# Software Requirements Specification

Version 1.0

For

Concordia Capstone Scheduler

Prepared by

| Name | StudentID | Email |
|---|---|---|
| Lance Lafontaine | 26349188 | lance.lafontaine92@gmail.com |
| Jason Tsalikis | 25892120 | jtsalikis@hotmail.ca |
| Lenz Petion | 26775837 | lenzpetion@gmail.com |
| Rameen Rastan-Vadiveloo | 27191863 | rameenrastanv@hotmail.com |
| Benny Zhao | 27205104 | bennyzhao@live.ca |
| Simeon Cvetkovic | 27430515 | bitugos@gmail.com |
| Lance Lafontaine | 26349188 | lance.lafontaine92@gmail.com |
| Sam Moosavi | 27185731 | sammoosavi94@gmail.com |

| | |
|---|---|
| Instructor | Professor Constantinos Constantinides |
| Course | SOEN 343<br>Software Architecture and Design I |
| Date: | November, 23, 2016 |

Document History

| Date | Version | Description | Authors |
|---|---|---|---|
| Saturday October 8th 2016 | 0.1 | Initial template and structure of document | All |
| Saturday October 23rd | 0.2 | Refine requirements, domain model, use cases, contracts | - Simeon<br>- Lance<br>- Rameen<br>- Zhipeng<br>- Sam |
| Sunday, November, 13th | 0.3 | Writing body text, contracts, diagrams | - Jason<br>- Rameen<br>- Sam<br>- Lenz<br>- Benny |
| Saturday, November, 19th | 0.4 | Continuing to write body text | - Rameen<br>- Sam<br>- Jason<br>- Simeon<br>- Lance |
| Sunday, November, 20th | 0.5 | Finalizing diagrams, use cases, contracts, body text | - Rameen<br>- Sam<br>- Jason<br>- Benny |
| Wednesday, November, 22nd | 1.0 | Formatting and editing | - Rameen<br>- Sam<br>- Jason<br>- Lance |

## Table of contents

**List of figures**

**List of tables**

# INTRODUCTION

## Introduction

The essence of the project is a web application that displays the availabilities of rooms that students can use for academic activities, and allows users to reserve those rooms for a specific time slot. In this document, the goal is to understand and document the requirements of the system and ensure that they are feasible and consistent with each other. The purpose and scope are clearly defined; the description of constraints and assumptions of the project are noted. In the specific requirements section, both the functional and nonfunctional requirements are documented. A use case model and domain model are provided, from which use cases are derived. Critical use cases are identified, their respective system sequence diagrams are shown, with the system operations, the operation contracts followed the interaction diagrams.

## Purpose

The SRS is a contract between developers and stakeholders; given such it is meant to be read by both stakeholders and developers so that both parties can validate and agree upon all requirements and specifications. The structure of the document is such that it firstly describes what is feasible and the constraints that have been set on the system. Following that, the requirements for the system, both functional and nonfunctional, are detailed. The use case model as well as the use cases and the domain model are then presented. The use cases tell the story of how our system behaves in response to an actor’s actions. Stakeholder can verify these "stories" in order to determine if they properly align with their requirements and vision of the system.

## Scope

The project allows a user to log into the webpage. A schedule appears with a list of rooms; the user can view the schedule for different rooms and see available timeslots. The user can thus book a timeslot for a room or if desired timeslot is taken can go on the waitlist. The goals essentially is to allow users find a room that is free and book a timeslot, and should it all be taken go on the waitlist. This will allow students to organize meetings with far more ease.

## Definitions, acronyms, and abbreviations

Table 1 . Glossary

| Term | Definition |
|---|---|
| User | Someone who interacts with the reservation system. |
| Registry | A system for keeping an official record of room reservations. |
| Room | A space that students occupy for academic activities |
| Wait list | An ordered queue of students with a pending reservation at an unavailable time slot. |
| Reservation | Occupying a study room at a given time slot |
| Time Slot | A 1-hour time interval. |
| Weekly Calendar | A table displaying room information (time slots and availabilities) for a given week. |
| Stakeholder | An individual or group who is affected by the development of this project. |
| SRS | Software Requirements Specification |
| SAD | Software Architecture Documentation |
| Python | Python is a high-level, general-purpose programming language |
| SSD | System Sequence Diagram |
| HTML | Hypertext Markup Language |
| CSS | Cascading Style Sheet |

# OVERALL DESCRIPTION

## Product perspective

The product is available to all Engineering students at Concordia University. It is a web based application that uses a layered architecture as well as a client server architecture. It is compatible with all the major browsers. The system allows the user to login to their accounts and provides functionality for updating and viewing their reservations. The system can support a small to medium number of users as it is hosted through a computer.

