Software Architecture Document

Version 1.1

for

LOTUS Calendar

Prepared by

Saif Mahabub	27392974	rhythmsaif@gmail.com
Alexander Rosser	27543069	arosser95@gmail.com
Philippe Kuret	27392680	philippekuret@gmail.com
An Ran Chen	27277385	archen94@gmail.com
Adriel Fabella	27466005	adriel.fab@gmail.com
Costa Papadakos	26665691	cotsop@gmail.com

Instructor: Dr. C. Constantinides

Course: SOEN 343

Date: 23/11/2016

Document history

Date	Version	Description	Author
13/11/2016	1.0	Rough Draft	Saif Mahabub Philippe Kuret
22/11/2016	1.1	Revision	Alexander Rosser Saif Mahabub An Ran Chen Costa Papadakos Adriel Fabella

Table of contents

List of figures

`

1. Introduction

The following introduction provides an overview of the Software Architecture Document of the project.

Purpose

This document provides a detailed architectural overview of the LOTUS Calendar system with the use of a few different architectural views, in order to represent different aspects of the system. It is intended to represent the significant architectural decisions that are made on the system. This is viewed by stakeholders and the development team.

Scope

The scope of the document is to depict the architecture of the LOTUS Calendar system. It describes the architectural goals and constraints, the Use case view, Logical view and Data view.

