

Concordia University

Department of Computer Science

and Software Engineering

Software Architecture and Design II

SOEN 344 S --- 2017

Tyrell Corp – Chronos

Software Requirements Specification

|  |  |
| --- | --- |
| **Team members** | |
| **Name** | **Student ID** |
| Nicholas Burdet | 29613773 |
| Georges Mathieu | 26863477 |
| Olivier Cameron-Chevrier | 27228805 |
| Stefano Pace | 27454716 |
| Julian Ippolito | 27419112 |
| Adam Trudeau-Arcaro | 27459157 |
| Joey Tedeschi | 27513062 |

**Document history**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| March 21st, 2017 | 0.1 | Preliminary Editing of Sections 1, 2, 3.1, 3.3, 3.4 | Stefano Pace |
| March 22nd, 2017 | 0.2 | Preliminary Editing of Sections 3.2, 4 | Stefano Pace |

Table of Contents

[1. Introduction 1](#_Toc475542312)

[1.1 Purpose 1](#_Toc475542313)

[1.2 Scope 1](#_Toc475542314)

[1.2.1 User Login/Logout 1](#_Toc475542315)

[1.2.2 Mutual Exclusion 1](#_Toc475542316)

[1.2.3 Equipment 2](#_Toc475542317)

[1.2.4 Waiting list 2](#_Toc475542318)

[1.2.5 Recurring Reservations 3](#_Toc475542319)

[1.2.6 Maximum Reservations 3](#_Toc475542320)

[1.2.7 Capstone Students 3](#_Toc475542321)

[1.3 Glossary of Terms 3](#_Toc475542322)

[1.4 References 3](#_Toc475542323)

[2. Overall Description 4](#_Toc475542324)

[2.1 Product Perspective 4](#_Toc475542325)

[2.2 Product Functions 4](#_Toc475542326)

[2.3 User Characteristics 4](#_Toc475542327)

[2.4 Constraints 4](#_Toc475542328)

[2.5 Assumptions and Dependencies 4](#_Toc475542329)

[3. Specific Requirements 5](#_Toc475542330)

[3.1 External Interfaces documentation 5](#_Toc475542331)

[3.2 Functionality documentation 5](#_Toc475542332)

[3.2.1 Actor Goal List documentation 5](#_Toc475542333)

[3.2.2 Use Case View documentation 6](#_Toc475542334)

[3.2.3 Use Case Descriptions 7](#_Toc475542335)

[3.3 Nonfunctional Requirements 11](#_Toc475542336)

[3.3.1 Reliability documentation 11](#_Toc475542337)

[3.3.2 Usability documentation 11](#_Toc475542338)

[3.3.3 Efficiency documentation 11](#_Toc475542339)

[3.3.4 Maintainability documentation 11](#_Toc475542340)

[3.3.5 Portability documentation 12](#_Toc475542341)

[3.3.6 Design Constraints documentation 12](#_Toc475542342)

[3.3.7 User Documentation and Help documentation 12](#_Toc475542343)

[3.4 Purchased Components documentation 12](#_Toc475542344)

[3.4.1 Web Server 12](#_Toc475542345)

[3.4.2 Domain Name 12](#_Toc475542346)

[4. Analysis Models 13](#_Toc475542347)

[4.1 Log In 15](#_Toc475542348)

[4.1.1 Log In System Sequence Diagram 15](#_Toc475542349)

[4.1.2 Log In Contracts 15](#_Toc475542350)

[4.2 Log Out 16](#_Toc475542351)

[4.2.1 Log Out System Sequence Diagram 16](#_Toc475542352)

[4.2.2 Log Out Contracts 16](#_Toc475542353)

[4.3 View Calendar 17](#_Toc475542354)

[4.3.1 View Calendar System Sequence Diagram 17](#_Toc475542355)

[4.3.2 View Calendar Contracts 17](#_Toc475542356)

[4.4 Request Reservation 17](#_Toc475542357)

[4.4.1 Request Reservation System Sequence Diagram 18](#_Toc475542358)

[4.4.2 Request Reservation Contracts 18](#_Toc475542359)

[4.5 Modify Reservation 19](#_Toc475542360)

[4.5.1 Modify Reservation System Sequence Diagram 19](#_Toc475542361)

[4.5.2 Modify Reservation Contracts 19](#_Toc475542362)

[4.6 View Reservation 20](#_Toc475542363)

[4.6.1 View Reservation System Sequence Diagram 20](#_Toc475542364)

[4.6.2 View Reservation Contracts 20](#_Toc475542365)

[4.7 View Reservation List 21](#_Toc475542366)

[4.7.1 View Reservation System Sequence Diagram 21](#_Toc475542367)

[4.7.2 View Reservation Contracts 21](#_Toc475542368)

[4.8 Cancel Reservation 22](#_Toc475542369)

[4.8.1 Cancel Reservation System Sequence Diagram 22](#_Toc475542370)

[4.8.2 Cancel Reservation Contracts 22](#_Toc475542371)

**List of Figures**

[Figure 1: Use Case Diagram 6](#_Toc475542372)

[Figure 2: Domain Model 13](#_Toc475542373)

[Figure 3. System State Diagram 14](#_Toc475542374)

[Figure 4: System Sequence Diagram for Log In 15](#_Toc475542375)

[Figure 5: System Sequence Diagram for Logout 16](#_Toc475542376)

[Figure 6: System Sequence Diagram for View Calendar 17](#_Toc475542377)

[Figure 7: System Sequence Diagram for Request Reservation 18](#_Toc475542378)

[Figure 8: System Sequence Diagram for Modify Reservation 19](#_Toc475542379)

[Figure 9: System Sequence Diagram for View Reservation 20](#_Toc475542380)

[Figure 10: System Sequence Diagram for View Reservation List 21](#_Toc475542381)

[Figure 11: System Sequence Diagram for Cancel Reservation 22](#_Toc475542382)