## Product functions

The major system functions provided by the product to a user include making a new reservation for an available room at a desired time slot, entering the waiting list for a booked room at a desired time slot, deleting an existing room reservation, modifying an existing reservation, and viewing a list of their current reservations.

## User characteristics

The intended users of this product are students studying in the Engineering department of Concordia University. The user is not expected to have a particularly high level of experience or technical expertise in software systems, they are merely are expected to be at a University educational level.

## Constraints

The system must promote fairness by limiting the number of reservations a given user can occupy at a time. The system must also incorporate a level of safety and security, by ensuring a user's credentials and information cannot be accessed by other users of the system, and also that a user's information, such as their current reservations or waiting list status, cannot be overwritten by another user. If two or more users make a reservation for an available room simultaneously, the system must award the reservation to one user, and provide an error message to the other user(s). This constraints applies to multiple users in real-time and fulfills the system's requirement for concurrency. This concurrency is handled by our relational database management system (RDBMS) by automatically monitoring and completing transactions according to ACID standards.

### Assumptions and dependencies

As a web application, the system's main dependency will be a web browser. It is assumed that the system will be compatible with all major operating systems (Windows, Linux and macOS). It assumes that rooms will be available for reservation on any day of the week, including weekends, with the exception of holidays recognized by the University.

# SPECIFIC REQUIREMENTS

### External interfaces

1- Input: Username and Password (login credentials)
Output: Error message if the user isn't found in the database

2- Input: Make room reservation (Userame, Room Number, Time slot)
Output: Success message if reservation was made, error message if not.

3- Input: Room Number
Output: Weekly Calendar showing the room information (time slot statuses)

4- Input: Modify room reservation (Username, Old Room Number, New Room Number, Old Time Slot, New Time Slot)
Output: Success message if modification is successful, error message if not.

5- Input: Cancel room reservation (Username, Time Slot, Room Number)
Output: Success message if cancellation is successful, error message if not.

## Functionality

### Functional Requirements

1.1 The system shall permit a user to book a room reservation at a given time slot.
1.2 The system shall permit a user to modify an existing room reservation to a new time slot.
1.3 The system shall permit a user to cancel an existing room reservation.
1.4 The system shall permit a user to view a weekly calendar of all reservation availabilities for a room.
1.5 The system shall permit a user to enter the wait list for a reservation at an unavailable time slot.
1.6 The system shall prevent a user to make more than 3 reservations per week.
1.7 The system shall remove a user from all other his/her waiting lists on any other room if he/she successfully obtains a reservation for the same date and time.