2. Architectural representation

2.1 Logical view

Class diagram

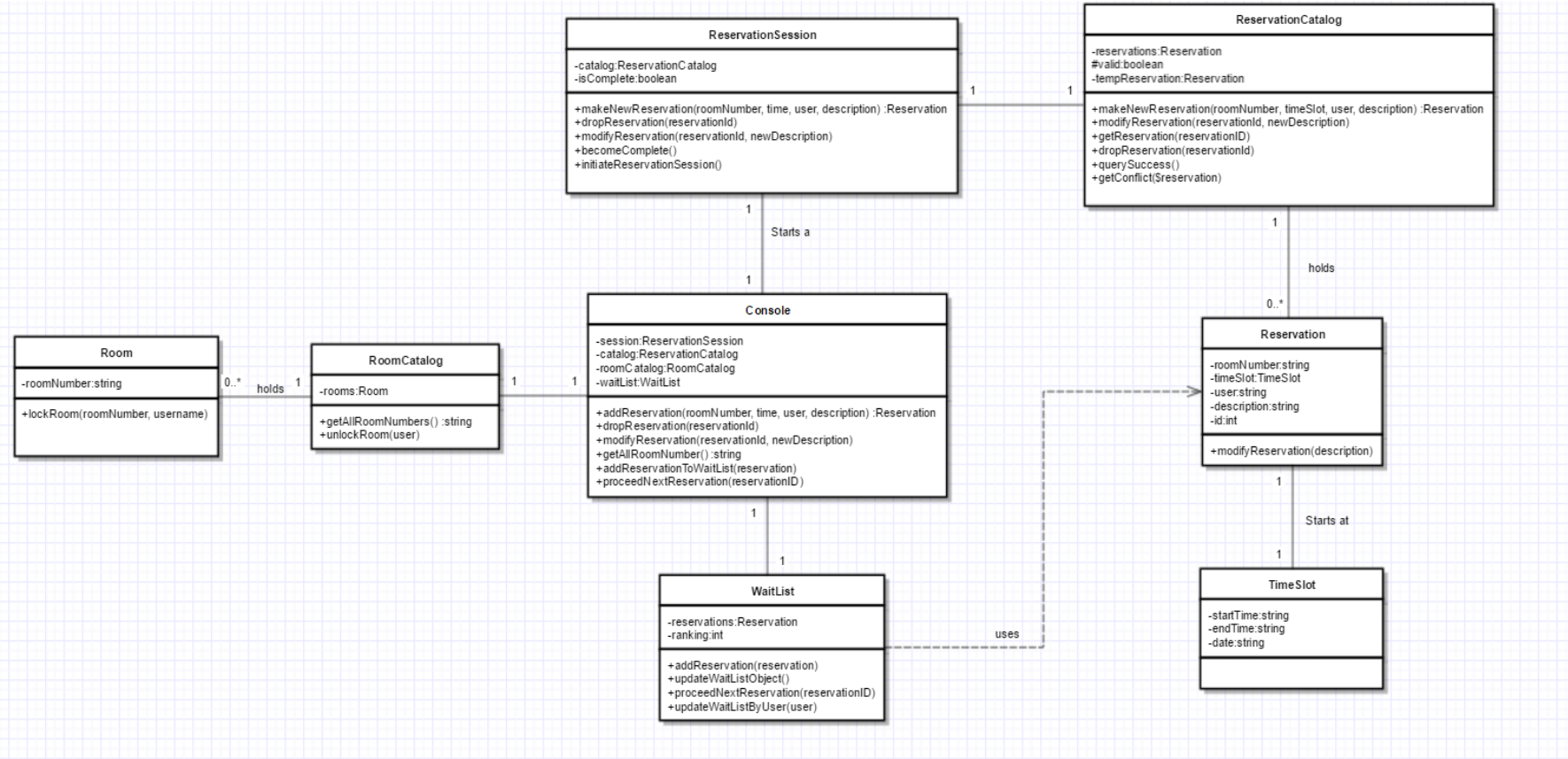

Figure 1: Class Diagram

System contract

Contract CO1	initiateReservationSession
Operation	initiateReservationSession(reservationCatalog)
Cross-Reference	UC8 – Create Reservation
Preconditions	• User is logged in • User is viewing the booking's schedule • Selected room is available to modify
Post-Conditions	1. An instance of ReservationSession ***rsession*** was created. 2. ***rsession*** was associated with Console 3. ***rsession*** was associated with ReservationCatalog

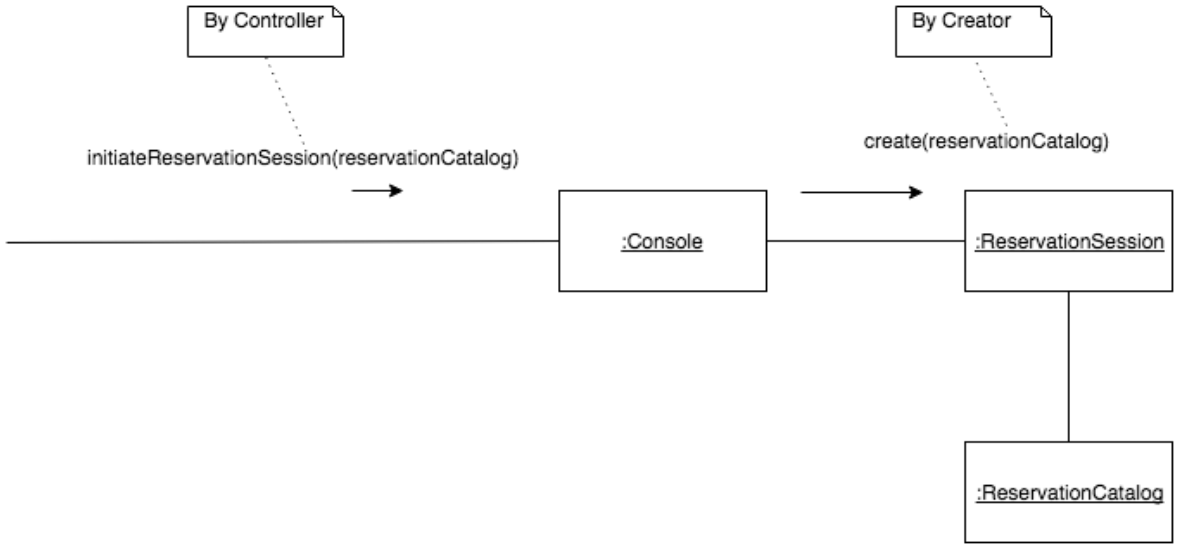

Figure 2: initiateReservationSession Communication Diagram

Contract CO2	addReservation
Operation	addRoom(room, timeslot, userID, description)
Cross-Reference	UC8 – Create Reservation
Preconditions	• Instance of ReservationSession is underway
Post-Conditions	1. An instance of Reservation ***res*** was created 2. ***res*** was associated with ReservationSession