**List of Tables**

[Table 1: Actor Goal List 5](#_Toc475542383)

[Table 2: Use Case for Log In 7](#_Toc475542384)

[Table 3: Use Case for Log Out 7](#_Toc475542385)

[Table 4: Use Case for View Calendar 8](#_Toc475542386)

[Table 5: Use Case for Create Reservation 8](#_Toc475542387)

[Table 6: Use Case for Modify Reservation 9](#_Toc475542388)

[Table 7: Use Case for View Reservation 10](#_Toc475542389)

[Table 8: Use Case for View Reservation List 10](#_Toc475542390)

[Table 9: Use Case for Cancel Reservation 10](#_Toc475542391)

[Table 10: Contract for View Calendar 17](#_Toc475542392)

[Table 11: Contract for Show Request Form 18](#_Toc475542393)

[Table 12: Contract for Request Reservation 18](#_Toc475542394)

[Table 13: Contract for showModifyForm 19](#_Toc475542395)

[Table 14: Contract for modifyReservation 19](#_Toc475542396)

[Table 15: Contract for View Reservation 20](#_Toc475542397)

[Table 16: Contract for View Reservation List 21](#_Toc475542398)

[Table 17: Contract for Cancel Reservation 22](#_Toc475542399)

# Introduction

In software development, an ordered process is required in engineering a software system. Software Engineering sets the rubric to delivering quality software with respect to the constraints agreed upon by a project's stakeholders. The main stages of this process include requirements engineering, design, implementation, testing, maintenance and version control. The Software Requirements Specification (SRS) is dedicated to the first stage of the software development process; requirements engineering.

In this SRS document, we will go into detail about the project's overall description and specific requirements.

## Purpose

The SRS is a documentation artifact that specifies all requirements the system has to meet. This includes both functional (user goals) and nonfunctional requirements (reliability, usability, etc.) for a room booking system.

This document is intended to be used by the team responsible for creating the Software Architecture Document (SAD). For this piece of software, the same team responsible for the SRS will be writing the SAD with help from the requirements detailed within this document. The SAD team will focus on designing a system that meets all functional requirements without sacrificing the nonfunctional requirements.

## Scope

The software product to be produced is Chronos, an online University conference room reservation system. The system should maintain a directory of rooms to be reserved at different time slots by users registered with ENCS. It should be accessible by multiple users at a time, where they can create, cancel, modify or view existing reservations. These reservations should also be maintained by the system.

### User Login/Logout

The application handles authentication of users already registered with the facility. A user must enter valid username and password credentials to have full access of Chronos’ features. A user is also able to log out when they are done using the application.

### Mutual Exclusion

The application allows authenticated users to access the system’s registry to view the availabilities and existing reservations of the rooms. It allows for multiple users to be logged in at once, however a specific room-timeslot combination can only be reserved by one user at a time.

Once a user enters a room at a selected Timeslot, they will be granted a reservation request for 60 seconds. During these 60 seconds, the room will be locked from use by any other users and they will have to complete the form to make the request. If they do not complete the form during the 60 seconds, regardless of how much they completed they will be redirected back to the calendar page.

NOT YET COMPLETED

In the scenario where they successfully complete their request or do not, once they are back on the calendar page, they will be forced to wait 30 seconds before they can make another request. This is to prevent the same user from repeatedly locking rooms and making them inaccessible to the rest of the users.

In the case where midway through the request the users’ browser happens to close/crash, the System has a backup timer running on the server which will automatically reopen the room after the 60 seconds has passed. As well, it will also signal that the user must now wait their allotted 30 seconds, regardless of the fact that the browser close/crash might not have been their fault.

The aim of the system to promote safety, liveness and fairness.

### Equipment

For every timeslot, there is a list of equipment which can be spread about various reservations in the 5 available rooms. The list of equipment includes Display Cables, Projectors, Laptops, and White Board Markers and there is a limited quantity of 3 for each listed.

When the user creates their reservation request, they must specify the amount of each equipment they desire. There is no limit on how much they can request, however the restriction is that the equipment must be available for the reservation to be successfully added.

This means that even if the room is available over the users desired time slot, they cannot get the reservation if the requested equipment is not available. If the equipment is not available, they will be placed on a waitlist (described below) pending availability of the equipment.

### Waiting list

The application implements waiting list functionality. If a user chooses to reserve a room at a time slot that is already reserved by another and/or where the equipment they request is not available at that given timeslot, the application places the user on a waiting list for that room-timeslot combination. There is a limited capacity of 3 people for any waitlist given a specific room-timeslot combination.

This allows wait-listed users to obtain their desired reservation upon cancellation from the user currently occupying the desired room or equipment. The application does so by removing the former user from the reservation slot or the associated equipment, adding the new user to the slot and removing this new user from all waiting lists for any other rooms reserved over the same time slot.

This means that if a user needs a room or equipment for a given timeslot, the user can reserve multiple rooms at that timeslot and will be placed on waiting lists for all of them. The user will only be taken off these waiting lists if they obtain one of the reservations or reach their maximum active reservations for the week (in this case 3).

As well, if a user cancels their reservation but the next person on the waiting list requires equipment that is still not available, they will be skipped in the list until someone is found whose equipment count is completely available for that timeslot.

### Recurring Reservations

The application allows a user to book the same reservation on a repeated time slot for a maximum of 3 weeks in the future. When requesting a reservation, the user specifies the amount of times to repeat the same reservation for the following weeks. However, if the user wants to delete the recurring reservations, they will need to do by individually selecting each and deleting them.

### Maximum Reservations

The system provides 5 rooms for booking on a 24-hour basis. The only limitation to this system is that a specific user can only create bookings totaling 3 hours per week (Equivalent to 3 timeslots). If a user has already booked for 3 hours in a given week, they cannot create a new reservation or be placed on a waiting list for that same week.

