<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

### For any user visiting the site

1. The system should provide a description of on-line patient services to all users visiting the site.

# The system should provide an interface for all user types to login by authenticating their username and encrypted password. The system should configure access conditions based on the type of user account and, for patients, which user is logged in.

1. The system should allow prospective patients who would like to contact the clinic for the first time the option to request an appointment or fill out and submit an online form explaining their injury.

### For patients

1. The system should allow the patient to create an account. This account will maintain basic personal information for the patients such as name (family name, given name), gender, date of birth, address (including city, region, and postal code), telephone number, health card number, marital status, occupation, and others.
2. The system should allow the patient to change his/her password, or request the system administrator to reset the forgotten password.
3. The system should allow the patient to book appointments with the physiotherapist online and receive immediate confirmation.
4. The system should provide an introductory form to be filled out on the application with the goal of help patients describe their injury in detail. This will be achieved through different levels and types of questions, including the ability to upload static photos of the patient.
5. The patient should be allowed to explore and view their treatment exercises (through an easy to follow menu, text-based guidance and/or simple animations).
6. The patient should be able to complete their self-assessment test questions through an (easy to use and simple) online form. The system should record these assessments and store them with reference to the patient and treatment plan so that the data can be used to assess the results of the treatment.
7. The patient should be able to perform payment for their services online through the system.

### For physiotherapists

1. The physiotherapist must be enabled to create clinic-defined standard rehabilitation plans (for common injuries) that include a set of exercises and self-assessment activities.
2. The physiotherapist must be able to create, and save a single exercise. The system should maintain the following information for each exercise: a unique identification code, name, description, author name, objectives, action steps, location, frequency, duration, target date, and a link to multimedia (via URL).
3. The physiotherapist must be able to create and save a single self-assessment test. The system should maintain the following information for each assessment test: a unique identification code, name, description, author name, assessment tools (e.g., rating/ranking questions, multiple choice questions), and an assessment rubric.
4. The physiotherapist should be able to assign a set of exercises and a set of self-assessment tests to be able to create a custom rehabilitation plan. The system should maintain the following information about each rehabilitation plan: a unique identification goal, name, description, author name, overall rehabilitation goal, an ordered list of exercises, a timeframe to complete the plan, and assessment tests.
5. The physiotherapist should be allowed to assign one or more rehabilitation plans to one or more patients.
6. The physiotherapist should be able to generate and print/send a written report summarizing the examination findings and treatment plan to the patient.
7. Based on the assessment results and pre-defined rubrics, the physiotherapist should be able to generate a data analysis that reflects the impact of the rehabilitation plan on the patient.
8. The physiotherapist should be able to decide whether to assign a follow up session with the patient (to reassess the case and provide an adjusted treatment) or to close the case.
9. The physiotherapist should be allowed to display and print a summary report for each patient. This report should include: patient personal information, the diagnosed case, the assigned treatments, calendar of appointments, invoice payments, and the final outcomes.

### For admin

1. The system should be shipped with an administrator account, in addition to two users’ roles types: Patient accounts (and corresponding functionalities) and Physiotherapist accounts (and corresponding functionalities).
2. The system should allow the administrator to customize (add, change, or delete) the questions of the introduction form without having to change the system’s code.
3. The administrator must be allowed to create user accounts for patients and/or physiotherapists.
4. The administrator should be able to reset a patient’s password upon verification of their identity and inform them of their new password.
5. The administrator should be allowed to disable or enable existing Patient and Physiotherapist accounts.

## Usability

### Required Training Time

Physiotherapist and Admin users should be able to become productive at all operations of the system after 5 hours of training. This training can be executed in groups. If one on one training is preferred, productive use of all operations can be achieved in only 4 hours. Patient users should not require any training beyond looking up simple online user documentation that will be provided on the site.