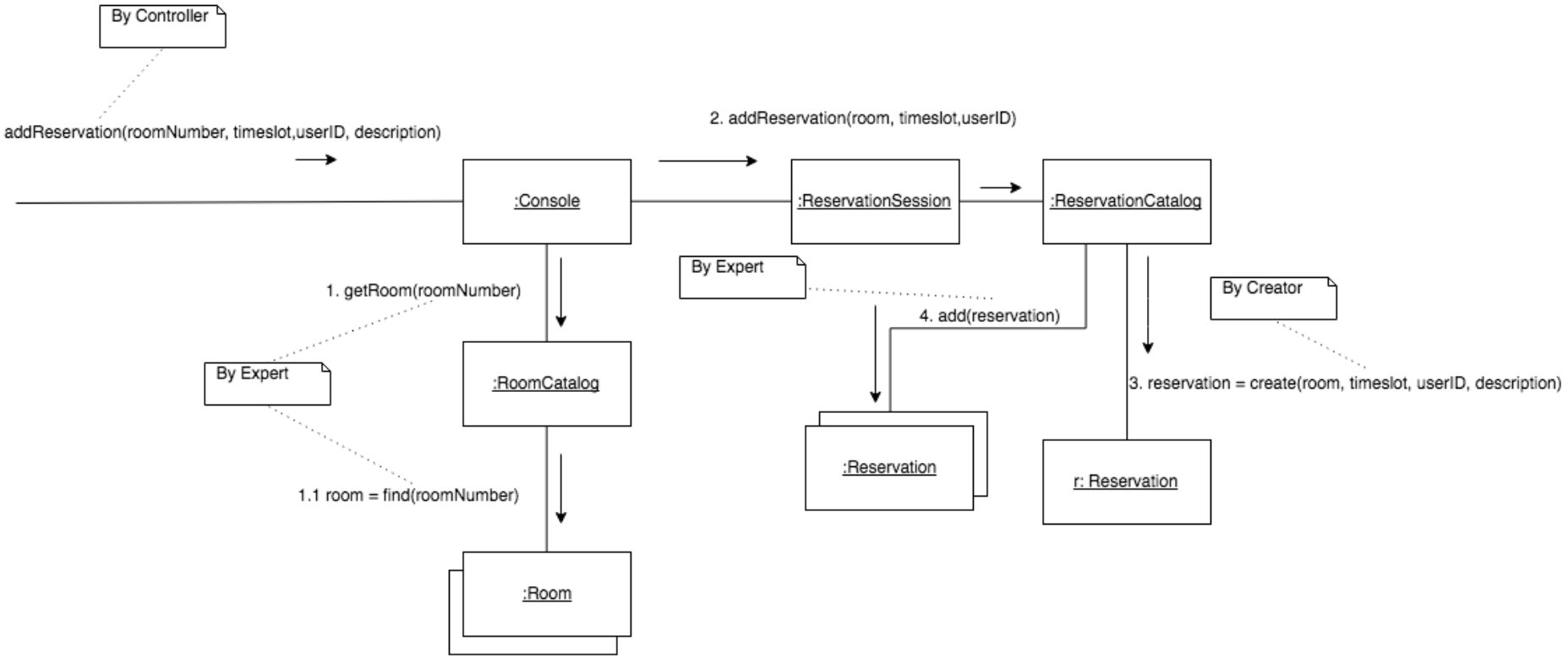

Figure 3: addReservation Communication Diagram

Contract CO3	endReservationSession
Operation	endReservationSession()
Cross-Reference	UC6 – Cancel Reservation
Preconditions	• Instance of ReservationSession is underway
Post-Conditions	1. ReservationSession.isComplete was set to true