### Capstone Students

This system provides priority to Capstone students. If a capstone student wants to book a given room but the room has already been taken by another student, they will be placed on the waiting list ahead of all “regular” students, but behind any other capstone students who are currently on the waiting list. However, when it comes to equipment they are subject to the same constraints as “regular” users.

## Glossary of Terms

|  |  |
| --- | --- |
| Term | Definition |
| Active Reservation | The earliest reservation created for a room at a specific time slot |
| Available | Room that can be reserved by a user |
| Reservation | Time slot which has been booked by a user |
| Time Slot | One-hour long time interval starting on the hour |
| User | Student or staff who is registered with the engineering faculty |
| Waiting list | A series of reservations that represents the order in which users will take over a canceled reservation for a specific room and time |

## References

|  |  |
| --- | --- |
| [1] | D. C. Constantinides, "Software Requirements Specification Template," September 2016. |
| [2] | IEEE, "Software Requirements Specification (SRS) Template," 1993. |

# Overall Description

The general factors that affect Chronos are the given objectives, constraints, data and time to complete the product. The functionality and restrictions of the application should correspond to what was asked for in the project description. We should also assume that the users already exist in the facility and therefore already possess existing credentials found in a database table of users with respectful authentication information.

## Product Perspective

Chronos is a self-contained system. Although it needs users to function as intended, it stands on its own. The way it handles the reservation of rooms and cancelations does not depend on someone behind the system entering the reservations. The algorithm verifies for availability before confirming any reservations.

## Product Functions

The primary functions of Chronos are to offer registered users the possibility to view the calendar as it is. By doing so, the user will be able to see which rooms are available and at what time. Furthermore, they will be given the possibility to create a reservation on a recurring basis and to confirm it. After reserving a room, the user will be able to look at a list of their reservations and cancel any of them if desired. After completing every task, the user has the possibility to log out.

## User Characteristics

The targeted users of this system are the engineering students (ENCS) of Concordia and its staff members. The whole purpose of this system is to allow them to easily and quickly reserve a room whenever they need to. With an intuitive and easy to learn UI, the user will not need any experience nor technical expertise to be able to work with the web application.

## Constraints

The utility of the reservation system is limited by various factors such as the number of rooms available for reservation, the operating hours of the facility (currently 24-hrs, but subject to change due to circumstances outside the control of the university, I.E damage or construction), and the event where students occupy rooms without having a reservation. The number of rooms available for reservation determines how many registered users can use the system. The operating hours of the facility determines when a room can be used. The event of a user is not respecting the booking system, there must be some sort of authority figure to enforce and supervise quarrels.

The Concordia University specific application of Chronos should meet the following constraints. The rooms available to the Chronos reservation system should be limited to 5 rooms provided to students. Rooms should only be made available on a 24-hr basis.

## Assumptions and Dependencies

In the case of this software based application, there is no need of any other applications besides a web browser. It is assumed that wherever the user is trying to reserve a classroom, they will have access to an up to date web browser. By making our system compatible with the most popular web browsers (Chrome, Firefox, Internet Explorer, Edge and Safari), it allows the users to use any browser of their liking.

# Specific Requirements

The specific requirements of Chronos can be broken down into the following sections: External Interfaces, Functional Requirements and Non-Functional Requirements. External interfaces take the form or external actors and/or subsystems. Functional requirements or user goals will be detailed in the form of Use Case descriptions and their functionality can be shown in a Use Case Diagram. Finally, a series of nonfunctional requirements will be explained in natural language.

## External Interfaces documentation

No external subsystems will be required to produce Chronos. All subsystems will be implemented as integral components of the system. The only external interface to the system will be the Code Repository. All the source code and documentation for the Chronos system can be found at <https://github.com/wolfcall/Tyrell-Corp>

## Functionality documentation

A detailed description of the functionality the system must provide to its various users is contained in the following subsections.

### Actor Goal List documentation

Table 1: Actor Goal List

|  |  |
| --- | --- |
| Actor | Goal |
| <<User>> | Log In |
| Log Out |
| View Calendar |
| Request Reservation |
| Modify Reservation |
| View Reservation |
| View Reservation List |
| Cancel Reservation |