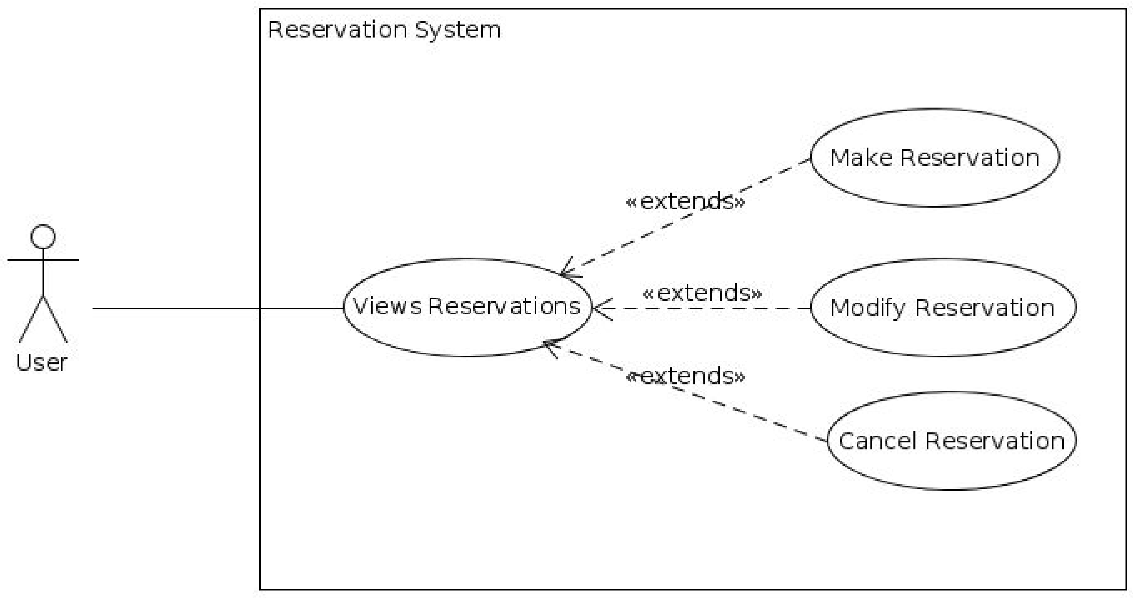
### Actor goal list

Table 2. Actor goal list

| | |
|---|---|
| Actor : User | Goal : View reservations |
| Actor : User | Goal : Make reservation |
| Actor : User | Goal : Cancel reservation |
| Actor : User | Goal : Modify reservation |

**Use case view**

Figure 1. Use case model



**Accessibility**

Only ENCS registered students in the database should have access to the web application.

**Usability**

90% of users will be able to perform all major activities provided by the system in less than 3 minutes.

**Portability**

The web application should function on all major operating systems.

**Design constraints**

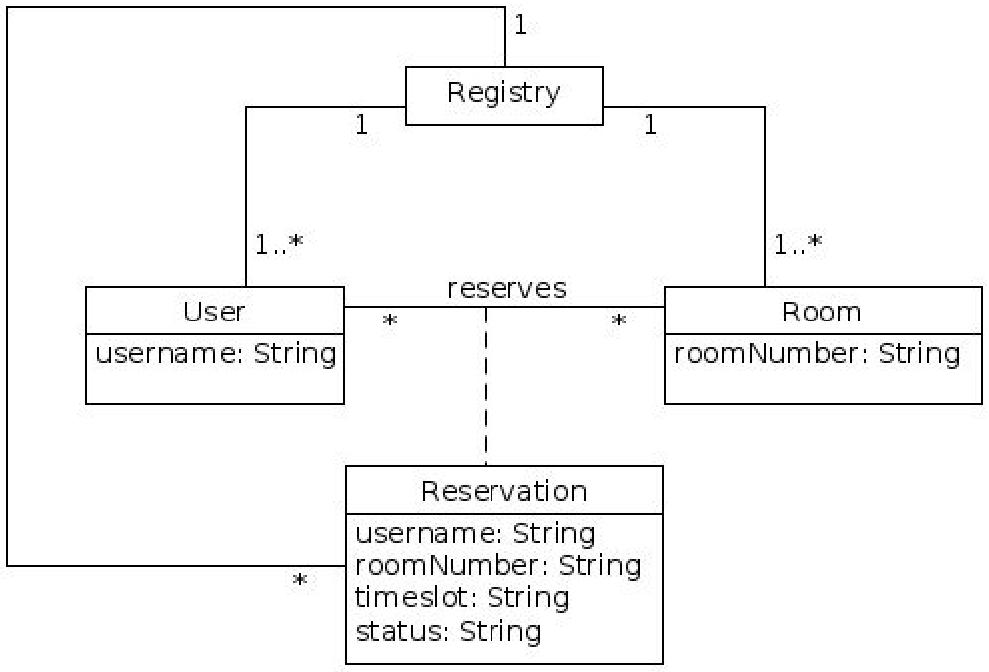
-Languages used: HTML, CSS JavaScript, Python, SQL

-Web Framework used: Django

# ANALYSIS MODELS

Figure 2. Domain model

**Domain Model**



## Use Cases

Use case UC1: Make Reservation
Level: user-goal
Primary Actor: User
Preconditions:

- User is authenticated in system

Success Guarantee (Postconditions):

- User has booked a room

Main Success Scenario (or basic flow):

1. User chooses the option to make a reservation.
2. System prompts for a room and timeslot
3. User selects a room and timeslot.
4. System creates a reservation and cancels all other User's reservations that are on a waiting list for the same timeslot in all rooms.
5. System returns a confirmation message.

