SE3352a: Software Requirements and Analysis

Brigade

Modern Software Requirements Specification

For Self Start System

Version 2.0



Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 02/11/17 | 1.0 | Started shared group document | Everyone |
| 17/11/17 | 1.1 | Completed section 1 | Brandon |
| 05/11/17 | 1.0 | Complete UC | Everyone |
| 07/11/17 | 1.0 | Complete section 2 | Seung |
| 09/11/17 | 1.0 | Complete 3.1 | Mustafa |
| 09/11/17 | 1.1 | Complete 3.2 | Seung |
| 10/11/17 | 1.0 | Complete 3.3-3.5 | Ramzi |
| 11/11/17 | 1.0 | Complete 3.6 - 3.9 | Omar |
| 12/11/17 | 1.0 | Complete 3.10 -3.12 | Yousep |
| 17/11/17 | 1.9 | Revised and formatted | Brandon |
| 17/11/17 | 2.0 | Formatted finish version | 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 5

1.5 Overview 5

2. Overall Description 5

2.1 Use-Case Model Survey 5

2.1.1 Introduction 5

2.1.2 Survey Description 5

2.1.3 Use-Case Model Hierarchy 5

2.1.4 Diagrams of the Use-Case Model 7

2.2 Assumptions and Dependencies 7

3. Requirements 8

3.1 Use-Case Specifications 8

3.2 Functionality 14

3.3 Usability 14

3.4 Reliability 15

3.5 Performance 16

3.6 Supportability 16

3.6.1 <Supportability Requirement One> 16

3.6.2 < Supportability Requirement Two> 16

3.7 Design Constraints 16

3.7.1 <Design Constraint One> 16

3.8 Online User Documentation and Help System Requirements 17

3.9 Purchased Components 17

3.10 Interfaces 17

3.10.1 User Interfaces 17

3.10.2 Hardware Interfaces 18

3.10.3 Software Interfaces 18

3.10.4 Communications Interfaces 18

3.11 Licensing Requirements 18

3.12 Legal, Copyright and Other Notices 18

3.13 Applicable Standards 18

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 physiotherapist 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 to be uploaded. Finally, the interface must be user-friendly, and a secure database will be kept that contains patient information.

## Scope

This Self Start software will be an online website that is integrated with a database containing Marcotte Physiotherapy patient information. The administrator of the website will be able to manage the database. Users can sign themselves up for Self Start. Self Start will improve the experience of physiotherapy for the patients of Marcotte Physiotherapy. Having an online website that allows for patients to get therapy at home gives Marcotte Physiotherapy a competitive edge over other Physiotherapists clinics.

Key functionalities in the system will be developed that allow for the database, system, administrators, and patients to interact. Using the features of Self Start, patients will be able to go through their entire physiotherapy treatment in an easy to access manner accompanied by a user-friendly interface.

## Definitions, Acronyms and Abbreviations

|  |  |
| --- | --- |
| **Term** | **Definition** |
| SRS | A software requirements specification (SRS) is a description of a software system to be developed. It lays out functional and non-functional requirements, and may include a set of use cases that describe user interactions that the software must provide. |
| UC | A Use Case (UC) is a list of actions or event steps typically defining the interactions between a role (known in the Unified Modeling Language as an actor) and a system to achieve a goal. The actor can be a human or other external system. |
| Database | Software that stores information. The database in Self Start will store the information of patients that use Self Start. |
| Administrator | A system user that has additional privileges. |
| MTBF | Mean Time Between Failures. The average predicted elapsed time between failures of the system. |
| KLOC | Thousand lines of code. |

## References

**Title:** Assignment 1: Developing the Software Requirements Specifications SRS

**Reference:** Ouda, A. “Assignment 1: Developing the Software Requirements Specifications SRS.” SE 3352 Software Requirements and Analysis. Western University, 1 Nov. 2017. Web. 2 Nov. 2017.