### Use Case View documentation

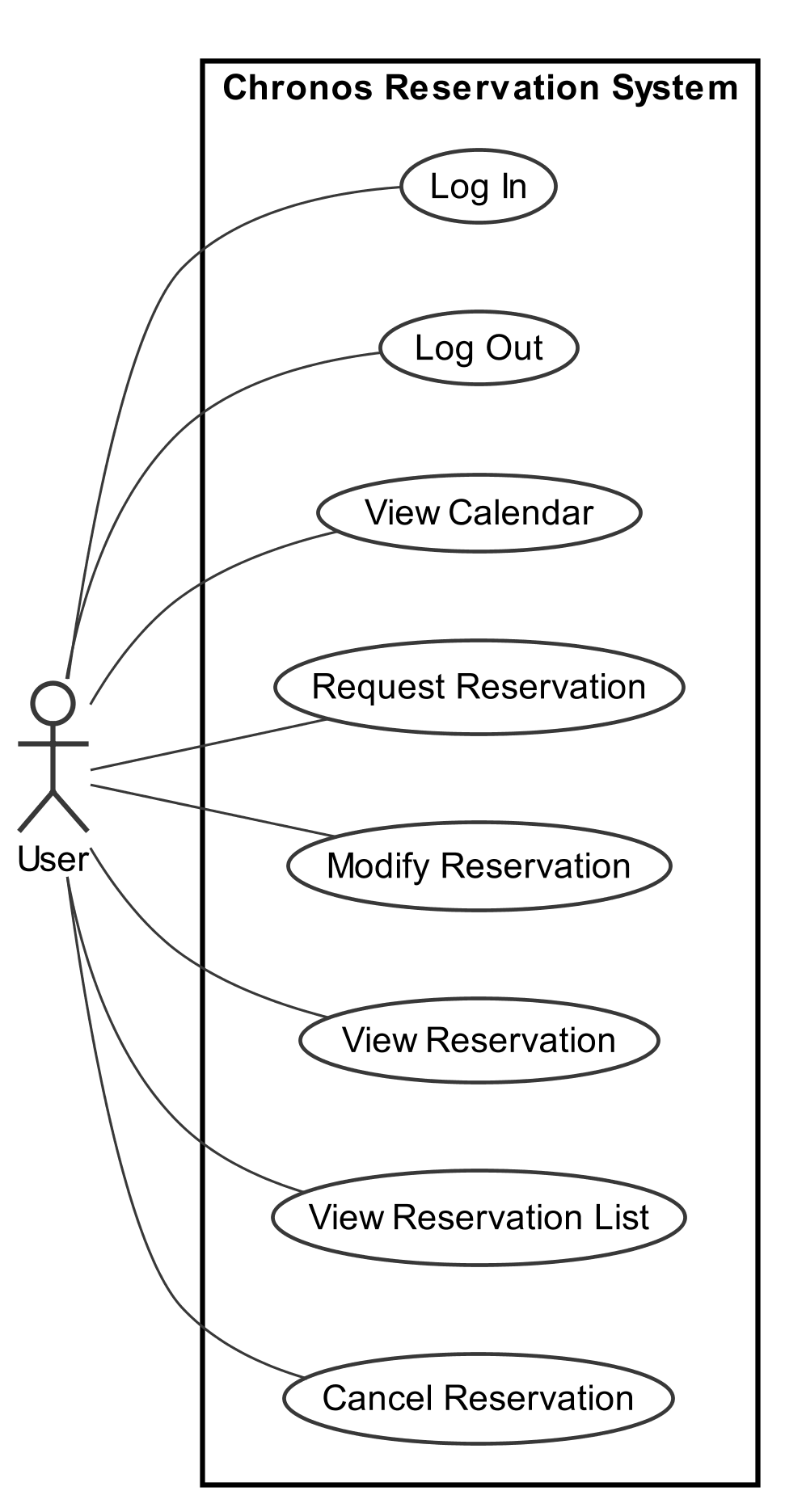


Figure 1: Use Case Diagram

### Use Case Descriptions

#### Log In

Table 2: Use Case for Log In

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC01 | | |
| **Use Case Name:** | Log In | | |
| **Created By:** | Steve Ferreira | **Last Updated By:** | Stefano Pace |
| **Date Created:** | October 14, 2016 | **Date Last Updated:** | February 22, 2017 |
| **Actors:** | * User | | |
| **Goal:** | User logs into system using their ID and password | | |
| **Summary:** | Users with an account log into it establishing a connection with the system. From this point until they manually log out the system will consider them logged in. | | |
| **Preconditions:** | * User is not logged in, and has an existing account with the System | | |
| **Post conditions:** | * User is considered logged in for future interactions with the system. | | |
| **Basic flow:** | 1. User indicates that they wish to log in 2. System prompts the student to log in 3. User provides his student ID and password 4. System provides a successful login message | | |
| **Minimum guarantee:** | None. | | |
| **Notes and Issues:** | None. | | |

#### Log Out

Table 3: Use Case for Log Out

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC02 | | |
| **Use Case Name:** | Log Out | | |
| **Created By:** | Steve Ferreira | **Last Updated By:** | Stefano Pace |
| **Date Created:** | October 14, 2016 | **Date Last Updated:** | February 22, 2017 |
| **Actors:** | * User | | |
| **Goal:** | The user to no longer be considered as logged in by the system | | |
| **Summary:** | Users logged into their account will be disconnected from the system and no longer considered to be logged in. | | |
| **Preconditions:** | * User is logged in | | |
| **Post conditions:** | * User is no longer considered to be logged in by the system | | |
| **Basic flow:** | 1. User indicates they wish to log out 2. System redirects user to home page | | |
| **Minimum guarantee:** | User is logged out of the system. | | |
| **Notes and Issues:** | None. | | |

#### View Calendar

Table 4: Use Case for View Calendar

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC03 | | |
| **Use Case Name:** | View Calendar | | |
| **Created By:** | Steve Ferreira | **Last Updated By:** | Stefano Pace |
| **Date Created:** | October 14, 2016 | **Date Last Updated:** | February 22, 2017 |
| **Actors:** | * User | | |
| **Goal:** | A user views a calendar listing time slots and rooms. | | |
| **Summary:** | User can access a calendar listing of time slots and rooms. The calendar shows whether a room is available or reserved for a given time and day. | | |
| **Preconditions:** | * User is logged in. | | |
| **Post conditions:** | * None | | |
| **Basic flow:** | 1. User indicates they wish to view the calendar 2. System redirects them to the calendar page | | |
| **Minimum guarantee:** | Calendar page is displayed to the user | | |
| **Notes and Issues:** | Calendar view should show that rooms are not available as well as rooms | | |

