|  |
| --- |
| **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 |
| November 1, 2016 | 1.1 | Rough finished copy of SRS document completed | Gabriele Bavaro + others |
|  |  |  |  |
|  |  |  |  |

**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

## Purpose

The purpose of the Software Requirements Specifications (SRS) is to define and communicate the many and different software requirements for the product Room Reservation. The version of the product that the SRS document deals with is version 1.0. The audience of the SRS document are the stakeholders of the Room Reservation product and the information contained within the SRS, particularly the requirements, are to be documented in such a way as to provide an understanding to the stakeholders on what the requirements are. The structure of this SRS is inspired by the I-EEE standard 830-1998[1].

## Scope

Room Reservation is an online conference room reservation system software product that reserves different rooms from different buildings, for a set period of time, for use by a student or a group of students which are the end users. The online software system allows students to reserve rooms for a given time slot and to reserve multiple time slots for the same room or different rooms up to a certain limit of rooms and/or timeslots. In addition, if a timeslot for a room is already taken a student may opt to be placed on a waiting list for that room and timeslot. The system only allows authorized users to access and manipulate it. The product is not an open system. The beneficiaries of the software product are the students who as the end users utilize it and Concordia University, which is the stakeholder. The product is developed for Concordia University, a university based in Montreal, Quebec.

## Definitions, acronyms, and abbreviations

FR Functional Requirements

FTP File Transfer Protocol

HTTP Hypertext Transfer Protocol

MYSQL My Structured Query Language

NFR Non-Functional Requirements

OO Object Oriented

SRS Software Requirements Specifications

## References

[1] IEEE std 830-1998, IEEE Recommended Practice for Software Requirements Specifications (SRS), IEEE Computer Society.

[2] Dr. C. Constantinides, Software Architecture and Design I Term Project, Concordia University, September 13, 2016.

# 2.      Overall description

## Product perspective

The purpose of this application is the same as an existing software from Concordia University. It is use to book room reservations for Capstone projects. However, both softwares are not connected to each other. Different database are also used. The software is a mobile application version it is self-contained.

## Product functions

View Reservations:

-Displays a 2 weeks calendar (current week and next week);

-Displays schedule with rooms including their available and reserved time slots;

Make reservation:

-Add student to selected available time slot of the room (maximum of 3 reservations per student per week).

Add to wait list:

-Add student to a waiting list of the selected reserved time slot (maximum of 3 waiting list per student per week).

Add rooms to favorite list:

-Add a specific room on a list of favorite.

Send notifications

## User characteristics

Users of this application are registered engineering students of Concordia University. Users must have applied to an Capstone project in order to .

## 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.**(USE CASE RISKS!!!!)(NO SCALABLE CODE)**

## Assumptions and dependencies

# 3.      Specific requirements

## 3.1 External interfaces

**3.1.1 User interface**

display\*\*, buttons, graph, layout, functions

**3.1.2 Hardware Interfaces**

The  application shall be used with an android smartphone. The smartphone is the only hardware required in order to use the application. The smartphone needs to support Android version 5.0 or higher.

**3.1.3 Software interface**

The system shall be connected to a MYSQL database in which it contains the informations of the students, the rooms, and reservations.

The application shall also be connected with the Spring framework.\*\*

Db, operating system, tools(emulatorGenymotion), librariesSPRING

Data item, messages coming into system and going out

**3.1.4 Communication interfaces**

The application shall have an notification sending system. It sends a notification message to the student 1 hour before the reservation time slot. It also sends a notification to the student when the student get a timeslot that he was on the waiting list.

Any communication functions (no external tools for notification), LINKS, Email, Server(raspberry pi), web browser(no web browsers), FTP or **HTTP**

## 3.2 Functionality

This section contains the *Actor Goal List* which represent the functional requirements and its actors. The list capture the intended behaviour of the system.  This section also contains the *Use Case view*.

## Actor goal list

|  |  |
| --- | --- |
| **Actor** | **Goal** |
| Student | View room reservations. |
|  | Make reservation |
|  | Cancel reservation. |
|  | Modify reservation |
|  | Add to waiting list |
|  | Remove from waitling list |
|  | Add room to favorite |
|  | Login, Logout??? |
| Scheduler subsystem | Purge all reservations weekly. |
| Notification subsystem | A student shall be notified by email 1 hour before the beginning of their reservation. |
|  | After a reservation or wait list cancellation, students on the waiting list shall be notified by email of their new position. |

## Use case view

The use case model is shown in Figure 1.

Use case diagram.png

Figure 1. Use case model.

## Functional Suitability

* **Functional completeness:** the system shall offer all functional requirements that are deemed critical (login,create and cancel reservations and add to waitlist) which represent 87% of the functional requirements and as many of the functional requirements mentioned in the above section as possible.
* **Functional correctness:** The system should present the correct information to the user in 90% of cases.
* **Functional appropriateness:** Each user activity (make, replace or cancel reservation) shall not take more than 3 steps for the user to accomplish.