### User Interface

All sections of the system, for all user views/configurations, will have an interface that is intuitive and simple. To ensure this is achieved, IBM Common User Access (CUA) standard will be followed. These standards will ensure our application conforms to many other interfaces that our users have most likely used before in the past and are comfortable with. For areas of the application that are slightly more complex (i.e., create a rehabilitation plan from a set of exercises and self-assessment tests or uploading a series of pictures), the field-level help standard will be used to guide the user.

### Single Page Application

Increasingly common in web applications is the use of a single, dynamic page. For each core functionality, a single, dynamic page will be used, with a menu bar above that allows access to other features and areas of the system. For example, once a Patient is logged in, a single, dynamic page will be used to display their exercises, and another will be used for their self-assessment tests.

### Measurable Times for Typical Tasks

For web applications, it is often easier to use “clicks” as a proxy for the time it takes to navigate to a specific function or complete a specific task. The expected number of clicks for some typical tasks are:

|  |  |  |
| --- | --- | --- |
| User | Task | Number of Clicks |
| All users | Login to account | 2 |
| Patient | Navigate to rehabilitation plan | 2 |
| Physiotherapist | Assign rehabilitation plan to user | 4 |
| Admin | Reset a password | 3 |

## Reliability

#### *Availability*

The application should be available to Patients, Physiotherapists, and Admin 99% of the time. 24/7 availability is the goal with the exception of scheduled monthly maintenance at off-peak times. Scheduled maintenance or ad-hoc maintenance must be posted 48 hours in advance and should not take longer than 6 hours to complete.

#### *Mean Time Between Failures (MTBF)*

The MTBF should be no shorter than 24 hours. Upon failure, maintenance team should be notified via email alerts.

#### *Mean Time to Repair (MTTR)*

The MTTR should be no longer than 10 hours. This includes the time to determine the source of failure and efforts to repair the failure.

#### *Accuracy*

The system does not perform any complex calculations or time-sensitive actions that require exceptional level of accuracy. However, the system should be consistently accurate in its core functionalities, such as: displaying the correct user views, ensuring patients are assigned the correct rehabilitation plan, recording and storing self-assessment responses, and creating treatment analysis reports. There should be no system errors in these types of activities, so 100% system accuracy is required. Note that in some instances user error might affect accuracy (e.g., assign incorrect rehabilitation plan to the patient).

#### *Maximum Bugs or Defect Rate*

The system should have a maximum of 4 bugs/KLOC. These bugs should be minor and not affect the core functionality of the system – they should only affect rare “edge cases” of operation. No significant or critical bugs should exist in the system upon deployment or after maintenance work.

#### *Bugs or Defect Rate*

1. Minor Bugs: Affect “edge cases” and are noticed in only rare circumstances. Will not affect the day-to-day functioning of the system.
2. Significant Bugs: Issues that affect the logic of the system. For example, patients are viewing incorrect rehabilitation plans, self-assessment tests are unable to be submitted, and new users are unable to sign up. These bugs must be fixed immediately upon notice as they affect the functionality of the system. These bugs must not exist upon deployment or ongoing maintenance. Once these bugs are fixed, the system can return to a recent state and system functionality can resume with minimal issues.
3. Critical Bugs: Issues that affect the integrity of the entire system. This could include loss of data, financial information leaks, and complete inability to use parts of the system. These bugs must be prevented at all costs, and the system should be designed to mitigate against critical bugs. For example, extensive security testing on the online payments function is necessary and the database that stores exercises, assessment tests, rehabilitation plans, and customer data should be backed up periodically.

## Performance

### Response Time

All functions performed by the user require little to no processing (primarily “fetch and display” operations), and so the average and maximum response times should be less than 0.5 seconds and 1 second, respectively. The only exceptions are for the online payment processing and resetting an account password. The average and maximum response times for confirming payment should be 3 seconds and 8 seconds, respectively. Resetting an account password requires action to be completed by the admin user, thus the system is unable to guarantee a response time as it is dependent on an external actor.