#### Request Reservation

Table 5: Use Case for Create Reservation

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC04 | | |
| **Use Case Name:** | Request Reservation | | |
| **Created By:** | Angelo Pengue | **Last Updated By:** | Stefano Pace |
| **Date Created:** | October 16, 2016 | **Date Last Updated:** | February 22, 2017 |
| **Actors:** | User | | |
| **Goal:** | A user requests to reserve a room for a specific time slot with equipment which will recur over a specific number of weeks. | | |
| **Summary:** | A user selects an available time slot from the calendar view. As long as the waiting list limit has not been reached and the user has not reached their maximum allowed number of time slots per week, a reservation will be requested with the equipment specified. If available, it is successfully created. The order in which reservations are created determines who currently holds the active reservation and who is on the waiting list. | | |
| **Preconditions:** | * User is logged in to the system * User is viewing the Calendar * The waiting list of the room and time slot is not full * User has not exceeded the maximum number of reservations allowed in a week | | |
| **Post conditions:** | * A reservation is created for the room and time slot for the specified number of weeks | | |
| **Basic flow:** | 1. User requests to make a reservation 2. System provides a form to request the reservation 3. User requests reservation of a room at a time slot with equipment for some number of weeks 4. System returns the result of the request | | |
| **Alternate flow (1):** | 1. User requests to make a reservation 2. System provides a form to request the reservation 3. User requests reservation of a room at a time slot with equipment for some number of weeks 4. System indicates that the room has already reached the maximum number of reservations | | |
| **Alternate flow (2):** | 1. User requests to make a reservation 2. System indicates that the user has reached the maximum number of reservations allowed for the week | | |
| **Minimum guarantee:** | None | | |
| **Notes and Issues:** | A recurring reservation will attempt to book the same room at the same time for multiple weeks. In the event a room is not available for one of the recurring weeks the user will be notified of conflicts and the time slot will still be booked in the subsequent weeks where no conflict is detected.  If a room does not have an active reservation but the requested equipment in the reservation is not available, then the user will be placed on a waiting list pending the availability of the equipment.  If a user is a member of capstone, then if the room already has an active reservation, prompting the capstone user to be placed on the waitlist, they will be placed ahead of any “regular students” but behind any other capstone student already on the waiting list. | | |

#### Modify Reservation

Table 6: Use Case for Modify Reservation

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC05 | | |
| **Use Case Name:** | Modify Reservation | | |
| **Created By:** | Steve Ferreira | **Last Updated By:** | Stefano Pace |
| **Date Created:** | November 7, 2016 | **Date Last Updated:** | February 22, 2017 |
| **Actors:** | User | | |
| **Goal:** | Modify the information of a given reservation | | |
| **Summary:** | While viewing a reservation the user requests to modify the information provided for their reservation. They are provided a form to modify the reservations information. The reservation is updated with the modified information. | | |
| **Preconditions:** | * User is logged in * User has an existing reservation * User is viewing a reservation | | |
| **Post conditions:** | * Reservation has been updated | | |
| **Basic flow:** | 1. User requests to modify a reservation 2. System provides a form to modify the reservation 3. User submits modified reservation information 4. System returns the result of the modification | | |
| **Alternate flow:** | None | | |
| **Minimum guarantee:** | None | | |
| **Notes and Issues:** | Reservation modifications are subject to many of the same checks as requesting a reservation. The new equipment (if changed) request must be available, the new timeslot (if changed) must be available and the new room (if changed) must be available. | | |

#### View Reservation

Table 7: Use Case for View Reservation

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC06 | | |
| **Use Case Name:** | View Reservation | | |
| **Created By:** | Angelo Pengue | **Last Updated By:** | Stefano Pace |
| **Date Created:** | October 14, 2016 | **Date Last Updated:** | February 22, 2017 |
| **Actors:** | * User | | |
| **Goal:** | A user views information regarding their existing reservation | | |
| **Summary:** | User can access an existing reservation from the calendar or from a list of their current reservations. This will display information relevant to that specific reservation. | | |
| **Preconditions:** | * User is logged in * User has an existing reservation | | |
| **Post conditions:** | * None | | |
| **Basic flow:** | 1. User indicates they wish to view information for a specific reservation 2. System presents user with information about the reservation | | |
| **Minimum guarantee:** | Reservation information is displayed to the user | | |
| **Notes and Issues:** |  | | |

#### View Reservation List

Table 8: Use Case for View Reservation List

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC07 | | |
| **Use Case Name:** | View Reservation List | | |
| **Created By:** | Angelo Pengue | **Last Updated By:** | Stefano Pace |
| **Date Created:** | October 16, 2016 | **Date Last Updated:** | February 22, 2017 |
| **Actors:** | User | | |
| **Goal:** | User views room reservations associated with their account. | | |
| **Summary:** | A user requests to view their reservations. The system retrieves the user’s room reservations and presents them to the user. | | |
| **Preconditions:** | * User is logged in to the system | | |
| **Postconditions:** | * None | | |
| **Basic flow:** | 1. User requests system to view their reservations 2. System presents all reservations associated with the user’s account | | |
| **Minimum guarantee:** | System presents room reservations associated with user’s account | | |
| **Notes and Issues:** | This view only shows upcoming reservations. All reservations in the past will not be shown. | | |

#### Cancel Reservation

Table 9: Use Case for Cancel Reservation

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC08 | | |
| **Use Case Name:** | Cancel Reservation | | |
| **Created By:** | Angelo Pengue | **Last Updated By:** | Stefano Pace |
| **Date Created:** | October 16, 2016 | **Date Last Updated:** | February 22, 2017 |
| **Actors:** | User | | |
| **Goal:** | User cancels an existing room reservation | | |
| **Summary:** | The user is viewing a reservation associated with their account. The user indicates to the system that they wish to cancel the reservation. The system removes the reservation associated with the user’s account for that room and time slot. If there is another user associated with the waiting list for that room and time slot, the system creates a reservation (only if equipment is available) and associates it to that user’s account for the canceled room and time slot. The wait-listed user is no longer associated with the waiting list for that room and time slot. | | |
| **Preconditions:** | * User is logged in to the system * User has a reservation with room and time slot they wish to cancel * User is viewing that reservation | | |
| **Post conditions:** | * Reservation for that room and time slot is no longer associated with the user’s account | | |
| **Basic flow:** | 1. User requests system to cancel room reservation for a room and time slot 2. System checks waiting list and updates the reservation with the next user information | | |
| **Minimum guarantee:** | User’s account no longer has a reservation associated with the room and time slot. | | |
| **Notes and Issues:** | None | | |

## Nonfunctional Requirements

The following sections describe all nonfunctional requirements the Chronos system is expected to meet.

### Reliability documentation

The room booking service is a relatively simple application. Due to its low complexity, the system is expected to have a 99.99% uptime during reservation hours. Furthermore, the system shall serve at least 100 concurrent users without jeopardizing uptime. Finally, the database shall keep current reservations saved without any data being unexpectedly wiped.