<https://owl.uwo.ca/access/content/attachment/672f3446-c1d2-46a8-9426-90707ad34952/Assignments/8ebc5171-3538-47ba-adeb-85d4791f515c/SE3352a_assignment1%20_2017_.pdf>

**Title**: Building Real Software

**Reference**: Bird, J. “Building Real Software” Blogspot. 24 Aug. 2011. Web. 11 Nov. 2017

<http://swreflections.blogspot.ca/2011/08/bugs-and-numbers-how-many-bugs-do-you.html>

## 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

## Use-Case Model Survey

### Introduction

To describe the functionality and implementation, UCs are used to portray the functions of the system. They show the interaction between the system and external users, called actors. UC diagrams simulate basic functionality. The UCs in this project are essential to achieve the end goal of an object-oriented system to serve Marcotte Physiotherapy. Each UC in this model is going to serve either the physiotherapist, administrator, or the patient.

### Survey Description

Self Start is a stand-alone, web-based application that extends the functionality of Marcotte Physiotherapy’s website. The purpose of the system is to automate the booking process to allow for online treatment access. The system is maintained by the administrator and allows currently employed physiotherapists to add rehabilitation plans, which contain assessment tests and treatment plans.

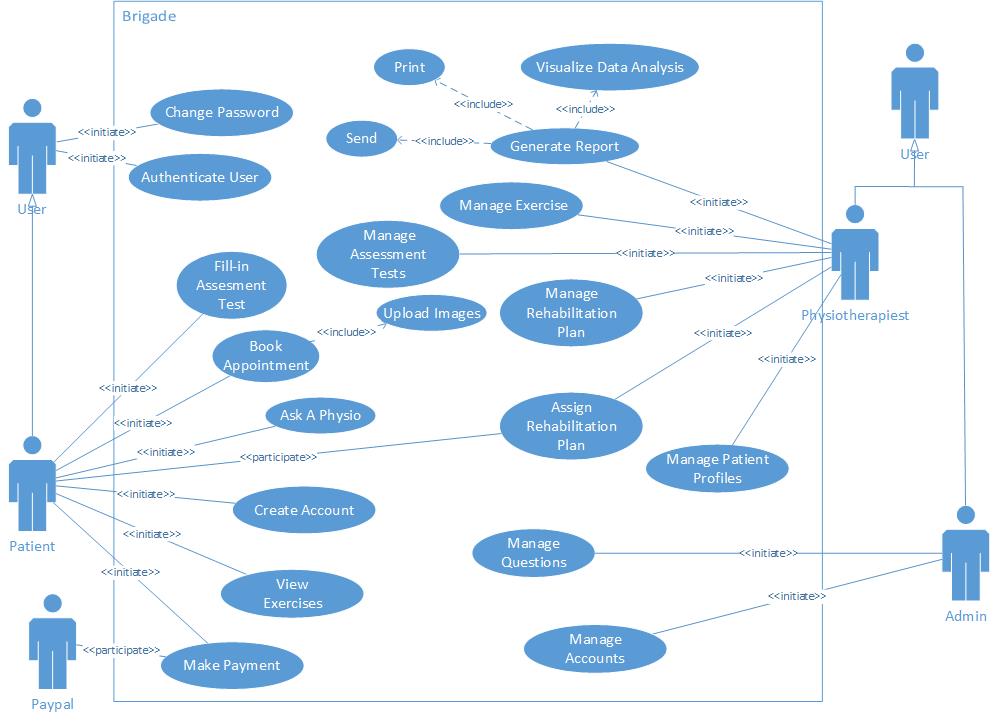
### Use-Case Model Hierarchy

**Self Start**

|  |  |
| --- | --- |
| **Actors** | **Descriptions** |
| User | The user has generic structure for other actors (admin, physiotherapist, patient). |
| Admin | The administrator of the system is responsible for maintaining data of the users. |
| Physiotherapist | Currently employed physiotherapist at Marcotte Physiotherapy. |
| Patient | New or current user accessing Self Start. |
| PayPal | Payment method for the patient. |

