INSE6260 Software Quality Assurance

Room Reservation Project

Software Requirements Specification

04/06/2024

Merlin Duval - 40286249
Yash Khosla - 40232363

Table of Contents

1. Introduction

1.1 Purpose

This Software Requirement Specification (SRS) aims to provide a thorough and detailed list of information that a software engineer needs to properly design and implement the room reservation software.

1.2 Scope

- The room reservation software will consist of an online, secure interface on which the user will be able to perform all reservation operations. It includes, according to the user's rights, choosing what room to reserve and for how long.
- The software will associate a room to a user for a certain amount of time, on a specified date. It will also limit the user to its right (a simple student cannot reserve an auditorium for an entire day)
- The software will provide a secure way of room reservation based on user right, specifications of the room and former reservations.

1.3 Overview

The SRS contains 4 important parts :

- A general description of the software explaining the context of the development and basic information to help the developer
- A specific requirements part where every technical and non-technical requirements are described in details
- Analysis models where different diagrams are presented such as sequence diagram and data flow diagram
- A change management process part explaining how the SRS can be updated formally.

2. General Description

2.1 Product Perspective

The room reservation software takes place in a well designed university intranet that supports databases and other software. It solves a lot of deployment concerns and provides security to the users.

2.2 Product Functions

The software will securely perform associations between user and room, based on a date, time, and duration of the reservation. It will also consider user rights and reservations rules which are adjustables.

2.3 User Characteristics

By user we mean every person likely to reserve a room, it includes undergraduate students, graduate students, interns, teachers, administration staff and others.
Each kind of user will have different rights as a teacher may need more specific rooms than an undergraduate student for example.

2.4 General Constraints

The different kinds of rooms and users must be adjustable as well as the reservations right according to the period. For example, the first week of the term may be more restrictive for students because the administration staff may need more room in this period. These restrictions can be seen as a tightening of the priority that users have over each other. It means that a teacher may reserve a room that was already reserved by a student, if the university rules allow it.

2.5 Assumptions and Dependencies

This software will be available on a secure interface proper to the university, such as an Intranet.

We also consider that the university already possesses an exhaustive database containing all the rooms and their specifications (capacity, type of room), and a database containing all the user id and their status in the university (student, teacher, staff). These databases should be securely accessible from the university private network where the room reservation software will be deployed.

The room reservation software will not do more than associate users to rooms online. It means that the users are supposed not to go in a room without a reservation. The software will not imply physical restriction about entering a room.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

1. The software must be accessible in a secure way : there must be a login page that follows in date security rules, such as using reliable login libraries.
2. The user interface should allow the user to choose only the room that he has the right to reserve, according to university rules.
3. There must be an administrator interface on which it is possible to remove reservation, add new rooms and other actions that can be added in the future.

3.1.2 Hardware Interfaces

1. The software must be accessible both on a computer and a mobile phone, regardless of the operating system.

3.1.3 Software Interfaces

1. The software interface must stay modifiable. New actions can be added to users administrators.

3.2 Functional Requirements

3.2.1 Reservation of a room

3.2.1.1 Introduction

This function should allow a user to reserve a room on a specified time slot, if the room is not already reserved on this time slot. We consider that the user interface is already protecting an user from reserving a room that he's not allowed to reserve, another verification about this point is not required.

3.2.1.2 Inputs

The reservation of a room takes as input an user, a room, and a time slot.

3.2.1.3 Processing

This function checks if the specified room is available for the specified time slot by requesting room.isReserved(timeSlot), if it returns true, it makes the link between an user, a room and a time slot by creating a new reservation.

3.2.1.4 Outputs

If the room is already reserved, the function doesn't return anything, the user just gets a message notifying that this room is not available on this time slot.

If the room is available on this time slot, the function creates a reservation and the user gets a success message.

3.2.1.5 Error Handling

The input verification is ensured by the interfaces, as an invalid user cannot access the software, the room input is among a verified database, and the time slot is checked in the function.
An unexpected error will raise an error message to the user.

3.2.2 Modification of a reservation

3.2.2.1 Introduction

This function allows an user to modify its reservation according to a new room and/or a new time slot.

3.2.2.2 Inputs

It takes as input a user, a reservation, a room and/or a time slot.

