<Team Name>

Modern Software Requirements Specification

For Self Start System>

Version <1.0>

[Note: Text enclosed in square brackets and displayed in blue italics (style=InfoBlue) is included to provide guidance to the author and should be deleted before publishing the document. A paragraph entered following this style will automatically be set to normal (style=Body Text).]

[To customize automatic fields (which display a gray background when selected), select File>Properties and replace the Title, Subject and Company fields with the appropriate information for this document. After closing the dialog, automatic fields may be updated throughout the document by selecting Edit>Select All (or Ctrl-A) and pressing F9, or simply click on the field and press F9. This must be done separately for Headers and Footers. Alt-F9 will toggle between displaying the field names and the field contents. See Word help for more information on working with fields.]

[Note: The Software Requirements Specification (SRS) captures the complete software requirements for the system, or a portion of the system.  The Modern SRS is a typical SRS outline for a project **using use-case modeling**. This artifact consists of a package containing use cases of the use-case model and applicable Supplementary Specifications and other supporting information. For a template of an SRS **not** using use-case modeling, which captures all requirements in a single document, with applicable sections inserted from the Supplementary Specifications (which would no longer be needed), see[\\program](file:///\\program) \program files\Rational\ RequisitePro\Outlines\ rup\_srs.dot.]

Many different arrangements of an SRS are possible. Refer to [IEEE93] for further elaboration of these explanations, as well as other options for SRS organization.]

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <dd/mmm/yy> | <x.x> | <details> | <name> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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

[The introduction of the Modern SRS should provide an overview of the entire Modern SRS. It should include the purpose, scope, definitions, acronyms, abbreviations, references and overview of the Modern SRS.]

## Purpose

[Specify the purpose of this Modern SRS. The Modern SRS should fully describe the external behavior of the application or subsystem identified. It also describes nonfunctional requirements, design constraints and other factors necessary to provide a complete and comprehensive description of the requirements for the software.]

## 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

[This subsection should provide the definitions of all terms, acronyms, and abbreviations required to interpret properly the Modern SRS.  This information may be provided by reference to the project Glossary.]

## 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

[This subsection should describe what the rest of the Modern SRS contains and explain how the Modern SRS is organized.]

# 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

The SOLVERE team will be documenting the provided code to enable basic troubleshooting capabilities as offline support. To provide online support, every member of SOLVERE will provide their Western email for contact in case of major unexpected errors. Every member of SOLVERE is also expected to return to Western next year, and as such one year of in-person support can be provided.

## Purchased Components

All parts of the product provided are proprietary software of SOLVERE. To the best of the SOLVERE team’s understanding, there will be no need for any third-party APIs. As a result, there will be no problems relating to third-party legal constraints or purchasing the right to use such software.

## Interfaces

### User Interfaces

User accounts, physiotherapist accounts, and admin accounts will all have different privileges and subsequently different views per those privileges. This will allow for easy and streamlined functionality even as the functionality differs based on the user’s level of permissions.

### Hardware Interfaces

The product will require a hardware interface to host the server that will be running. Communication to the server, which will run on Ember.js will take place over HTTPS connections on port 80, though it is possible a back-end server with its own port will be utilized.

### Software Interfaces

To the best of the SOLVERE team’s knowledge, there is no external software to interface with, nor are any components of the system being developed for use in other projects.

### Communications Interfaces

As touched on above in 3.10.2, Hardware Interfaces, the server will need to communicate with various actors. The client machine, used by a user, physiotherapist, or admin, will connect to the server over HTTPS to ensure secure and efficient communication.

## Licensing Requirements

The final product is the proprietary creation of SOLVERE, and it will be sold to Marcotte Physiotherapy Clinic. For the product to be used outside of that scope, written consent from SOLVERE must be obtained.

## Legal, Copyright and Other Notices

All patient information, including but not exclusive to treatment plans, assessment tests, and user progress are private and cannot be shared to people who are not Marcotte Physiotherapy Clinic staff without the consent of the patient in question.

## Applicable Standards

The product that SOLVERE provides will be held to certain standards, outlined below:

The product can be held to certain standards regarding data integrity. To remove risk of foreign agents compromising the accuracy of the data, the database will have security measures limiting access to the data to easily controlled paths. The product will also have precautions against corrupted data to ensure accuracy of database contents.

The product can be held to a standard of compatibility with any browser. As it will be built with the Node.js framework, any browser supporting HTTP and JavaScript will be able to access and use the product. Cookies are also expected to be used to help streamline the user experience, and while the product will be usable without cookies it is recommended that they be available.