certain parts of the functionality of the system).]

#### *<Reliability Requirement One*>

[The requirement description.]

## Performance

[The performance characteristics of the system should be outlined in this section. Include specific response times. Where applicable, reference related Use Cases by name.

1. Response time for a transaction (average, maximum)
2. Throughput (e.g., transactions per second)
3. Capacity (e.g., the number of customers or transactions the system can accommodate)
4. Degradation modes (what is the acceptable mode of operation when the system has been degraded in some manner)
5. Resource utilization: memory, disk, communications, etc.]

### <Performance Requirement One>

[The requirement description.]

## Supportability

[This section indicates any requirements that will enhance the supportability or maintainability of the system being built, including coding standards, naming conventions, class libraries, maintenance access, maintenance utilities.]

### <Supportability Requirement One>

[The requirement description.]

## Design Constraints

[This section should indicate any design constraints on the system being built. Design constraints represent design decisions that have been mandated and must be adhered to. Examples include software languages, software process requirements, prescribed use of developmental tools, architectural and design constraints, purchased components, class libraries, etc.]

### <Design Constraint One>

[The requirement description.]

## Online User Documentation and Help System Requirements

[Describes the requirements, if any, for on-line user documentation, help systems, help about notices, etc.]

## Purchased Components

[This section describes any purchased components to be used with the system, any applicable licensing or usage restrictions, and any associated compatibility/interoperability or interface standards.]

## Interfaces

[This section defines the interfaces that must be supported by the application. It should contain adequate specificity, protocols, ports and logical addresses, etc, so that the software can be developed and verified against the interface requirements.]

### User Interfaces

[Describe the user interfaces that are to be implemented by the software.]

### Hardware Interfaces

[This section defines any hardware interfaces that are to be supported by the software, including logical structure, physical addresses, expected behavior, etc.]

### Software Interfaces

[This section describes software interfaces to other components of the software system. These may be purchased components, components reused from another application, or components being developed for subsystems outside of the scope of this SRS, but with which this software application must interact.]

### Communications Interfaces

[Describe any communications interfaces to other systems or devices such as local area networks, remote serial devices, etc.]

## Licensing Requirements

[Defines any licensing enforcement requirements or other usage restriction requirements that are to be exhibited by the software.]

## Legal, Copyright and Other Notices

[This section describes any necessary legal disclaimers, warranties, copyright notices, patent notice, word mark, trademark, or logo compliance issues for the software.]

## Applicable Standards

[This section describes by reference any applicable standards, (and the specific sections of any such standards that apply to the system being described). For example, this could include legal, quality and regulatory standards, industry standards for usability, interoperability, internationalization, operating system compliance, etc.]