|  |
| --- |
| **Disclaimer**  This is a **template** for the Software Requirements Specification (SRS) that students may use. It povides a **starting point** for the preparation of SRS.  **Note to authors**  If you add any new sections to the document please make sure that you maintain the header and text styles.  Before submission of the first draft of this document please make sure to update the Table of Contents and to delete this page.  **Author**:  Dr. C. Constantinides <cc@cse.concordia.ca> |

**Software Requirements Specification**

Version 1.0

for

<Project name>

Prepared by

|  |  |  |  |
| --- | --- | --- | --- |
| Gabriele Bavaro | 27399103 | gabriele.bavaro@bell.net | |
|  |  |  | |
| Instructor: | | | Dr. C. Constantinides |
| Course: | | | SOEN 343 |
| Date: | | | November 23, 2016 |

**Document history**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| October 1, 2016 | 1.0 | Functionalities and Risks added | Gabriele Bavaro |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of contents**

1.      Introduction. 5

Purpose. 5

Scope. 5

Definitions, acronyms, and abbreviations. 5

References. 5

2.      Overall description. 6

Product perspective. 6

Product functions. 6

User characteristics. 6

Constraints. 6

Assumptions and dependencies. 6

3.      Specific requirements. 7

External interfaces. 7

Functionality. 7

Actor goal list. 7

Use case view.. 8

Reliability. 8

Usability. 8

Efficiency. 8

Maintainability. 9

Portability. 9

Design constraints. 9

(On-line) user documentation and help. 9

Purchased components. 9

Licensing requirements. 9

Legal, copyright and other notices. 9

3.      Analysis Models. 10

**List of figures**

Figure 1. Use case model. 7

# 1.      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.

## Scope

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

## 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.

## References

Provide a list of all documents referenced in the SRS.

# 2.      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.

## Product functions

Provide a summary of the major system functions.

## User characteristics

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

## Constraints

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

There are a few potential risks and constraints associated with the system. There could be compatibility risks between different software, such as a user who utilizes an outdated computer system that can’t handle the software and subsequently fails to initialize it. In addition there could be communication issues between the different software components that make up the system such as the selected database not meshing well with the chosen Object Oriented Programming Language which could result in corrupted data.  Other risks include running out of time to implement critical or non-critical aspects of the system, team members leaving the project in the middle of the work period and writing the system code very poorly which can result in unforeseen consequences. In addition there could be use case risks in which the use cases themselves are poorly done or understood as well as the possibility of there being no scalable code in the end.

## 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.

# 3.      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.

**3.1  Functional Requirements**

The following are the functional requirements of the system which describe what the system will need to do in order to meet the demands of the stakeholder:

·         The system shall maintain a directory of rooms and their availabilities at different timeslot.

. A user shall be able to login and logout of the system.

. A user shall be able to see room availabilities.

. A user shall be able to reserve up to 3 timeslots per week.

. A user shall be able to able to put themselves on a waiting list if a room at a specific timeslot is not available.

. A user shall be able to cancel or replace their reservation by another one.

·         Only one user at a time on the system can have access to a specific room when they are reserving it or are updating or cancelling their reservation.

·         The system shall notify a user on the waiting list that was moved to a confirmed reservation.

·         A waitlist algorithm must be implemented so that users can wait to see if a room is available and then the room given to the next user on the list should the original user un-reserve a room.

·         The system needs to have an algorithm in place to help resolve the conflict of two users arriving at the exact same time to reserve a room.

**3.2  Non-Functional Requirements**

## Non-functional requirements define constraints on the way the software-to-be should satisfy its functional requirements or on the way it should be developed. Functional Suitability

To be completed once the coding is done, it says what was actually implemented vs what was supposed to be implemented.

This characteristic represents the degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions. This characteristic is composed of the following subcharacteristics:

* **Functional completeness.** Degree to which the set of functions covers all the specified tasks and user objectives.
* **Functional correctness.** Degree to which a product or system provides the correct results with the needed degree of precision.
* **Functional appropriateness.** Degree to which the functions facilitate the accomplishment of specified tasks and objectives.

## Performance efficiency

This characteristic represents the performance relative to the amount of resources used under stated conditions. This characteristic is composed of the following subcharacteristics:

* **Time behaviour.** Once
* **Resource utilization.** Degree to which the amounts and types of resources used by a product or system, when performing its functions, meet requirements.
* **Capacity.** Degree to which the maximum limits of a product or system parameter meet requirements.

The following are the non-functional requirements of the system:

·         The system must handle a large number of users at any given time.

· The system must be constructed using an Object Oriented Language (OO). Examples include Java, C++, C# etc.

·         The system must be safe and fair to use for every user.

·         The system must have be fair to every user.

·         The waiting list needs to be implemented as some sort of data structure such as an array or a queue.

·         A room can be booked multiple times up to a certain limit.

·         Each user needs to have a weekly allowance of time that they can use to reserve a room.

·         A room can be reserved for consecutive or non-consecutive time slots.

·         When reserving a room, users should have a certain view page that allows them to select which room they can reserve and at what times they can do so.

·         The wait list should have a limit for each room.

·         Users have a specific allowance of time that they can use to reserve rooms each week that will be renewed at the start of a new week.

·         The system could have extra features implemented such as a favourites list which contains a list of rooms that they prefer to reserve.

## External interfaces

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

## 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 |
|  |  |

## Use case view

The use case model is shown in Figure 1.

Figure 1. Use case model.

## Reliability

Description goes here.

## Usability

Description goes here.

## Efficiency

Description goes here.

## Maintainability

Description goes here.

## Portability

Description goes here.

## Design constraints

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

## (On-line) user documentation and help

Description.

## Purchased components

Description.

## Licensing requirements

Description.

## Legal, copyright and other notices

Description.

# 3.      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.