**Software Requirements Specification**

Version 1.1

for

The Force Awakens

Prepared by

|  |  |  |
| --- | --- | --- |
| Nicholas Burdet | 29613773 |  |
| Georges Mathieu | 26863477 |  |
| Olivier Cameron-Chevrier | 27228805 |  |
| Stefano Pace | 27454716 | Stefano.pace12@gmail.com |
| Julian Ippolito | 27419112 |  |
| Adam Trudeau-Arcaro | 27459157 |  |
| Joey Tedeschi | 27513062 |  |

|  |  |
| --- | --- |
| Instructor: | Dr. Constantinos Constantinides, P.Eng. |
| Course: | SOEN343: Software Architecture and Design I.  Section H, Fall term, 2016. |
| Date: | November 25th, 2016 |

**Document history**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| November 12th, 2016 | V1.0 | Preliminary Merging of Documentation Sections | Stefano Pace |
| November 13th, 2016 | V1.1 | Continuation of Merging Documentation Sections | Stefano Pace |
|  |  |  |  |
|  |  |  |  |

**Table of contents**

[1. Introduction 5](#_Toc437438919)

[Purpose 5](#_Toc437438920)

[Scope 5](#_Toc437438921)

[Definitions, acronyms, and abbreviations 5](#_Toc437438922)

[References 5](#_Toc437438923)

[2. Overall description 6](#_Toc437438924)

[Product perspective 6](#_Toc437438925)

[Product functions 6](#_Toc437438926)

[User characteristics 6](#_Toc437438927)

[Constraints 6](#_Toc437438928)

[Assumptions and dependencies 6](#_Toc437438929)

[3. Specific requirements 7](#_Toc437438930)

[External interfaces 7](#_Toc437438931)

[Functionality 7](#_Toc437438932)

[Actor goal list 7](#_Toc437438933)

[Use case view 8](#_Toc437438934)

[Reliability 8](#_Toc437438935)

[Usability 8](#_Toc437438936)

[Efficiency 8](#_Toc437438937)

[Maintainability 9](#_Toc437438938)

[Portability 9](#_Toc437438939)

[Design constraints 9](#_Toc437438940)

[(On-line) user documentation and help 9](#_Toc437438941)

[Purchased and Installed Components 9](#_Toc437438942)

[3. Analysis Models 10](#_Toc437438945)

**List of figures**

[Figure 1. Use case model. 7](#_Toc353553752)

# Introduction

The introduction of the Software Requirements Specifications Document provides an overview of the entire document.

## Purpose

This section defines the role or purpose of the Software Requirements Specifications Document and briefly describes the structure of the document. Identify the intended audience for the document is identified, with an indication of how they are expected to use the document.

The purpose of this Software Requirements Specifications Document (SRS) is to outline the capabilities and functions of the Room Reservation System by explaining the goal and the characteristics of the system. It will provide further details on the system behavior such as what the system shall do under its constraints and how it will respond depending on the user’s actions. This document is addressed to the stakeholders (the students and the staff), the developers, the professors and the correctors who will be evaluating the Online Conference Room Reservation System.

## Scope

A brief description of what the Software Requirements Specifications Document applies to; what is affected or influenced by this document.

This Software Requirements Specifications document for the Online Conference Room Reservation System will be a Web-based System designed for the students of Concordia University. The system will be built to better organize and simplify the process to reserve a conference room by allowing the users of the system to easily select a room to book, view the booked rooms, and to be placed on a waiting list of their desired room if the room happens to be full. Making a simple and effective interface to perform these actions will improve the efficiency of the users allowing them to better visualize and access the rooms required.

## Definitions, acronyms, and abbreviations

Provides the definitions of all terms, acronyms, and abbreviations required to properly interpret the Software Requirements Specifications Document. This information may be provided by reference to the project’s Glossary.

|  |  |
| --- | --- |
| **Word** | **Definition** |
| Corrector | Person responsible to evaluate the project and its components in order to assign a result. |
| Database | Collection of all the information monitored by this system. |
| Interface | Computer hardware or software designed to communicate information between hardware devices, between software programs, between devices and programs, or between a device and a user. |
| Software Requirements Specification | A document that captures complete description about how the system is expected to perform usually signed off at the end of requirements engineering phase. |
| Stakeholder | A person or group that has an investment, share, or interest in something, as a business or industry. |
| User | Person who uses the system. |

|  |  |
| --- | --- |
| **Word** | **Acronym** |
| Software Requirements Specification | SRS |

## References

Provide a list of all documents referenced in the SRS.