|  |  |
| --- | --- |
| **Use Case** | **Descriptions** |
| Change Password | The system allows the user to change their password. |
| Authenticate User | The system authenticates users via their username and password. |
| Fill-In Assessment Test | System provides a simple closed question form for the patient to fill out after doing their exercises. |
| Book Appointment | System allows patient to book an appointment based on employee's schedule. It Includes Upload Images. |
| Ask A Physio | The system allows patients to fill out a form to describe their injury, ask a question, etc. |
| Create Account | The system allows patient to create account, register username and password to the database. |
| View Exercises | The system allows authenticated patient to view the list of exercises assigned by the physiotherapist |
| Make Payment | The system allows patients to make payments via PayPal. |
| Upload Images | System allows the patient to upload images of their injury when booking an appointment. |
| Generate Report | System allows users to generate a report of the patient, summarizing symptoms and treatment. This includes Visualize Data Analysis, Print and Send use cases. |
| Visualize Data Analysis | System provides physiotherapists with the assessment tests results so that they can visualize the effects of the treatment. |
| Print | The system provides an ability to print the generated report. |
| Send | The system enables physiotherapist to send generated report by email. |
| Create Exercise | The system allows physiotherapist to create new exercise to save it in the database. |
| Manage Rehabilitation Plan | The system allows physiotherapist to add, delete or update rehabilitation plans which has lists of exercises and assessment test(s). |
| Assign Rehabilitation Plan | The system allows physiotherapist to assign rehabilitation plan(s) to the patient. |
| Manage Assessment Tests | The system allows physiotherapist to manage the questions that are on the assessment tests, and it allows them to create or delete tests. |
| Manage Patient Profiles | The system allows physiotherapist to manage the details stored in a patient’s profile. |
| Manage Questions | The system allows admin to add, delete, or update the question made by the patient. |
| Manage Accounts | The system allows the administrator to add, delete or update the account of all users in the database. |

### Diagrams of the Use-Case Model



## Assumptions and Dependencies

In the Self Start system, we assume that the unauthenticated user can only post once in the Ask A Physio, for future ones, patients must have an account before posting another one. Secondly, we have assumed that the client is going to pay using PayPal.

# Requirements

## Use-Case Specifications

|  |  |
| --- | --- |
| Use case name | Authenticate User  This use case covers: Objective 10 |
| Participating actor | Initiated by User |
| Entry condition | The user (Patient, Admin, Physiotherapist) is on the Self Start website |
| Flow of event | 1. The user initiates “Authenticate User” from the website interface by logging in.   2. The Self Start website responds by presenting a login screen that includes username and password fields.   1. The user fills the username and password fields. Once complete, the user submits the form by clicking on log in.   4. The Self Start website authenticates the submitted form from the database and displays the acknowledgment |
| Exit condition | * The user receives confirmation of authentication and continues to the home page of Self Start while logged into their account. * The user receives an error message that the username and password submitted was incorrect and to try again. |
| Quality requirement | * Not applicable |

|  |  |
| --- | --- |
| Use case name | Change Password  This use case covers: Feature 7 |
| Participating actor | Initiated by User |
| Entry condition | The user (Patient, Admin, Physiotherapist) is logged on the Self Start website |
| Flow of event | 1. The user initiates “Change Password” from the website’s interface.  2. The Self Start website responds by presenting an interface that includes an old password field (to confirm that it is the proper person making the changes) and a new password field.  3. The user inputs the appropriate information regarding their account in their respective fields.  4. User confirms password update.  6. The Self Start system updates the changes made by the user in the database.  7. The Self Start system sends an email to them telling them that a new password has been set. |
| Exit condition | * User is done changing their password. |
| Quality requirement | * Not applicable |

|  |  |
| --- | --- |
| Use case name | Ask A Physio  This use case covers: Feature 2 |
| Participating actor | Initiated by Patient |
| Entry condition | The patient is on the Self Start website, either logged in or not. |
| Flow of event | 1. The user initiates “Ask A Physio” from the website’s interface.  2. The Self Start website responds by presenting an interface that includes a form asking the patient to input some general information about themselves and their comment explaining their injury.  3. The user inputs the appropriate information in the form.  4. User submits form.  6. The Self Start system adds the form into the database.  7. The Self Start system sends an email to them telling them that they have successfully submitted an “Ask A Physio” form. |
| Exit condition | * Patient submits the “Ask A Physio” form. |
| Quality requirement | * If this is the patient’s first time submitting an “Ask A Physio”, they don’t need to make an account, but to make more in the future, they must make an account (This is an assumption, subject to change if requirement is undesired). |