3.2.2.3 Processing

The current reservation will be removed and a new one with the new parameters will be created the same way it is usually done (see 3.2.1).

3.2.2.4 Outputs

This function returns a new reservation and notify the user that the reservation is done.

3.2.2.5 Error Handling

An error message is raised in case of an unexpected error. The current reservation is removed only if the new one can effectively be created, it means that all the verifications are done before that the reservation is removed.

3.2.3 Removing a reservation

3.2.3.1 Introduction

This function should allow a regular user to remove a reservation he made.
This function should also allow an admin to remove any reservation.

3.2.3.2 Inputs

The inputs are the user and the reservation.

3.2.3.3 Processing

The function simply removes the reservation. The list of reservations that are removable is stated by the status of the user, and the university rules. It means that this function should not check if the user has the right to remove this reservation or not.

3.2.3.4 Outputs

There is no data output to this function. A message is raised to inform the user that the operation is a success.

3.2.3.5 Error Handling

An unexpected error will stop the function and raise an error message.

3.2.4 Check if a room is reserved

3.2.4.1 Introduction

This function allows a user to check if a room is reserved on a specific time slot.

3.2.4.2 Inputs

It takes as input a room and a time slot.

3.2.4.3 Processing

Room must have a boolean isReserved parameter that is associated with a time slot, the function simply checks this parameter.

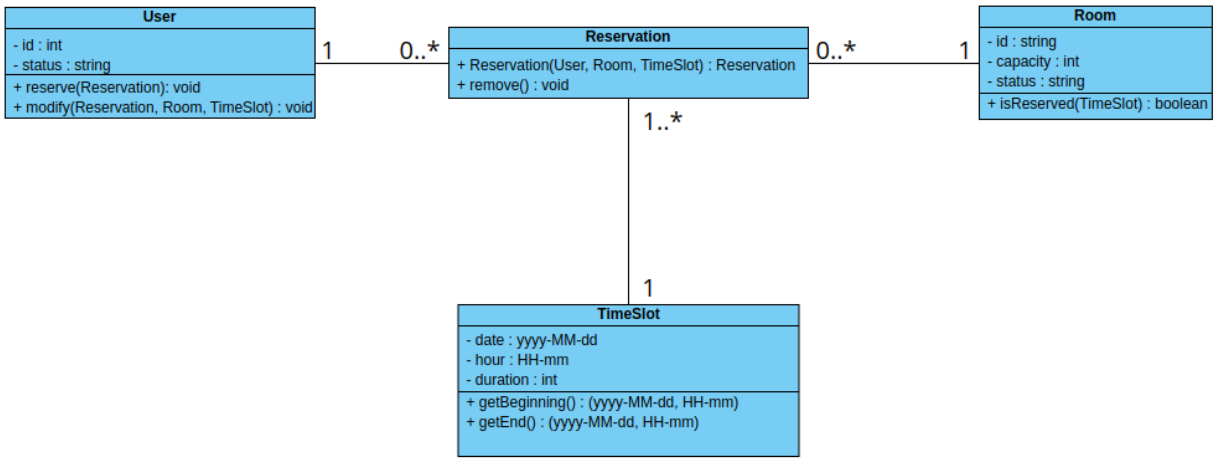
3.2.4.4 Outputs

The output is the value of the isReserved parameter.

3.2.4.5 Error Handling

An unexpected error will raise an error message.

3.4 Classes / Objects



3.4.1 User

3.4.1.1 Attributes

- id : int, refers to a unique id which is associated with the identity of the user.
- status : string, refers to the position of the user in the university (student, teacher, staff …)
- reservations : ReservationList, refers to the list of reservations that a user has.

3.4.1.2 Functions

- reserve(Reservation) : void, this function creates a reservation. (Refers to 3.2.1)
- modify(Reservation, Room, TimeSlot) : void, this function modify a reservation with the new Room and TimeSlot parameters (Refers to 3.2.2)

3.4.2 Room

3.4.2.1 Attributes

- id : string, refers to the room's location (for example, the 11th room of the H building 4th floor can be thoroughly identified by the string H411)
- capacity : int, refers to maximum amount of people that can be in the room
- status : string, refers to what kind of room it is
- reservations : ReservationList, refers to the list of reservations associated with this room

3.4.2.2 Functions

- isReserved(timeSlot) : boolean, indicates if a room is reserved on a specific time slot. (Refers to 3.2.4)

3.4.3 TimeSlot

3.4.3.1 Attributes

- date : yyyy-MM-dd, a date format, referring to the date of the time slot
- hour : HH-mm, a time format without the second (we consider that a time slot starts at the beginning of a minute, so the seconds will always be 00). It refers to the beginning time of the time slot.
- duration : int, number of minutes of the time slot
- reservations : ReservationList, refers to the list of reservations associated with this time slot.