### Usability documentation

The room booking service shall have a friendly user interface (UI) to facilitate the overall user-experience (UX). For Chronos to be an intuitive experience for students, simple UI elements shall be used to help the users. Furthermore, the application shall have a separate UI for smaller screens to allow for a better user experience across mobile devices and tablets.

### Efficiency documentation

Since Chronos is a web application, it is expected to perform rapidly during any operation. It is expected that any request sent from the client shall receive a response from the server in less than 2 seconds. The system must be performant in response time, or else it shall detract from the overall usability and reliability.

### Maintainability documentation

System components shall remain relatively independent: for example, the design of a time slot should not depend on the design of available room. This will produce a highly modular system that is easier to maintain.

### Portability documentation

Since Chronos is a web application, it must be portable across most modern browsers. These include Google Chrome, Mozilla Firefox, Microsoft Edge and Safari. The OS is independent since the application is running inside the browser. It is also expected that these browsers are all running up to date versions.

### Design Constraints documentation

Chronos shall be built using the Laravel framework, which is based on the PHP 7 programming language. Laravel allows for the development of Chronos to be rapid and elegant. Additionally, Bootstrap CSS shall be used to model the user-facing interface of the web application, as it provides ready-built and styled components which will promote reusability and fast prototyping.

The source code of Chronos is located on a GitHub repository to allow for source control and pushing with Git. Furthermore, GitHub is used to track issues and each issue is assigned to a member to complete.

### User Documentation and Help documentation

The room booking service Chronos is an online web application. Therefore a manual should be created showing how to setup and install the developed system. This is available in the corresponding SAD as well as on GitHub.

## Purchased Components documentation

Chronos is an online web application. Therefore a web server and domain address will be required to host the website.

### Web Server

A server will need to be purchased for the system. The server must provide access to the website over the internet. The server will hold the source files and database required for the system to function.

### Domain Name

A domain name is needed so users can access the system easily over the internet. Entering the purchased domain into their internet browser will direct them to Chronos**.**

# Analysis Models

The system can be represented using a set of conceptual classes, derived from the use case descriptions and flow steps. These conceptual classes and their associations are shown below.

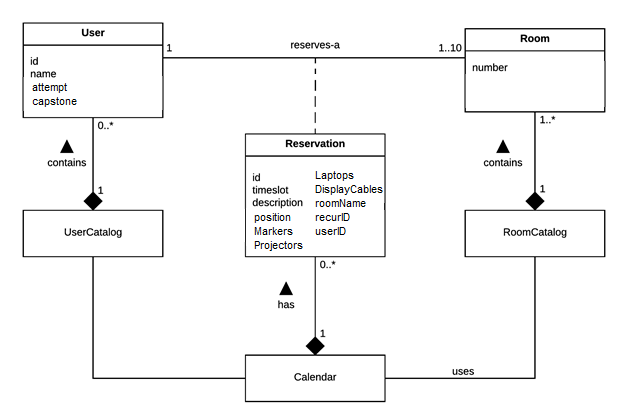


Figure 2: Domain Model

The Calendar class is the primary point of interaction with the system. It is formed of Reservation classes, which associate a User and a Room for a specific timeslot. To be able to form these associations, the Calendar uses a UserCatalog and a RoomCatalog, which serve as repositories for the respective domain classes.