|  |  |
| --- | --- |
| Use case name | Create Account  This use case covers: Feature 6 |
| Participating actor | Initiated by Patient |
| Entry condition | The patient is on the Self Start website. |
| Flow of event | 1. The user initiates “Create Account” from the website’s interface.  2. The Self Start website responds by presenting an interface that includes personal information fields.  3. The patient inputs the appropriate information in the appropriate fields.  4. Patient confirms that they want to create an account.  5. The Self Start system updates the database.  6. The Self Start system sends an email to them telling them that they have successfully created an account.  7. System logs in patient automatically after the account has been confirmed. |
| Exit condition | * User creates account. |
| Quality requirement | * Email must be unique (Assumption, subject to change if needed to) |

|  |  |
| --- | --- |
| Use case name | Book Appointment  This use case covers: Feature 2, Feature 3, Feature 4 |
| Participating actor | Initiated by Patient |
| Entry condition | The patient is logged onto their account on the Self Start website. |
| Flow of event | 1. The user initiates “Book Appointment” from the website’s interface.  2. The Self Start website responds by presenting the book appointment form to the patient to have them fill it out.  3. The patient inputs the appropriate information in the appropriate fields and answers the questions in the online evaluation form.  4. Patient has the option of uploading an image.  5. The Self Start system updates the database.  6. The Self Start system sends an email to them telling them that they have submitted a book appointment request. |
| Exit condition | * Patient submits book appointment. |
| Quality requirement | * None |

|  |  |
| --- | --- |
| Use case name | Make Payment  This use case covers: Feature 30 |
| Participating actor | Initiated by Patient |
| Entry condition | The patient is logged onto their account on the Self Start website. |
| Flow of event | 1. The user initiates “Make Payment” from the website’s interface.  2. The Self Start website responds by presenting the payment form to the patient to have them fill it out.  3. The patient inputs the appropriate information in the appropriate fields.  4. The Self Start system contacts the bank API to authenticate the user and make the payment.  5. The Self Start system updates the database.  6. The Self Start sends an email to the patient confirming that the payment has been confirmed and gone through. |
| Exit condition | * Patient confirms payment. |
| Quality requirement | * Payment method used must be PayPal. |

|  |  |
| --- | --- |
| Use case name | View Exercises  This use case covers: Feature 18 |
| Participating actor | Initiated by Patient |
| Entry condition | The patient is logged onto their account on the Self Start website. |
| Flow of event | 1. The user initiates “View Exercises” from the website’s interface.  2. The Self Start website responds by presenting an interface that shows guides on what exercises to do and how to do them. |
| Exit condition | * Patient is done viewing exercises. |
| Quality requirement | * Exercises are presented in the form of text and/or simple animations. |

|  |  |
| --- | --- |
| Use case name | Fill-in Assessment Test  This use case covers: Feature 19, Feature 20 |
| Participating actor | Initiated by Patient |
| Entry condition | The patient is logged onto their account on the Self Start website. |
| Flow of event | 1. The user initiates “Fill-in Assessment Tests” from the website’s interface.  2. The Self Start website responds by presenting the “Fill-in Assessment Test” form for the patient fill out.  3. The patient inputs the appropriate information in the appropriate fields.  4. The patient submits the form by clicking on the submit button.  5. The Self Start system updates the database. |
| Exit condition | * Patient submit Assessment Test. |
| Quality requirement | * None. |

|  |  |
| --- | --- |
| Use case name | Manage Accounts  This use case covers: Feature 7, Feature 27, Feature 28, Feature 29 |
| Participating actor | Initiated by Admin |
| Entry condition | The Admin is logged onto Self Start using the admin account. |
| Flow of event | 1. The admin initiates “Manage Accounts” from the admin’s account interface.  2. The Self Start website responds by presenting an interface that contains a variety of different options to manage accounts of patients and physiotherapist.  3. The admin can reset a patient’s forgotten password.  4. The admin can create, edit, or delete patient and physiotherapist accounts.  5. The Self Start system saves all changes to the database. |
| Exit condition | * Admin updates changes. |
| Quality requirement | * None. |