3.4.3.2 Functions

- getBeginning() : (yyyy-MM-dd, HH-mm), return the date and time corresponding to the beginning of the time slot. (Refers to 3.2.4)
- getEnd() : (yyyy-MM-dd, HH-mm), return the date and time corresponding to the end of the time slot. (Refers to 3.2.4)

3.4.4 Reservation

3.4.4.1 Attributes

- user : User, refers to the user who made the reservation
- room : Room, refers to the room that is reserved
- timeSlot: TimeSlot, refers to the time slot allocated to the reservation

3.4.4.2 Functions

- Reservation(User, Room, TimeSlot) : Reservation. It is the constructor of the class Reservation. (Refers to 3.2.1 and 3.2.2)
- remove() : void, remove a reservation (refers to 3.2.3)

3.5 Non-Functional Requirements

3.5.1 Performance

This software must be able to handle databases containing a maximum of 10,000 rooms and 100,000 users. As the databases may contain heavy files such as photos of rooms, we consider that the software must be able to deal with 10 To databases.
Moreover, a minimum of 5000 simultaneous connections has to be possible without any interference.

3.5.2 Reliability

As a room must not be reserved by two users at the same time, the reservation function must process in less than 5 seconds. We consider that a reservation of a room for the same time slot won’t happen in less than 5 seconds.

3.5.3 Availability

The maintenance of this software must not last more than 24 hours.

3.5.4 Security

As the room reservation software will be deployed on the university servers, we consider that the security concerns are already well managed. It means that the hardware and firmware are up to date regarding known security breaches and that the databases are handled in a secure way.

3.5.5 Maintainability

This software must stay maintainable both in terms of performances: efficient hardware must be available to increase performances if estimations are exceeded.

3.5.6 Portability

The software must work properly on Windows, Linux, MacOs and Android.

3.6 Design Constraints

As the software will be deployed on the university's servers, it must follow university policies. Moreover, the hardware standards will be defined regarding the university facilities.