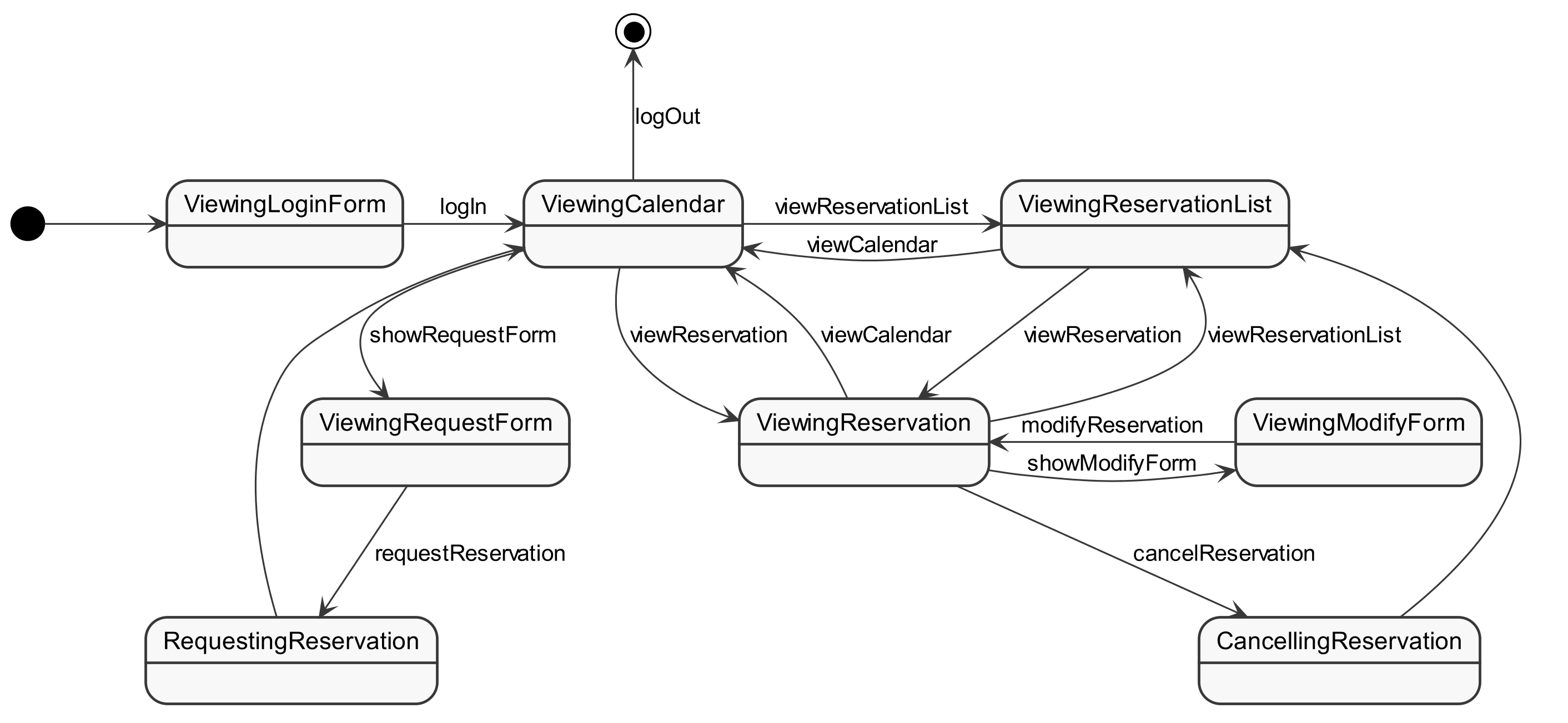


Figure 3. System State Diagram

The above state diagram can describe the various conceptual states and legal flows the system can be in, depending on the user input it receives. Each state loosely represents a page of the web application, with state transitions mapping to user input actions on the web pages.

To start, a user must log in, and is shown the main calendar view. From there, the user can request a reservation by viewing the ReservationForm, view a Reservation, or view the ReservationList. While viewing a Reservation, the user may opt to cancel it or modify it. If the user logs out of the system, the session will terminate and wait for a new log in.

## Log In

Registered users log in and establishes a connection with the system. From this point until they manually log out, the system will consider them logged in.

### Log In System Sequence Diagram

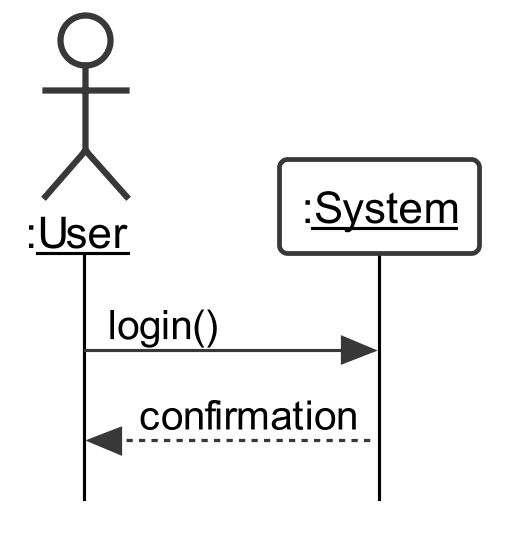


Figure 4: System Sequence Diagram for Log In

### Log In Contracts

|  |  |
| --- | --- |
| Contract CO01.1: | login |
| **Operation** | login() |
| **Cross Reference** | UC01: Log In |
| **Preconditions** | * None |
| **Post conditions** | * None |

## Log Out

Users that are logged in will be disconnected from the system and will no longer be considered as logged in.

### Log Out System Sequence Diagram

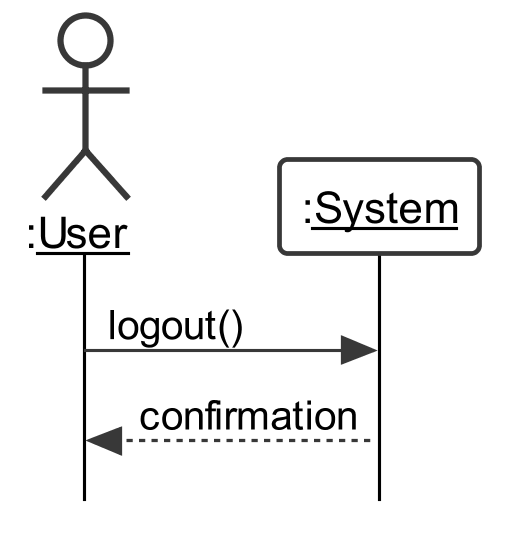


Figure 5: System Sequence Diagram for Logout

### Log Out Contracts

|  |  |
| --- | --- |
| Contract CO02.1: | logout |
| **Operation** | logout() |
| **Cross Reference** | UC02: Log Out |
| **Preconditions** | * None |
| **Post conditions** | * None |

## View Calendar

Users that are logged in can access a calendar listing all time slots and rooms. The calendar shows whether a room is available or reserved for a given time and day

### View Calendar System Sequence Diagram

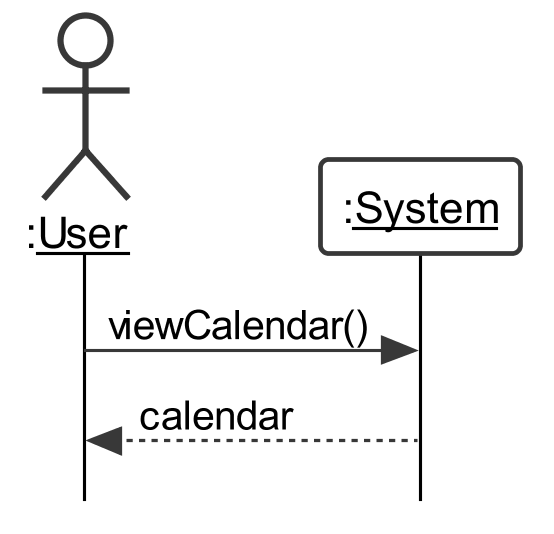


Figure 6: System Sequence Diagram for View Calendar

### View Calendar Contracts

Table 10: Contract for View Calendar

|  |  |
| --- | --- |
| Contract CO03.1: | viewCalendar |
| **Operation** | viewCalendar() |
| **Cross Reference** | UC03: View Calendar |
| **Preconditions** | * The user is logged in |
| **Post conditions** | * None |

## Request Reservation

A user selects an available time slot from the calendar view. As long as the waiting list limit has not been reached, the user has not reached their maximum allowed number of time slots per week, and the equipment is available, a reservation is successfully created. The order in which reservations are created determines who currently holds the active reservation and who is on the waiting list.