|  |  |
| --- | --- |
| Use case name | Manage Questions  This use case covers: Feature 5 |
| Participating actor | Initiated by Admin |
| Entry condition | The Admin is logged onto Self Start using the admin account. |
| Flow of event | 1. The admin initiates “Manage Questions” from the admin’s account interface.  2. The Self Start website responds by presenting an interface that contains the current introduction form.  3. The admin can add, edit, or delete questions from the introduction form.  4. The admin, after updating the introduction form, clicks on the update button.  5. The Self Start system saves all changes to the database. |
| Exit condition | * Admin updates changes. |
| Quality requirement | * None. |

|  |  |
| --- | --- |
| Use case name | Manage Patient Profiles  This use case covers: Feature 8 |
| Participating actor | Initiated by Physiotherapist |
| Entry condition | The Physiotherapist is logged onto Self Start using the physiotherapist account. |
| Flow of event | 1. The physiotherapist initiates “Manage Patient Profiles” from the physiotherapist’s account interface.  2. The Self Start website responds by presenting an interface that contains a list of the physiotherapist’s patients.  3. The physiotherapist can add, change, or delete information regarding their patient’s profile.  4. The physiotherapist, after updating the patient’s account, clicks on the update button.  5. The Self Start system saves all changes to the database. |
| Exit condition | * Physiotherapist updates changes. |
| Quality requirement | * None. |

|  |  |
| --- | --- |
| Use case name | Assign Rehabilitation Plans  This use case covers: Feature 16, Feature 22 |
| Participating actor | Initiated by Physiotherapist |
| Entry condition | The Physiotherapist is logged onto Self Start using the physiotherapist account. |
| Flow of event | 1. The physiotherapist initiates “Assign Rehabilitation Plans” from the physiotherapist’s account interface.  2. The Self Start website responds by presenting an interface that contains the rehab plans and patients (subject to change).  3. The physiotherapist can assign one or more rehab plans to one or more patients.  4. The physiotherapist, after assigning the rehab plan, clicks on the update button.  5. The Self Start system saves all changes to the database.  6. The Self Start system sends an email to the patients to notify them that they have received a rehab plan(s). |
| Exit condition | * Physiotherapist updates changes. |
| Quality requirement | * None. |

|  |  |
| --- | --- |
| Use case name | Manage Rehabilitation Plan  This use case covers: Feature 9, Feature 11, Feature 12, Feature 13 |
| Participating actor | Initiated by Physiotherapist |
| Entry condition | The Physiotherapist is logged onto Self Start using the physiotherapist account. |
| Flow of event | 1. The physiotherapist initiates “Manage Rehabilitation Plans” from the physiotherapist’s account interface.  2. The Self Start website responds by presenting an interface that includes the current rehab plans made and some other functionalities.  3. The physiotherapist can create rehab plans.  4. The physiotherapist can add, edit, or delete exercises and self-assessment activities to the rehabilitation plans.  5. The physiotherapist, after managing the rehabilitation plans, clicks on the update button.  6. The Self Start system saves all changes to the database. |
| Exit condition | * Physiotherapist updates changes. |
| Quality requirement | * None. |

|  |  |
| --- | --- |
| Use case name | Manage Assessment Tests  This use case covers: Feature 15 |
| Participating actor | Initiated by Physiotherapist |
| Entry condition | The Physiotherapist is logged onto Self Start using the physiotherapist account. |
| Flow of event | 1. The physiotherapist initiates “Manage Assessment Tests” from the physiotherapist’s account interface.  2. The Self Start website responds by presenting an interface that includes the current assessment test made and some other functionalities.  3. The physiotherapist can create, edit, and delete assessment tests.  4. The physiotherapist, after managing the assessment tests, clicks on the update button.  6. The Self Start system saves all changes to the database. |
| Exit condition | * Physiotherapist updates changes. |
| Quality requirement | * None. |

|  |  |
| --- | --- |
| Use case name | Manage Exercises  This use case covers: Feature 10, Feature 14 |
| Participating actor | Initiated by Physiotherapist |
| Entry condition | The Physiotherapist is logged onto Self Start using the physiotherapist account. |
| Flow of event | 1. The physiotherapist initiates “Manage Exercises” from the physiotherapist’s account interface.  2. The Self Start website responds by presenting an interface that includes the current exercises made and some other functionalities.  3. The physiotherapist can create, edit, and delete exercises.  4. The physiotherapist, after managing the exercises, clicks on the update button.  6. The Self Start system saves all changes to the database. |
| Exit condition | * Physiotherapist updates changes. |
| Quality requirement | * None. |