[1] C. Constantinides, "SRS", 2016.

[2] "The Definition of Stakeholder", *Dictionary.com*, 2016. [Online]. Available: http://www.dictionary.com/browse/stakeholder?s=t. [Accessed: 01- Nov- 2016].

[3] "The Definition of Interface", *Dictionary.com*, 2016. [Online]. Available: http://www.dictionary.com/browse/interface?s=t. [Accessed: 01- Nov- 2016].

[4] "Software Requirement Specification", *www.tutorialspoint.com*, 2016. [Online]. Available: https://www.tutorialspoint.com/software\_testing\_dictionary/software\_requirement\_specification.htm. [Accessed: 31- Oct- 2016].

[5] “Self-contained system definition” , *www.thefreedictionary.*com, 2016. [Online}. Available: http://www.thefreedictionary.com/self-contained [Accessed 13-Nov-2016].

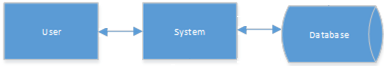
# Overall description

This section describes a background to the requirements: The general factors that affect the product, such as constraints, assumptions and dependencies.

## Product perspective

Is the product self-contained? If not, then put the product into perspective with other related products. Use a block diagram to show the big picture.

A self-contained system is defined as a system which constitutes a complete and independent unit in and of itself. It has no dependency on other systems and is completely self-sufficient [5]. Based on this definition, it can be can concluded that the Room Reservation system is not a self-sufficient system. Whenever a user invokes a feature of the system, the system must perform many queries to the database to check conditions on availabilities etc.. Therefore it can be concluded that the system has a direct dependency on the database.



## Product functions

Provide a summary of the major system functions.

Reserve Function: A user can reserve a specific room at a specific timeslot. If the room is already reserved, the user is put on a waiting list. If the reservation is cancelled, the first user on the waiting list is given the reservation and they are consequently taken off any other waiting list at that timeslot. This is part of the write functions, where only one user can be active at a time.

Cancel Function: A user can delete a reservation at a specific timeslot for a specific room, as long as they are the user who created the reservation. This is part of the write functions, where only one user can be active at a time.

Modify Function: A user can modify the timeslot or the room for their reservation. This is under the condition that the room or timeslot the user wants to modify to is available. This is part of the write functions, where only one user can be active at a time.

View Reservations Function: A user can view any reservations or availabilities. This is a read function so multiple users can view simultaneously.

Identification and authentication: When a user attempts to log in, their username is compared to those in the database to identify the user. If their username is not found, then they are told that no account information can be found for their username. The password entered is used for authentication. If the password is confirmed, they will be allowed to access the system. If not, the user is notified the password is wrong and may try again.

If a client wishes to write and the resource is not available (another client is writing on the resource), then the client must wait, until the resource becomes available. Writers and readers operate in mutual exclusion. The system may allow multiple readers at a time, however only one writer may make a reservation to a specific room at any point in time. The system must provide safety, liveness and fairness.

## User characteristics

Who are the intended users of the system what is their expected educational level, experience and technical expertise?

The only intended users of the system are students of Concordia University. They are expected to be enrolled in either full-time or part-time courses. Only basic familiarity with using a website is expected to be able to use the system. The graphical user interface was made in a very user friendly way, as to provide step by step instructions to the user when he/she wants to use a feature.

## Constraints

Describe any items that will limit the options of the developers (such as regulations, hardware limitations, safety and security etc.)

Since we cannot publicly host our system using public web address, we are limited to opening a port on our server to allow for access. This means we will need to do the following:

* The site will be privately available to only selected students. These users will be sent the link directly to access the website.

To properly address fairness, safety, mutual exclusion and data integrity in the system, (so that a user does not reserve too many rooms too often, or does not conflict with another person trying to make a reservation) we will include the following:

* Reservations can only be made once every 5 minutes.
* A specific user cannot have more than 3 reservations a single week period (This is considered on a Monday to Sunday basis).
* A room can only be repeatedly booked for up to a 3-week period.
* Only one user may access a specific room at any given time in order to create, modify or cancel a reservation.

## Assumptions and dependencies

What assumptions are there? For example, a specific operating system should be present on a given hardware platform. If not, this document would have to be changed.