## Performance efficiency

* **Time behaviour:** The response to each user click or touch screen tap should take less than 2 seconds.
* **Resource utilization:** On the front-end, the user shall use an android mobile device with version 4.0 or 4.1 installed on it to be able to run the application. The system back-end will use a Wamp Server which consists of an Apache web server, MySQL database which will be connected to the Spring framework (Java language).
* **Capacity**: The system shall accommodate 100 concurrent users.

## Usability

* **Appropriateness recognizability:** 80% of users should find that the system satisfies their needs.
* **Learnability:** It should take less than 30 minutes for a new users to figures out how to add, change and cancel reservations.
* **Operability:**  It should take less than 3 clicks (taps on touch screen) for the user to accomplish any of the main activities (add reservation, add to waitlist, change reservation or cancel reservation).
* **User error protection:** The system shall send reminder messages 1h before reservation time starts so student can cancel them if they can’t make it. CRUD operations shall require confirmation at the end for the changes performed to be saved in the database.
* **User interface aesthetics:** UI should implement many interface patterns found in similar applications to reduce confusion. A small sample survey shall be used to test it.
* **Accessibility:** User experience should feel familiar in its implementation and UI to 90% of users . Color blindness shall be taken into consideration when using colors to indicate results of operations. Front-end design shall follow Android best practices to allow augmentation of font size by user if necessary.

## Compatibility

* **Co-existence.** The Android front-end mobile application shall co-exist with the Spring backend framework.
* **Interoperability.** The Android front-end mobile application shall exchange and communicate and receive information from the backend Spring framework through GET and POST HTTP requests.

## Reliability

Degree to which a system, product or component performs specified functions under specified conditions for a specified period of time. This characteristic is composed of the following sub-characteristics:

* **Maturity.** Mean time between failures should be ?
* **Availability**. The system shall maintain a directory of rooms and their availabilities at different time slot. A Room instance shall only be accessed by one user at a time for the operations reserve/cancel/update.

The system shall not make a Room instance available to other users if its WRITE status is set to ‘true’ as it is being reserved/canceled/updated by another user.

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.

* **Fault tolerance.** Degree to which a system, product or component operates as intended despite the presence of hardware or software faults.
* **Recoverability.** Degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system.

## Security

* **Confidentiality.** The system shall not disclose the identity of the room holders who have confirmed reservations to other users nor the identity of the people on the waiting list.
* **Integrity.** The system shall be safe and fair to every user.
* **Non-repudiation.** Database transactions shall be logged and saved.
* **Accountability.** Logs shall not be modifiable by Administrator.
* **Authenticity.** Long password (minimum 8 characters) shall be required from users.

## Maintainability

* **Modularity.** A multi-layered system shall be designed to separate responsibilities and lower coupling. An object-oriented architectural style shall be used.
* **Reusability.** Multi-layer architecture shall allow main domain classes to be reused if need be as they do not directly communicate with low level layers.
* **Analysability.**  Logiscope shall be used to analyse the code. The report produced for this characteristic will include analysis of weighted methods per class , class comment rate, number of base classes and direct classes associated to each class. The resulting grade shall not be below fair.
* **Testability.** Logiscope shall be used to analyse the code. The report produced for this characteristic will include analysis of weighted methods per class, the total number of methods per class and the number of classes used directly by each current class. The resulting grade shall not be below fair.

## Portability

* **Adaptability.** Android app shall work for devices with an Android OS version of 4.0 or 4.1. User interface shall automatically fit different screen sizes of users’ mobile devices that run Android OS 4.0 or 4.1.

## Design constraints

The design constraints that need to be followed are that the product must be an online system that utilizes an OO programming language alongside any libraries that can be incorporated into said language. The system may also be a mobile app as well, such as an android app. In addition the system must implement a database that will be incorporated into the finished product. In addition the system must support multiple users accessing and using it without any software bugs or performance issues showing up. The finished product must be constructed using an online system constructing framework, an example would be Android Studios. However the framework cannot do automatic work such as automatically implementing databases. The code structure must include structural and behavioural patterns. No purchased components were needed for the product.

## (On-line) user documentation and help

The on-line documentation that a user can use to better understand the initial product will be developed for a later version of the software system.

## Purchased components

No components were purchased to construct the product. All components used were either open sourced or free licenses.

## Licensing requirements

No products used to construct the system needed to be licensed for a monetary value. All the tools licensed to construct the system had free licenses.Geny motion (an emulator) and Intellija (backend code developer) have free licenses which were used for the project. Those tools which were used but did not have any licenses associated with them were Android Studio (front end code developer), Spring (framework) and MYSQL(database).

## Legal, copyright and other notices

All products used to create the product were free online tools which were legally downloaded from their respective websites. Some of the products utilized did have licenses associated with them. A list of these is found in the Licensing Requirements section. The rest of the products used to construct the final system were open source products and as such did not have any licenses or copyrights associated with them.

# 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.