|  |  |
| --- | --- |
| Use case name | Generate Reports  This use case covers: Feature 17, Feature 21, Feature 23 |
| Participating actor | Initiated by Physiotherapist |
| Entry condition | The Physiotherapist is logged onto the Self Start using the physiotherapist account. |
| Flow of event | 1. The physiotherapist initiates “Generate Reports” from the physiotherapist’s account interface.  2. The Self Start website responds by generating a report and downloading it to the physiotherapist’s computer.  3. The physiotherapist can print, email the report to the patient, or generate data analysis. |
| Exit condition | * Physiotherapist either prints, or sends the report. |
| Quality requirement | * None. |

## Functionality

1. The system should provide a way for users (patient, physiotherapist, admin) to log in. Upon login, the server system authenticates encrypted username and password. The system should dynamically provide user different level of power depending on their status.
2. The system will offer a single admin account. This admin account has the capacity to create, edit, or delete accounts of different power (patient, physiotherapist). Additionally, administrators have the power to manage accounts in the database.
3. Every user can change their own password, and the administrator is allowed to reset user’s password.
4. The system should provide the physiotherapist with the ability to create new exercises and manage plans accordingly.
5. The system should provide a form when modifying user data, providing all the attributes of its corresponding user.
6. The system should provide a form for a user to fill out that enables the user to include an image, book an appointment, and send it to the available physiotherapist to evaluate.
7. The system should provide the physiotherapist with the functionality of continuing or terminating the treatment(s) of the user.
8. The system should allow user to fill out the response after the treatment have concluded.
9. The system should support Node.js framework.
10. The system administrator shall have the ability to add users with different level of privilege.

## Usability

1. The newly developed Self Start website portal will allow users to view and interact with pertinent information about their account and treatment plans while still being easy to use and navigate. Through intuitive and logical design choices, Self Start will remain robust and maintain user-friendliness. As such, the portal will require minimal time for power and normal users alike to become productive at operations.
2. We anticipate the most common tasks for users on the Self Start website would be to make an appointment and view a personal treatment plan. Continuing the theme of concise and intuitive design, we measure these tasks will take less than five clicks to accomplish.
3. Responsiveness is key for excellent user experience in today’s world. It is frustrating for users to click on something and have seemingly nothing to happen for a few moments while the page loads. As such, we will implement a loading screen or small indications to acknowledge a user’s requests while background data is being loaded, such as personal treatment plans that are being transported from a database or the authorization of login credentials.
4. Taking into the account the special consideration that many of the potential users of the Self Start portal will be patients suffering physical pains, conforming to a specific usability standard is important. For this reason, we will be conforming to IBM’s CUA standards and allowing users to navigate the whole website using just the keyboard. Not only does this make it easier for users who struggle using a mouse but navigating through inputs such as those for login credentials is faster when using the Tab key versus using the mouse.

## Reliability

1. Ideally, the Self Start system should be available 100% of the time. Users should be able to access the site at any time of their choosing. If maintenance needs to be done, two days of notice should be given so users can be sufficiently aware of the temporary service outage and developers can implement important updates quickly. Maintenance should never take more than 8 hours and should be done during predetermined off-peak system usage hours. Due to the multiple functionalities of the system, degraded operations should be available in certain scenarios. For example, if there is an issue with the database pulling information on treatment plans, the system should still allow users to book appointments with the Marcotte clinic.
2. The Self Start system will be developed in such a way that we believe will never fail. This of course is not a realistic assumption; unforeseen failures (or bugs) will surely arise in the system. But a bug is a one-off problem that will be fixed with a patch. The time between the finding of bugs can be used to predict the timeline of when new bugs will be exposed in the future. Although this can give us an estimate of Mean Time Between Failures, this number is likely to change throughout the product lifecycle and very difficult to estimate prior to the release of the system. Thus, unlike the predictable failures of physical systems, the MTBF of Self Start cannot be accurately measured.
3. The Mean Time to Repair should be no longer than 10 hours. This gives developers a few hours to find the bug and then a reasonable amount of time to fix the failure.
4. The system will be outputting patient treatment plans. Since these plans are administered by the physiotherapists and not the system itself, accuracy is not a major concern with our system. The system should only display these plans upon request in exactly the format the physiotherapist has transcribed.
5. The Self Start system should have no more than 15 bugs/KLOC. This is derived from the industry average of having 85% of bugs found before deployment, leaving about .75 bugs per function point which is roughly 50 lines. Although 15 bugs sound high, it is actually a very good balance between time/cost and effectiveness. Also, these 15 bugs/KLOC are the bugs that remain in the system after all the testing is complete and performance is deemed fit to release—as such any obvious points of failures or chronic issues will not be a part of this figure.
6. Minor bugs: bugs which effect the general presentation of the site. Examples of these type of bugs include graphical bugs like textboxes appearing out of place or miscellaneous abnormalities like “auto-grow” properties of input fields not functioning correctly. 15bugs/KLOC allowed.