Several factors that may affect the requirements specified in the SRS include:

* Users of the system are assumed to be (currently) enrolled students of Concordia University. As connecting to an actual student registry is outside the scope of the project, mock accounts are used to validate this assumption.
  + This includes having a premade student account, as this same account (e.g.: Used to connect to my.concordia.ca) would be used for this web application.
* Users are assumed to be using an up-to-date web browser. Though this is unlikely to be an issue for most people, using an outdated web browser may lead to inconsistencies in the user interface, and thus possible changes in the requirements listed.
* This system is dependent on the user having a stable connection to the World Wide Web during usage. Possible cuts in connection could lead to inconsistent storage of user requests, and thus changes in the requirements specified.

# Specific requirements

This section contains all requirements in detail: Functional as well as non-functional requirements (quality attributes and constraints). The quality attributes are listed according to the *ISO/IEC 25010* standard that classifies software quality in a structured set of characteristics and sub-characteristics.

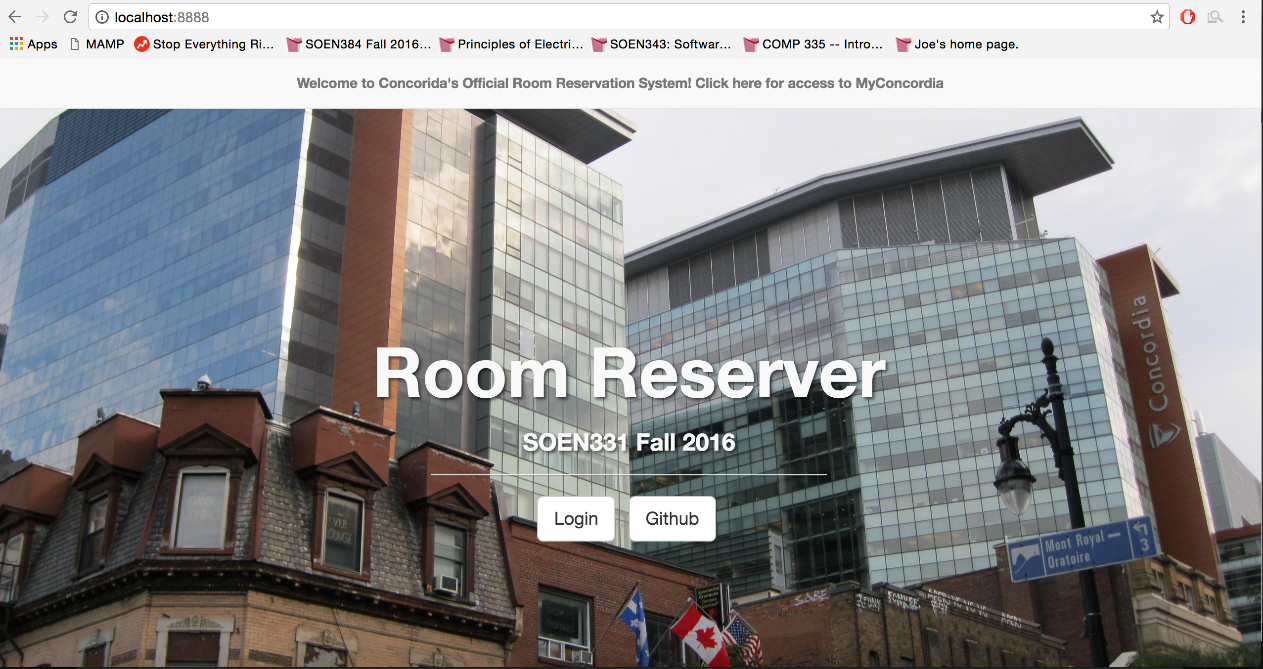
## External interfaces

A detailed description of all inputs into the system and all outputs from it (in terms of content and form).

This section provides a detailed description of all inputs and outputs from the system. Additionally, a description of the hardware, software and communication interfaces used are provided.

User Interfaces

Upon first accessing the website, the user is greeted by the welcome page (Figure 1). From here, the system allows navigation to my.concordia.ca, the team GitHub page, as well as providing the user with the option to login.



Once the user has selected the login feature, a pop-up is generated which accepts the user’s credentials, namely the student’s email and password (Figure 2). This login prompt also provides the option to change their password should the old one be forgotten (Figure 3). If the user’s information is invalid, they are notified as such and can attempt to fix their error.