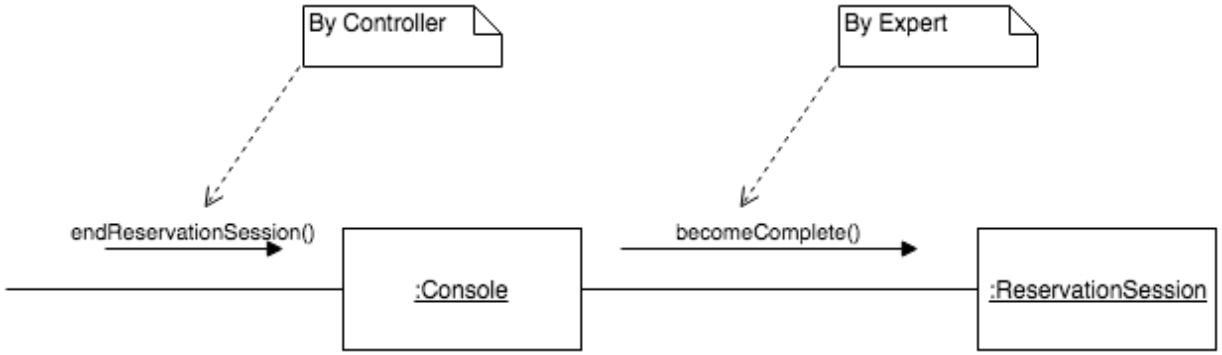

Figure 4: endReservationSession Communication Diagram

Contract CO3	addReservationToWaitList
Operation	addReservationToWaitList(reservation)
Cross-Reference	UC9 – Create Reservation
Preconditions	• Operation addReservation is under way
Post-Conditions	1. ***res*** was associated with Waitlist

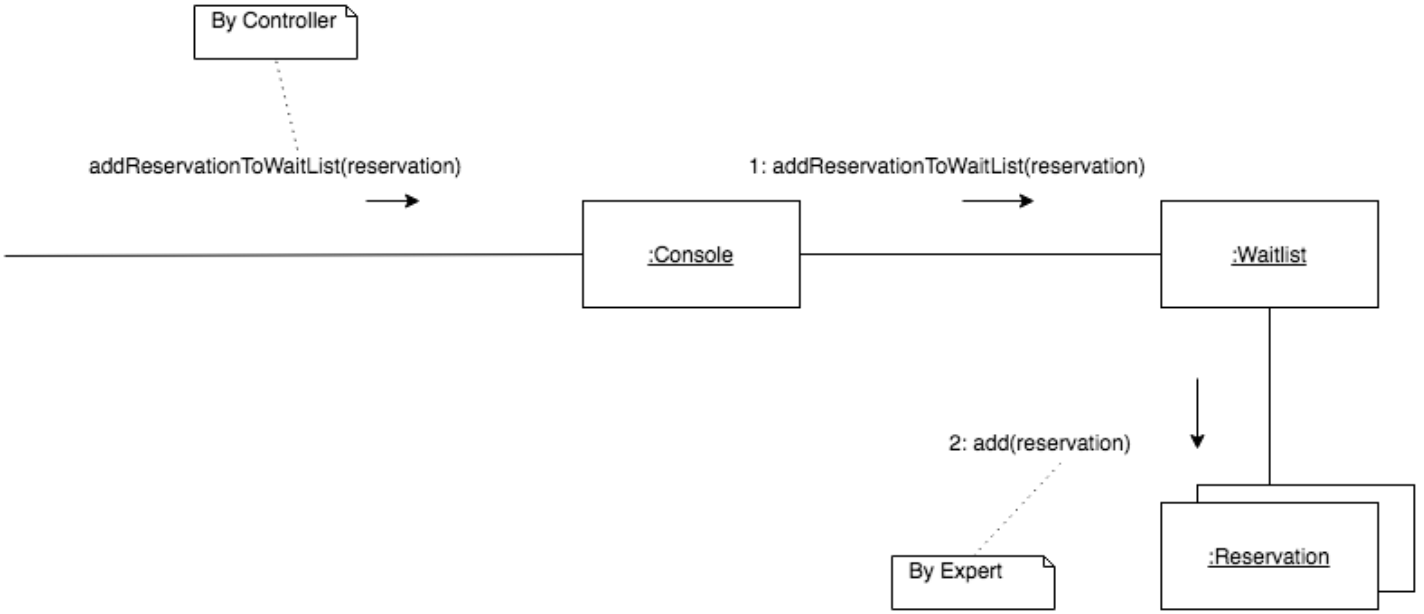

Figure 5: addReservationToWaitList Communication Diagram

Contract CO4	modifyReservation
Operation	modifyReservation(reservationID, description)
Cross-Reference	UC7 – Modify Reservation
Preconditions	• Instance of ReservationSession is underway
Post-Conditions	1. Reservation.description has been modified