Extensions (or alternative flows):

4a. Timeslot is already reserved by another user
4a.1 System creates a reservation and place it on the waiting list for that timeslot in the selected room.
4b. User has exceeded the number of allowed reservations for that week.
4b.1 System returns an error message.
4c. User already has a reservation for that timeslot.
4c.1 System returns an error message.

Use case UC2: View Reservation
Primary Actor: User
Level: user-goal
Stakeholders and Interests:
Preconditions:

- User is authenticated in system

Success Guarantee (Postconditions):

- User can view a room’s weekly calendar

Main Success Scenario (or basic flow):

1. User selects room and week.
2. System displays all weekly reservations for the selected room.

Use case UC3: Modify Reservation
Level: user-goal
Primary Actor: User
Preconditions:

- User is authenticated in system
- User is authenticated. User has a reservation.

Success Guarantee (Postconditions):

- User reserved a room for a timeslot.

Main Success Scenario (or basic flow):

1. User chooses the option to modify a reservation.
2. System prompts for a room, week and time slot.
3. User selects a room, week and time slot.
4. System modifies the reservation and cancels all other User's reservations that are on a waiting list for the same timeslot in all rooms.
5. System returns a confirmation message.

Extensions (or alternative flows):

4a. Timeslot is already reserved by another user

4a.1 System creates a reservation and place it on the waiting list for that timeslot in the select room.

4b. User already has a reservation for that timeslot.

4b.1 System returns an error message.

Use case UC4: Cancel Reservation
Level: user-goal
Primary Actor: User
Preconditions:

- User is authenticated in system
- The user has an existing reservation

Success Guarantee (Postconditions):

- User has cancelled his reservation

Main Success Scenario (or basic flow):

1. User chooses the option to cancel a reservation.
2. System prompts for a reservation.
3. User selects one of his/her reservations.
4. System cancels the reservation.

Use case UC5: Log in
Level : subfunction
Primary Actor: User
Preconditions:

- User is registered in the system

Success Guarantee (Postconditions):

- User is identified and authenticated

Main Success Scenario (or basic flow):

1. User enters credentials
2. System confirms authentication

Extensions (or alternative flows):

2a. Invalid username or password

2a.1 System returns error message

Use case UC5: Log out
Level: subfunction
Primary Actor: User
Stakeholders and Interests:
Preconditions:

- User is authenticated in system

Success Guarantee (Postconditions):

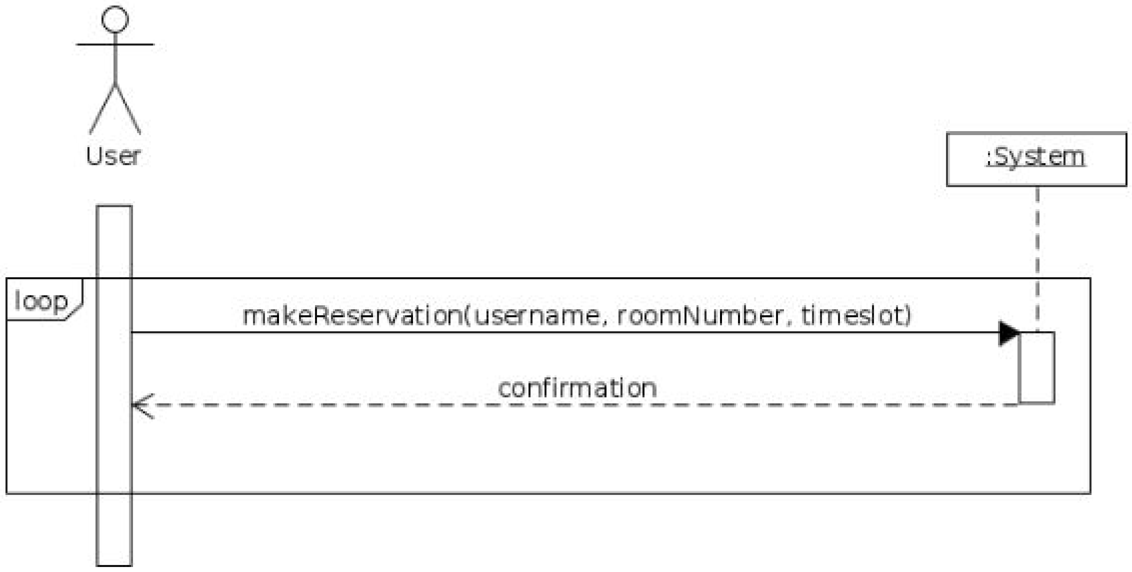
- User is logged out

Main Success Scenario (or basic flow):