Almost all functions performed by the admin and physiotherapist are slightly more complex as they make changes to the state of the system. Thus, the average and maximum response times for the functions should be 1.5 seconds 3 seconds, respectively. The only exception is for the “Assess Test Results” use case, where information needs to be aggregated from the database and analyzed. The average and maximum response times for pulling this report will be 5 seconds and 10 seconds, respectively.

### Throughput and Capacity

The system should be able to serve up to 200 different active patients without degradation to its performance. There will be less than 200 transactions per second of the server for any function. The system should support all these transactions without exceeding the maximum response times state above.

### Degradation Mode

If performance is degraded in some way (e.g., malicious attack, run-time error, significant or critical bug), the system will take action to resolve it as soon as possible. Upon realization of the system issue, the administrator will be notified via email. The system will then transition into read-only mode – the state of the system will not be able to be modified until the administrator has taken action to reset the system state to a time prior to the event. During read-only mode, patients will be able to follow their exercise plans, however assessment responses will not be able to be recorded and will have to be completed at a later time.

## Supportability

### Modular Programming

System code will be developed modularly. The system will follow strict folder guidelines, that group pages, features, and assets to together in a logical way. For example, all system features and assets used on a specific page of the site will be required to be grouped together. An effort will be made to separate “structure” (e.g. HTML), “presentation” (e.g. CSS), and “interaction” (e.g. JavaScript) elements of code where possible.

### Cross-Browser and Cross-Hardware Support

We are unable to predict which browser and hardware will be used by clients when accessing the system. For this reason, the system must be able to run on all popular browsers (Chrome, Edge, Firefox, Safari, and Opera) on any device. The system must maintain the integrity of its user interface when displayed on a tablet or mobile device.

### Code Readability

Non-trivial code must be well-documented and easily understood by someone with a limited knowledge of Ember.js. System maintenance past April 30, 2018 will have to be carried out by external parties. Thus, a high effort must be made to ensure that new developers will quickly understand how the system functions.

### System Maintenance

The system must be designed to be low maintenance, as access to technical support is limited upon deployment.

## Design Constraints

### Security

The system must use the Hypertext Transfer Protocol Secure (HTTPS) as its internet communications protocol to generate secure encryption keys between the web server and the clients’ browsers. The system will rely on a certified security certificate to enable the use of HTTPS.

### Hosting

The system must be installed on and hosted by the Marcotte Physiotherapy Clinic central server. The Node.js framework will be used, and the server must be capable to support it.

### Web Application Framework

Ember.js will be used as the framework for the system. This framework will aid in achieving the Supportability requirements listed above.

## Online User Documentation and Help System Requirements

Ademidun Hart & Company will provide documentation for simple troubleshooting activities to be carried out by admin and physiotherapists at Marcotte Physiotherapy. If admin and physiotherapists require additional help for advanced troubleshooting, they will be provided with the name, email address, and phone number of a primary, secondary, and tertiary contact at Ademidun Hart & Company. In-person support will continue until April 30, 2018.

All support documentation for patient users will be integrated into the user interface of the system using the field-level help feature of the interface as defined by IBM CUA guidelines. For complex support, we recommend the patient user to contact admin at Marcotte Physiotherapy.

## Purchased Components

All components of the system will be created by Ademidun Hart & Company with the exceptions of:

1. Existing MyHealth System – some functionality of Self Start may integrate with the existing system in the future. Ademidun Hart & Company will not be purchasing or licensing this system. Nor do we guarantee the compliance of Self Start with Ontario’s Personal Health Information Protection Act (PHIPA).
2. Exercise Animations/Videos/Descriptions – these will not be purchased or supplied Ademidun Hart & Company for this system. Self Start will allow the user to integrate these as necessary when creating an exercise, but are not responsible for the content and can in no way be held liable for patient issues arising as a result of following the prescribed exercises on the Self Start system.

Thus, all features in Self Start are proprietary property of Ademidun Hart & Company. Third-party APIs relating to online payments may be required for the Self Start system. If it is recognized that a paid third-party API or service is necessary to implement this feature, the cost will be passed on to Marcotte Physiotherapy.

## 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.]