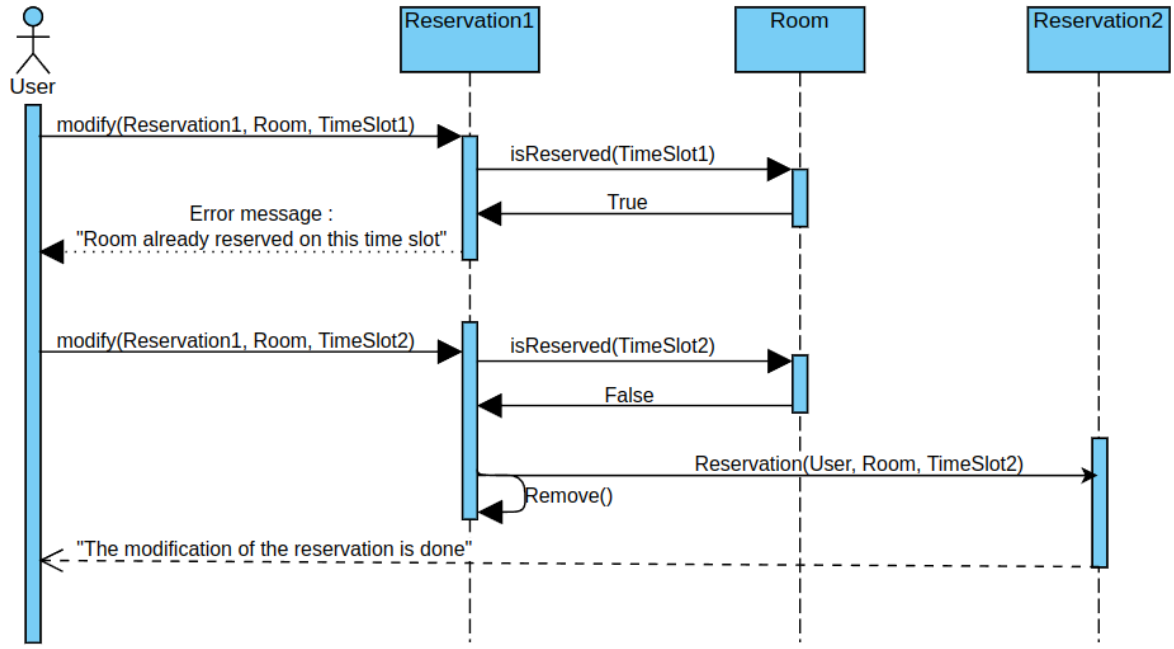
3.7 Logical Database Requirements

As specified in 2.5, the software will use several databases that are already existing in the university's servers. The logical database requirements are defined regarding the current properties of the databases and can change as the database is modified.

4. Analysis Models

4.1 Sequence Diagrams

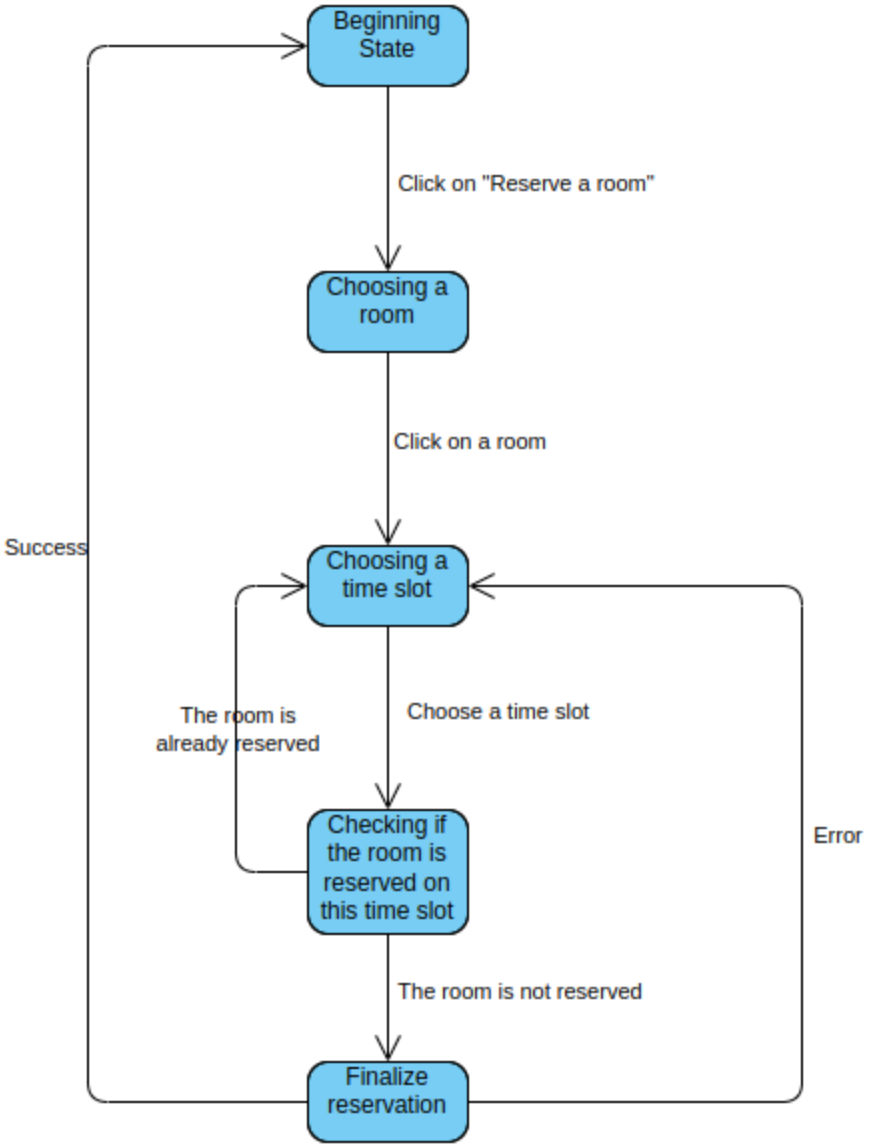
This diagram represents the case of reservation's modification from a regular user.