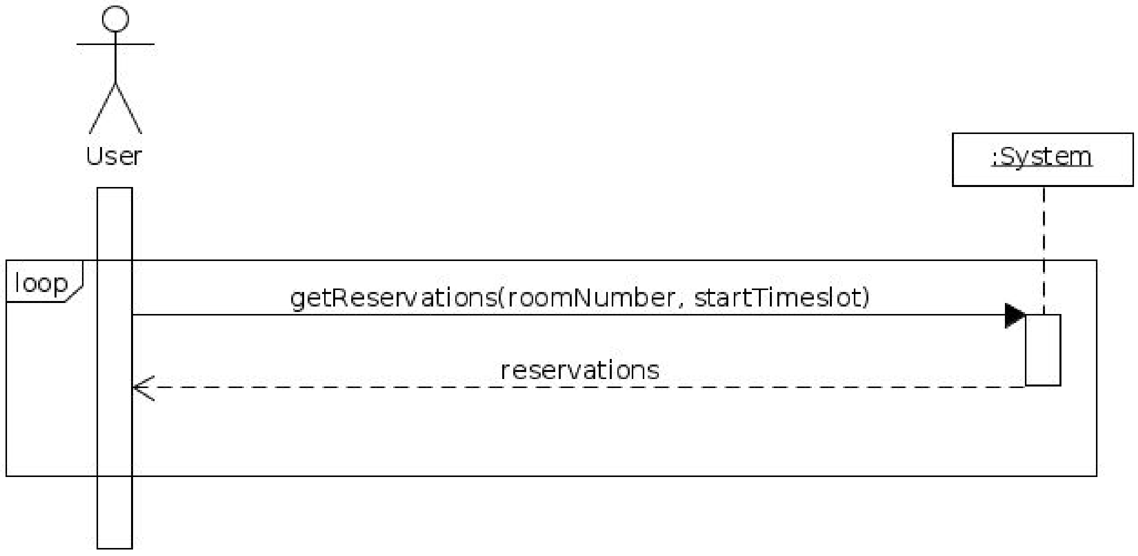
1. User indicates that he/she wants to log out
2. System logs out User

## System Sequence Diagrams

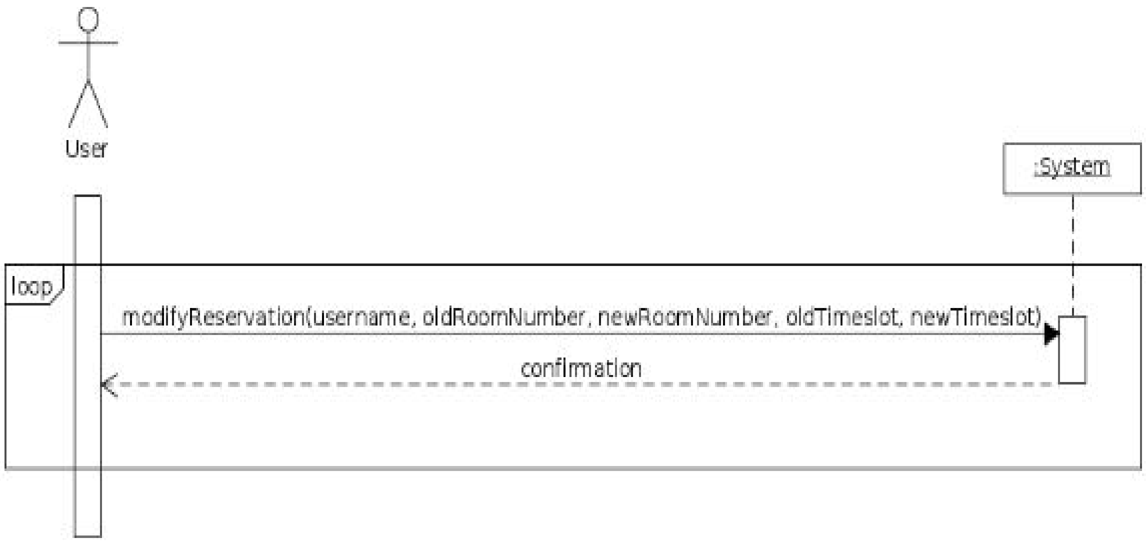
Use Case UC1: Make Reservation
Figure 3. Make Reservation SSD