Successfully logging in to the web application will redirect the user to the Reservation page, and can dictate the exact date they would like to see displayed. Upon selection, a table of times is generated for the user based on the day chosen, including all times currently booked by them, or other students who have used the system (Figure 4). Clicking a time slot that has already been taken would provide the user with the option to be placed on a waitlist, receiving a notification once the time slot has been freed and the place has been given to them (Figure 5). Should the time slot requested not already be booked, the user will be able to then edit the time they would like to book, and confirm their decision via a pop-up (Figure 6). The reservation allows provides the user with the opportunity to edit their student profile through an option in the upper navigation bar. Selecting this option will display a pop-up, upon which all-current user information will be displayed, thus allowing further editing (Figure 7).

Hardware Interfaces

The Hardware Interfaces of the system is summarized plainly by the machine being used to host both the web server and the database. The specification can be found in Purchased and Installed Components section.

Software Interfaces

* Operating System

The Operating System being used by the run-time environment is Windows 7 Professional, which includes Internet information Services (IIS), version 6.1 with PHP 7.0.9.

* Web Server

The website is being designed to run on Internet Information Server version 6.1, with PHP 7.0.9.

* Database

The website accesses a MySQL 5.6.26 MySQL Community Server (GPL) for the following features:

* + - Retrieving Student Information
    - Updating Student Information
    - Creating and Populating the central reservation table
    - Storing Student Reservation Times
    - Updating Waitlists for each available room
* Libraries

No external interfaces are to be used for this system with regards to frameworks, as object-relational structural and behavioral patterns are to be manually implemented. With regards to libraries, jQuery is being used to simplify front-end construction, as is Bootstrap (a front-end framework), though this does not compromise the requirements for an object-oriented system.

Communication Interfaces

* Web Interface

The entire reservation system is accessed over the Internet through a web site. As such, all communication between the various parts of the system is done through the web interface provided to the user, allowing their inputs to be translated to various events. This is all done through the use of HTTP protocol.

## Functionality

Functional requirements capture the intended behaviour of the system. This section contains the *Actor Goal List* and the *Use Case view*.

## Actor goal list

|  |  |
| --- | --- |
| Actor | Goal |
| User/Student | * Log In * Make a reservation * Cancel a reservation * Modify a reservation * View list of made Reservations * View availability of rooms * Edit Profile Information |

## Use case view

The use case model is shown in Figure 1.

\*\*\*\*Missing Edit Profile Information and waitlist functionality

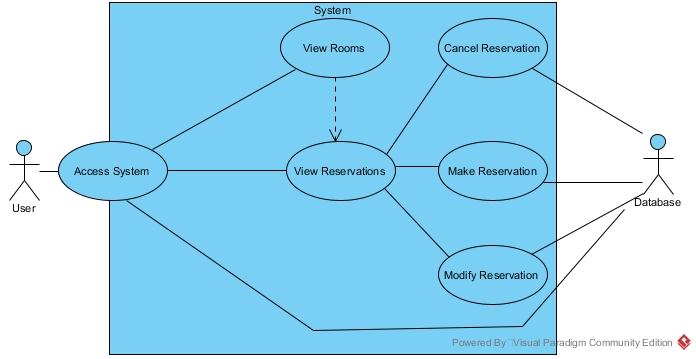


Figure 1. Use case model.

## Reliability

Description goes here.

* The mean time between failures shall exceed 1 month.
* The system shall provide for replication of databases to off-site storage locations.

*Note: In testing the system, we can shorten or prolong this amount of time, since it is currently difficult to determine*

## Usability

Description goes here.

Graphical User Interface

* The system shall provide a uniform look and feel between all the web pages.
* The system shall provide a table with all reservations for the user logged in, as well as other users in the system. Different colors will represent these reservations, as well as a 3rd color will represent available times.
* The system shall provide toolbars for easy navigation between features.