The first case with TimeSlot1 is an error case where the room Room is already reserved on the time spot TimeSpot1, so the user gets an error message.
Then the user tries with another time slot, this time it works. The current reservation is removed and the new reservation is created.

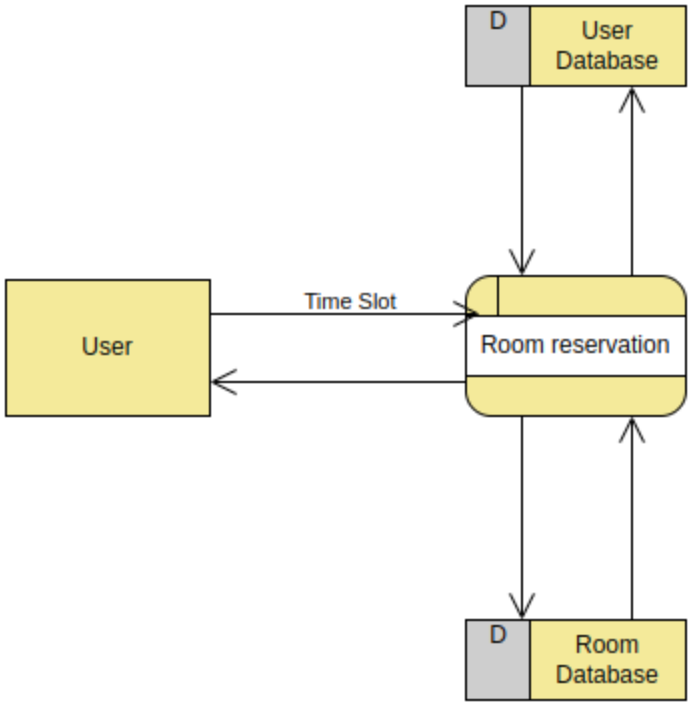
4.2 State-Transition Diagram (STD)

This diagram represents the case of a regular reserving a room from the user interface. It takes into account errors and if the room is already reserved on the wanted time slot



4.3 Data Flow Diagrams (DFD)

This diagram represents the data flow during the room reservation process.
The functions are requesting both the user database and the room database to see if every rule is verified.



5. Change Management Process

The Change Management process for updating the Software Requirement Specification (SRS) involves a structured approach to handle changes in project scope or requirements. Any project stakeholder can submit proposed changes through a formal change request process. These requests are evaluated by a Change Control Board (CCB), comprising project managers, technical leads, and stakeholders. The CCB assesses the feasibility, urgency, and impact of proposed changes, approving or rejecting them accordingly. Approved changes are prioritized and implemented by the development team, with updates made to the SRS document. The updated SRS undergoes review to ensure accuracy before final approval and communication to stakeholders. This process ensures that the SRS remains aligned with project objectives and evolving requirements.

The change management process outlined below will facilitate the timely and effective updating of the SRS:

Change Identification:

Any stakeholder involved in the project, including developers, project managers, users, or clients, can submit proposed changes to the SRS.
Changes can be identified through various means such as feedback from users, evolving business requirements, technological advancements, or regulatory updates.

Change Submission:

Proposed changes to the SRS should be submitted through a formal change request process.
Change requests should include detailed information about the proposed modification, rationale for the change, and its potential impact on the project.

Change Review and Evaluation:

A designated change control board (CCB) consisting of project managers, technical leads, domain experts, and stakeholders will review and evaluate each change request.
The CCB will assess the feasibility, urgency, and implications of the proposed changes.
Evaluation criteria may include factors such as impact on project timeline, budget, resources, and alignment with project objectives.

Change Approval:

The CCB will approve or reject change requests based on their assessment of the proposed changes.
Approved changes will be prioritized and scheduled for implementation based on their urgency and importance.
Rejected changes will be communicated back to the requester with explanations for the decision.

Change Implementation:

Once a change is approved, it will be implemented by the development team according to the established procedures.
Developers will update the SRS document to reflect the approved changes accurately.
Changes should be implemented in a controlled manner to minimize disruption to ongoing development activities.