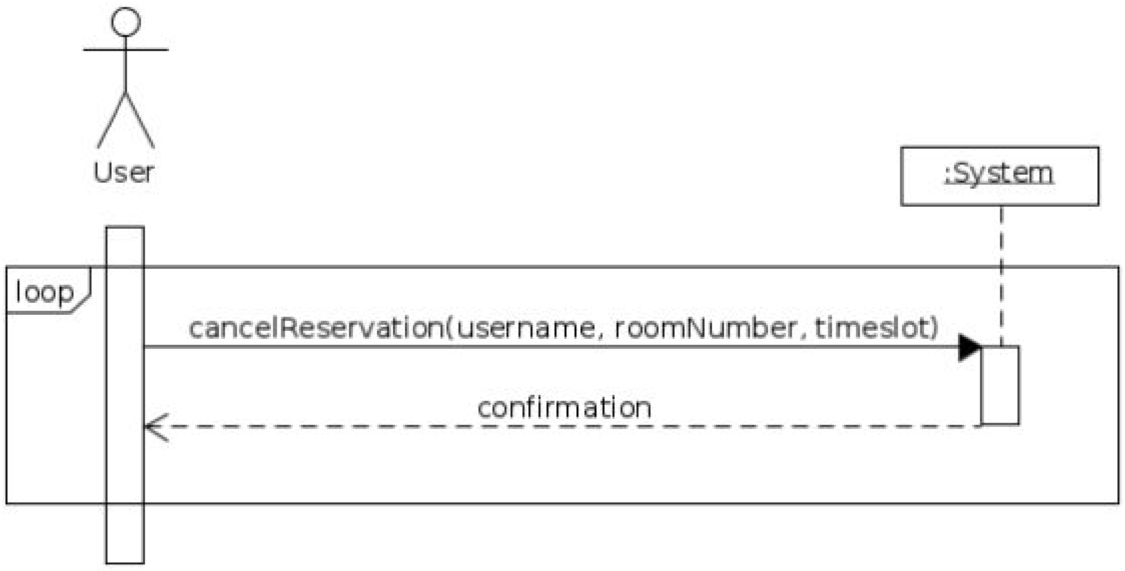
Use Case UC2 : View Reservation
Figure 4. View Reservation SSD



Use Case UC3: Modify Reservation
Figure 5. Modify Reservation SSD


User
:System
loop
modifyReservation(username, oldRoomNumber, newRoomNumber, oldTimeslot, newTimeslot)
confirmation

Use Case UC4: Cancel Reservation
Figure 6. Cancel Reservation SSD



## System Operations

makeReservation(username, roomNumber, timeslot)

getReservations(roomNumber, startTimeslot)

cancelReservation(username, roomNumber, timeslot)

modifyReservation(username, oldRoomNumber, newRoomNumber, oldTimeslot, newTimeslot)

## Operation Contracts

Contract C01: Make Reservation
Operation: makeReservation(username, roomNumber, timeslot)
Cross Reference : UC1 Make Reservation
Preconditions:
- None

Postconditions:
- `A Reservation instance reserv was created. (instance created)`
- `reserv was associated with Room. (association formed)`
- `reserv was associated with User. (association formed)`

Contract C02: View Reservation
Operation: viewReservations(roomNumber, starTimeslot)
Cross Reference : UC2 View Reservation
Preconditions:
- None

Postconditions:
- None

Contract C03: Modify Reservation
Operation: `modifyReservation(username, oldRoomNumber, newRoomNumber, oldTimeslot, newTimeslot)`
Cross Reference : UC3 Modify Reservation
Preconditions:

- A Reservation reserv has been created and is associated to a User (user) and a Room (oldRoom).

Postconditions:

- Reservation.timeInterval was set to newTimeslot (attribute modification)
- Reservation.roomNumber was set to newRoomNumber (attribute modification)
- Reservation was association with a Room (newRoom)
- Association between oldRoom and reserv has been broken.

Contract C04: Cancel Reservation
Operation: cancelReservation(userId, roomNumber, timeslot)
Cross Reference : UC4 Cancel Reservation
Preconditions:

- A Reservation has been created.

Postconditions:

- A Reservation instance was deleted. (instance deletion)