Accessibility

* The site will be privately available to only selected Students. These users will be sent the link directly to access the website.
* The reservation system must be available to use from any commonly used web browser application (and relatively recent version).
* Users will only be allowed to make a reservation in another room after already making one, once every 5 minutes.
* Users will only be allowed to add themselves to a single waiting list for a specific room.
* Users cannot exist in a single waiting list more than once.
* Users cannot maintain more than 3 reservations for a single week period.
* Users may hold a position in any number of waiting lists for reservations, so long as they have not already reached the maximum number of reservations for the week (currently 3).
* Users holding the top position in a waiting list queue will be given the reservation for that room and timeslot if the current holder cancels their reservation.
* Users given a reservation for a room will be removed from all other waiting lists if they have reached the maximum allotted reservations for the week.

*Note: Timeframe and numbers are subject to change upon negotiations*

*Note: We will only be supporting the following commonly used web browsers: Google Chrome, Firefox, Internet Explorer, Opera, Microsoft Edge.*

## Efficiency

Description goes here.

For the system to be as efficient as possible, there are 2 criteria which much be considered:

* Acceptable response time
* Number of mouse clicks/redirects needed to get to information or functionality

To achieve an acceptable response time, the system will be constructed so that when a functionality is invoked, it will take **AT MOST** 1 minute for the system to process the request. As well, to address the number of mouse clicks needed to get to information or functionality, the system GUI was designed in a way to allow the user access to all features on one simple page. By providing all the features of the system on one page, this will limit the amount of redirects and clicks because the user will not have to search for features, they are all available right after the log in.

## Maintainability

Description goes here.

Application Extendibility

Once the final version of the system is completed, it is always possible for more requirements are added later on from the stakeholder. Therefore, the current state of the system should be easily extendable to allow for future requirements to be smoothly integrated into a newer version of the system. The Object Orientation of the system should be implemented in a way that favors the addition of new functions.

Application Testability

In order for the system to be thoroughly tested and vetted before a final version is submitted to the stakeholder, a test environment will be necessary. This test environment will be a separate environment from the live environment. Commonly these are labelled “UAT” and “PRD” respectively. Once a component is validated in the “UAT”, it will then be added to the “PRD” for the stakeholder to verify. As more progress is made in the system, the “UAT” becomes more critical, as when adding new functionality to the system there is always a risk that this make other components unstable.

## Portability

Description goes here.

The service is provided to the user via a web page using bootstrap to format the front end & graphical user interface. The framework allows for flexibility and adjustments to the different screen sizes. This flexibility allows our site to run on all devices including mobile. The web services give our system the flexibility to be run on all machines regardless of the operating system provided there is a compatible browser which is parted of our supported list (see Accessibility section).

## Design constraints

Decisions that must be followed, such as languages, processes, prescribed use of tools, architectural and design constraints, purchased components, class libraries, etc.

Standard Development Tools

The system shall be built using a standard web page development tools that conforms to either IBM’s CUA standards or Microsoft’s GUI standards. The languages used will be a mix of the following; HTML, CSS, PHP, JavaScript, Ajax, JQuery, MySQL. As well, bootstrap will be integrated with the system to provide better styling.

Web Based Product

* The use of a dynamically accessible server will be required to store all relevant information from the users, reservations and rooms.
* The computers must be equipped with compatible and supported web browsers (All major web browsers are supported except for Safari)
* The product must be stored in such a way that allows the client easy access to it
* Response time for loading the product should take no longer than 1 minute
* A general knowledge of basic computer skills is required to use the product

## (On-line) user documentation and help

Non-Applicable

## Purchased and Installed Components

Description.

During the Implementation and Testing of our system we purchased/installed the following hardware and software.

The server used to host our system, The Force Awakens, has the following hardware specifications:

* + Windows Server 2012 Operating System (64-bit)
  + Quad-Core Processor (i7-4790 CPU @ 3.60GHz)
  + 16 GB DDR2 RAM
  + 25 Mbps Bandwidth
  + 2TB Memory

The Software used on the server will be the following:

* + HTML, CSS, PHP, MySQL (WorkBench)
  + GitHub desktop
  + NotePad++
  + Eclipse
  + Bootstrap
  + Windows 7
  + Microsoft Internet Information Services 7

# Analysis Models

List all analysis models used in developing specific requirements previously given in this SRS. Each model should include an introduction and a narrative description. Furthermore, each model should be traceable the SRS’s requirements.

Illustrate (system) ***UML sequence diagrams*** (one for each critical scenario), identify system operations and describe operation contracts, one per critical system operation. You may also use ***UML state diagrams*** to describe critical use cases. Additionally, create a **domain model** for the system. Make sure that each model is traceable to the requirements.