Significant bugs: bugs that can hamper a user experience on the system. Examples of significant bugs include issues where users cannot access their treatment plan, users can’t book an appointment with the clinic, and available physiotherapists aren’t properly displayed to the user. No Bugs of this nature will be tolerated. They should be found during the testing phase and fixed before release of the system.

Chronic bugs: bugs that will unintentionally delete information, delete users from the database, allow unauthorized users to view medical treatments plans or records of other users (which is a huge violation of doctor patient confidentiality laws), shutdown the operation of the site temporarily or permanently, and/or otherwise make unauthorized changes to any of the system’s databases. Absolutely no bugs of this nature will be tolerated in the release of Self Start.

## Performance

1. The longest response times should come from the “Create Account”, “View Exercises”, and “Make Payment” use cases as these use cases are that ones that receive or send the most information to the database.

* Create Account (Average: 3 seconds, Max: 5 Seconds)
* View Exercises (Average: 3 seconds, Max: 6 Seconds)

The loading of small videos or animations depicting exercises will increase the response time.

* Make Payment (Average: 4 seconds, Max 8 Seconds)

The interaction with online banking services is out of the control of the system and may take a significant amount of response time.

1. Assuming the system will have a total of 300 regular users, the system should be able to do 60 transactions per second, or 20% of the user base at once.
2. In the event of system degradation, the system should be able to assume degraded modes of operations as specified in section 3.4. At the same time, the system should notify system admin of the issue so that developers can perform maintenance as soon as possible.

## Supportability

### <Supportability Requirement One>

Cross-Platform Support: The system should function the same on all browsers and on all operating systems. This means using libraries and languages that are supported by all browsers and operating systems.

### < Supportability Requirement Two>

Maintenance Tools: The system will provide basic maintenance access to the administrator account so that the administrator can make modifications without needing to modify the code.

## Design Constraints

### <Design Constraint One>

Accessibility: The system should be fully functional at all times. This includes if the user has JavaScript disabled or if some elements fail to load. The system also needs to be accessible to people with disabilities such as the visually-impaired.

3.7.2 *<Design Constraint Two>*

Code Quality: The code needs to be written so that it conforms to good coding practices. The code should be written in a modular way in order to ensure adaptability and maintenance is possible.

*3.7.3 <Design Constraint Three>*

Robustness: The system needs to be able to cope with errors and perform well under unusual conditions. This means writing clean, easy to read code. This also means ensuring the system does not need to be rebooted often.

## Online User Documentation and Help System Requirements

The team will provide technical documentation that can be used to troubleshoot the program.