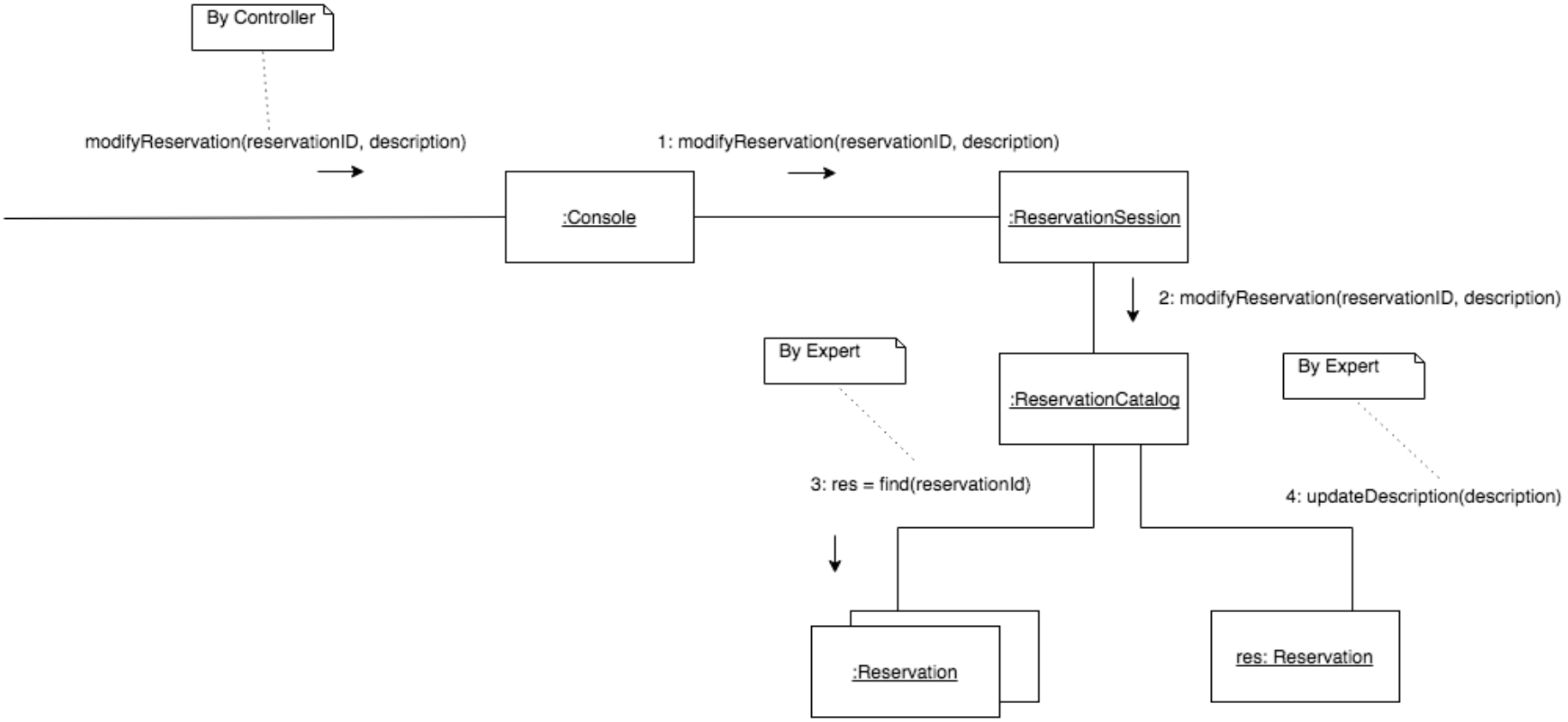

Figure 6: modifyReservation Communication Diagram

Contract CO5	dropReservation
Operation	dropReservation(reservationID)
Cross-Reference	UC9 – Drop Reservation
Preconditions	• Instance of ReservationSession is underway
Post-Conditions	- The multi-object of Reservation attribute in ReservationCatalog has been modified

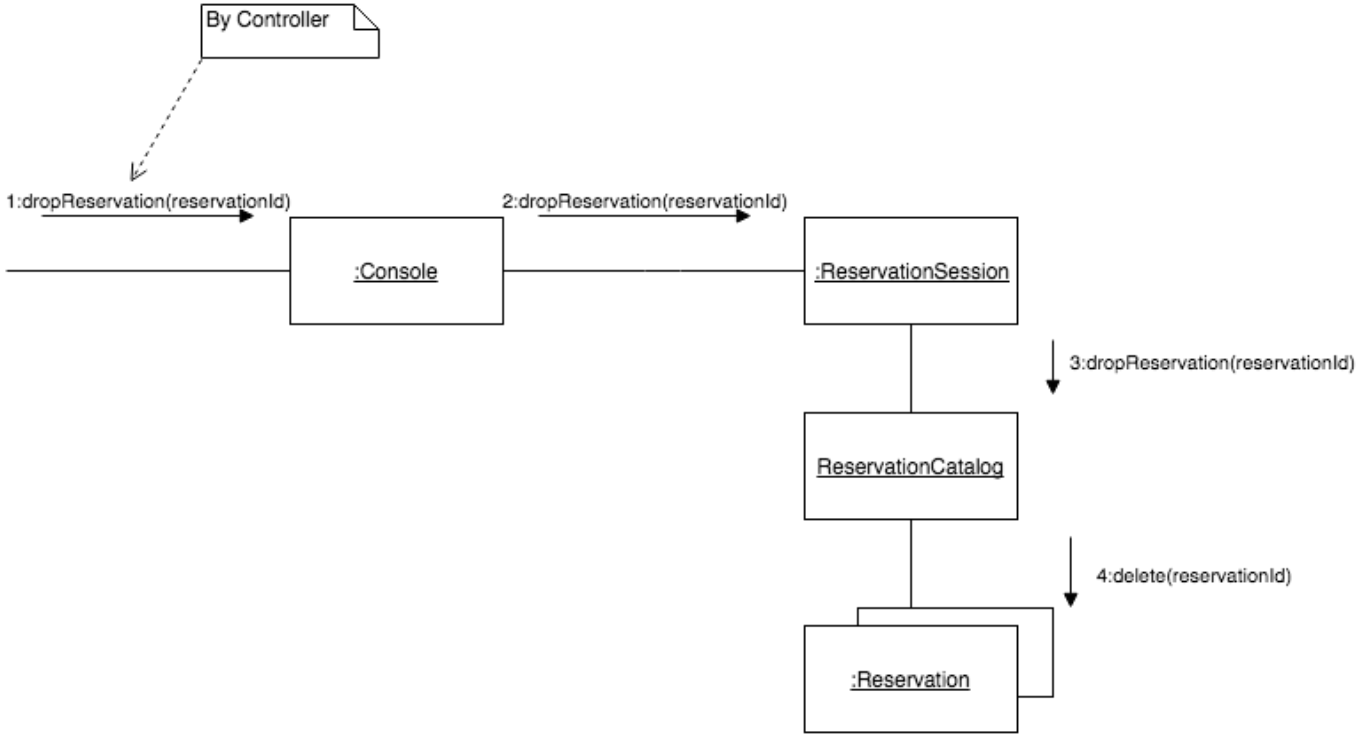

Figure 7: dropReservation Communication Diagram

2.2 Use case view

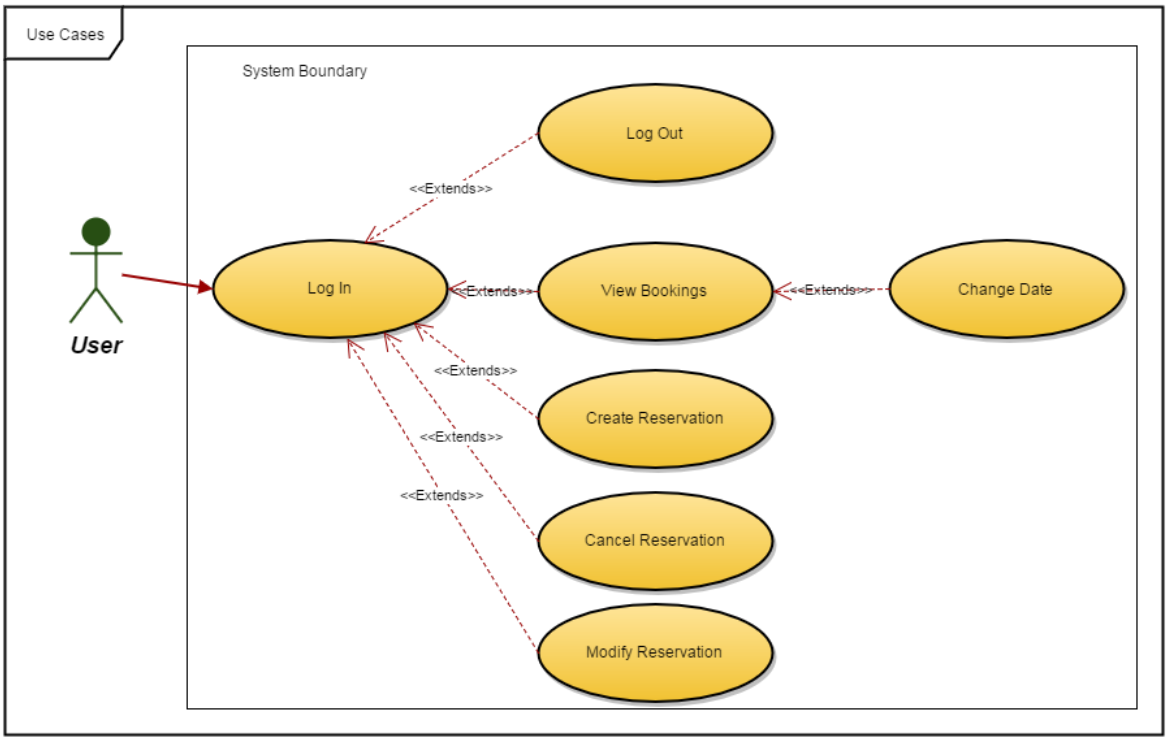

Figure 8: Use Case Diagram

2.3 Data view

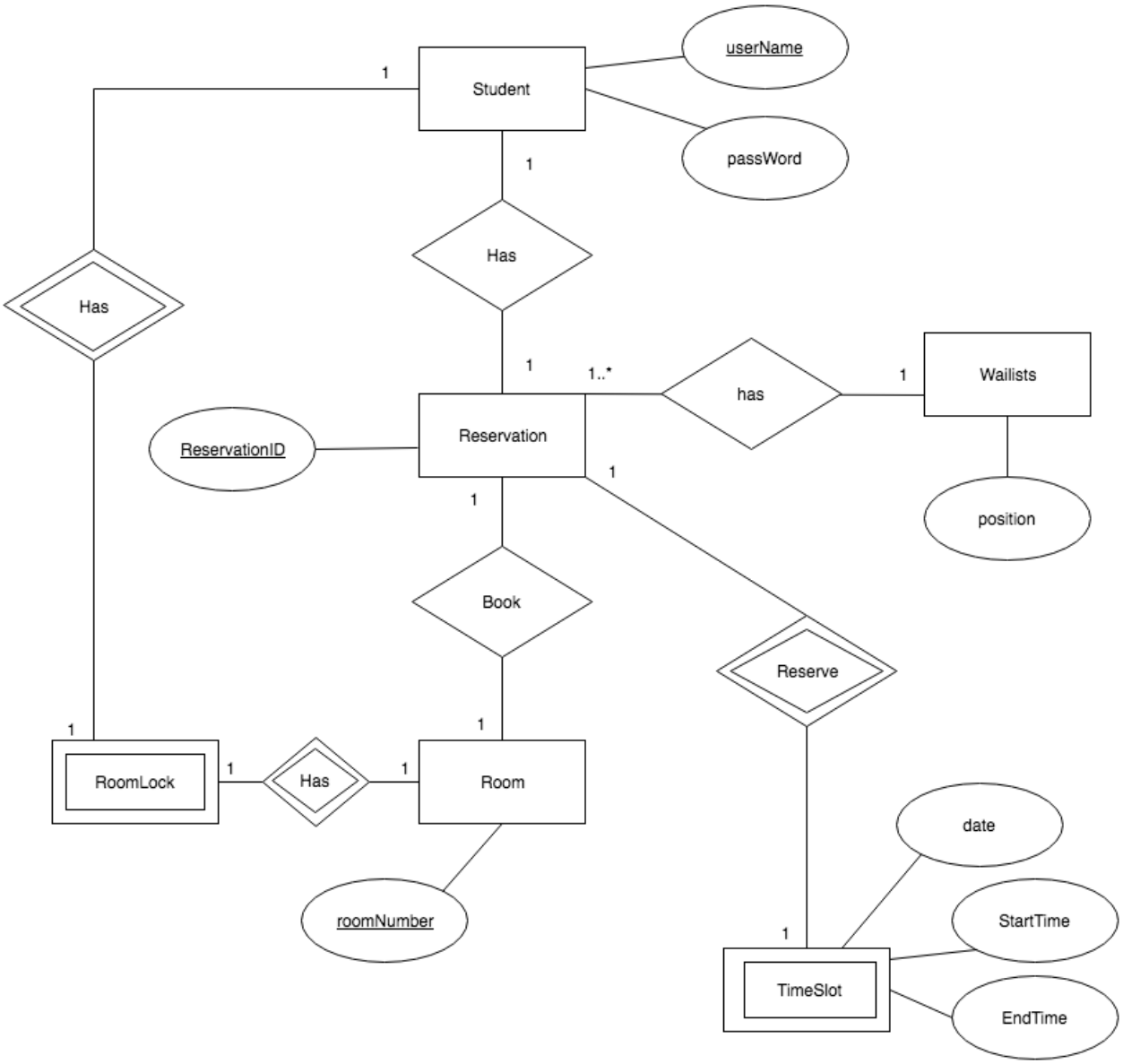

Figure 9: Entity-Relational Diagram

Relational schema for the Entity-Relational diagram

Student (<u>userName</u>, passWord)

Reservation (<u>ReservationID</u>, ***roomNumber***, ***userName***)

RoomCatalog (***roomNumber***)

TimeSlot (StartTime, EndTime, date, ***<u>ReservationID</u>***)

RoomLock (***<u>lockRoom,</u> userName***)

WaitList (***<u>ReservationID</u>***, position)

**Foreign keys are in bold. Primary keys are underlined.*

3. Architectural requirements: goals and constraints

Functional requirements (Use case view)

The system will allow the user to securely login.

The system will allow all users to see the reservations made in all room.

The system will allow the user to change the date while looking at the reservations.

The system will allow only one user per room to add, cancel and modify reservation their reservation.

The system will allow the user to make a maximum of 5 reservations.

If a user on a wait list acquires the reservation, he is removed from all other wait lists with identical time slot.

The overview below refers to architecturally relevant Use Cases from the Use Case Model (see references).

Source	Name	Architectural relevance	Addressed in:
UC1	Log in	Securely log the user into the system	2.2
UC03	View Bookings	View the reservations and availabilities schedule.	

UC04	Change View Booking Date	Change the viewed date.	
UC05	Create Reservation	Create a new reservation	
UC06	Cancel Reservation	Cancel a pre-existing reservation	
UC07	Modify Reservation	Modify a pre-existing reservation	

Non-functional requirements

Source	Name	Architectural relevance	Addressed in:
SRS	Authentication	The System shall provide system behavior only to users who enter valid credentials (usernames and passwords).	2.2
SRS	Student Database	The System shall refer to an external database in order to retrieve and verify the credentials of the Students	2.3
SRS	Reservation Database	The System shall use a MySQL database in order to store and retrieve all reservation information.	2.3