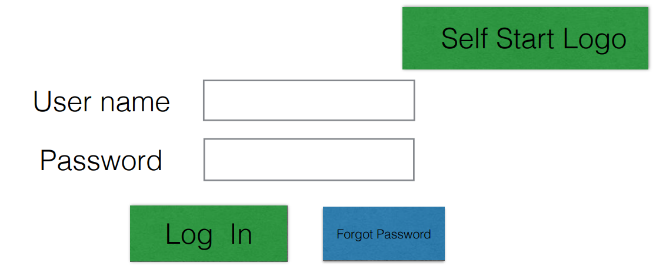
## Purchased Components

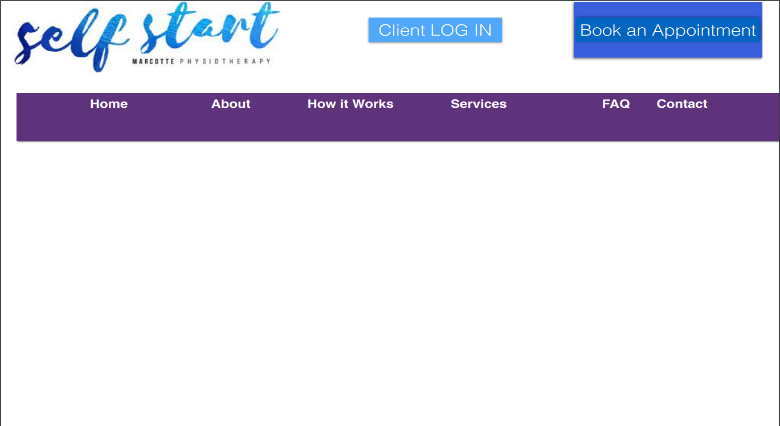
All components will be created by the Brigade team and will not require any purchases. The team does not believe any third-party APIs will be needed.

## Interfaces

### User Interfaces

All users of the web-based application should see the welcome page when he/she opens the website, see Figure 1. At the top of the welcome page the user will be allowed to log-in or book an appointment, both directing the user to the log-in page, see figure 2.





Welcome

Figure 1 – Welcome page Figure 2 – Login page

In Figure 3, the video of the exercise is shown. When the physiotherapist creates the client’s clinic-defined standard rehabilitation plans (for the common injuries), it will include exercises and self-assessment activities.

The physiotherapist and administrators interact with the system through the easy to follow browser based screens. A physiotherapist should be able to register on the webpage to log-in and manage the client exercises, rehabilitation plans and assessment tests. An administrator should also be able to log-in to the webpage where he/she can administer the system by for instance editing user accounts and settings.

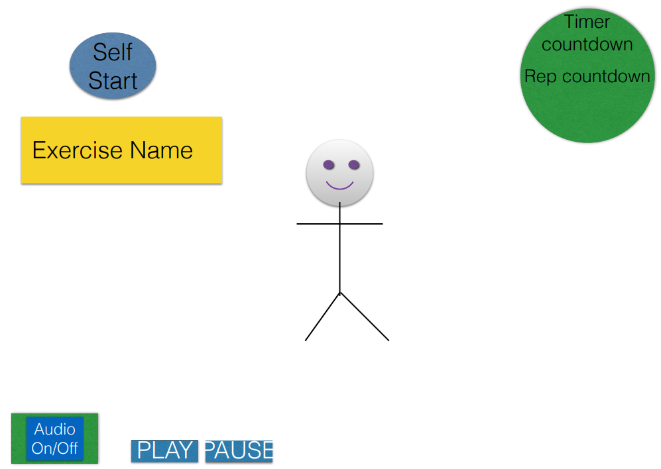


Figure 3 – Exercise video

### Hardware Interfaces

This project requires a hardware interface, including a connection to the internet. A server that is accessed via HTTP requests and a Secure-Socket Layer (SSL) will be used to ensure security.

### Software Interfaces

1. The Self Start system shall communicate with the PayPal system to identify available payment methods, validate the payments and process payments.
2. The Self Start system shall communicate with a video conferencing tool to provide remote communication between the client and physiotherapist
3. The Self Start system shall communicate with Let’s Encrypt to publish the Self Start application over a secure web using TLS technology

The user interface for the software runs on cross web browsers by which the user can have access to the system.

### Communications Interfaces

As briefly discussed in the hardware interface, the Self Start system shall use the HTTP requests for communication over the internet and SSL providing encryption for the communication.

## Licensing Requirements

All the software used on this project are open source and don’t need the purchasing of a license.

## Legal, Copyright and Other Notices

Brigade is the sole owner of its logo, trademark and has certain copywrite notices.

## Applicable Standards

The privacy of the client data and actions will be protected using the standard encryption and key exchange protocol existing in the transport layer security protocol suite (TLS). Clients, however, need to consent to the exercises they are doing on their own, as the clinic will not be reliable for any injuries. The Self Start system will generate its security certificates from the Let’s Encrypt services. Let’s Encrypt is a free, automated, and open certificate authority brought to you by the non-profit Internet Security Research Group (ISRG).