### Request Reservation System Sequence Diagram

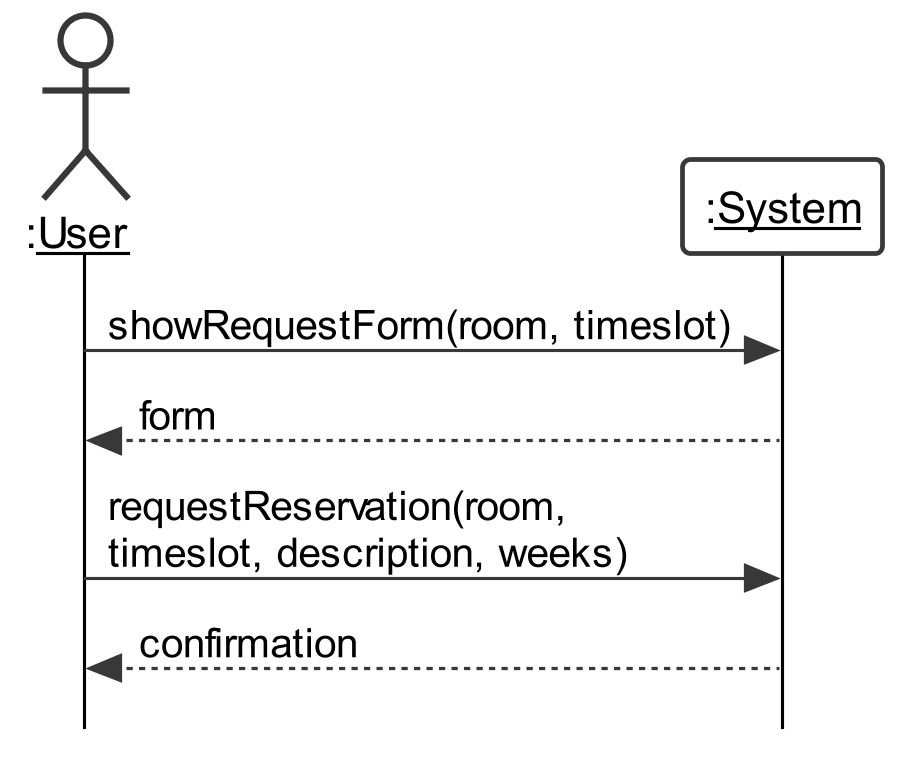


Figure 7: System Sequence Diagram for Request Reservation

### Request Reservation Contracts

Table 11: Contract for Show Request Form

|  |  |
| --- | --- |
| Contract CO04.1: | showRequestForm |
| **Operation** | showRequestForm(room, timeslot) |
| **Cross Reference** | UC04: Request Reservation |
| **Preconditions** | * User is logged in * User is viewing the calendar * The waiting list of the room and time slot is not full * User must not exceed the maximum number of reservations allowed in a week |
| **Postconditions** | * None |

Table 12: Contract for Request Reservation

|  |  |
| --- | --- |
| Contract CO04.1: | requestReservation |
| **Operation** | requestReservation(room, timeslot, description, weeks) |
| **Cross Reference** | UC04: Request Reservation |
| **Preconditions** | * User is viewing the reservation’s request form |
| **Postconditions** | * An instance of Reservation r has been created for each room and time slot (based on the specified number of weeks recurring) * The created instances of Reservation have been associated with User * The created instances of Reservation have been associated with Calendar |

## Modify Reservation

While viewing a reservation, the user can request to modify the information provided for their reservation. They are provided a form to modify the reservation’s information. The reservation is successfully updated with the modified information if all the changes meet the constraints previously outlined (Equipment available, no conflicts with the room and/or timeslot).

### Modify Reservation System Sequence Diagram

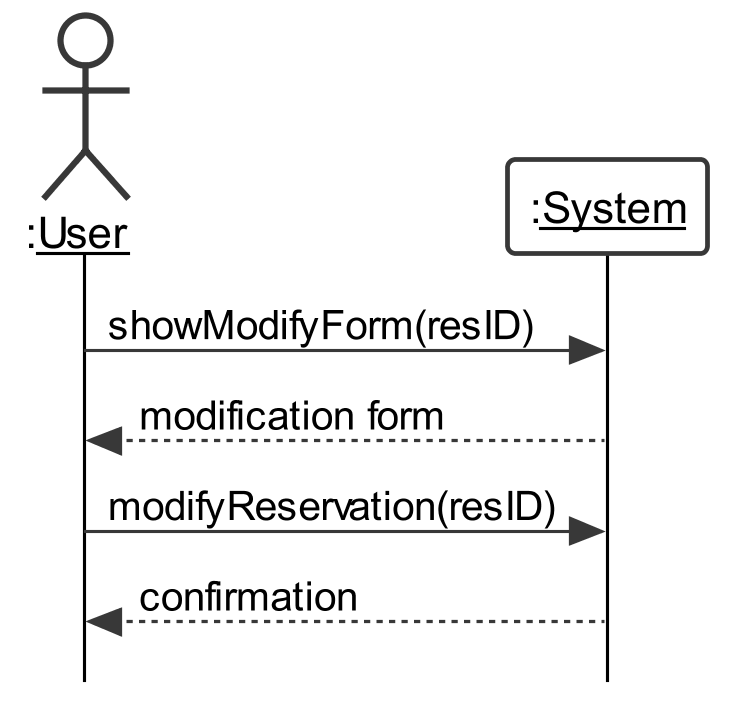


Figure 8: System Sequence Diagram for Modify Reservation

### Modify Reservation Contracts

Table 13: Contract for showModifyForm

|  |  |
| --- | --- |
| Contract CO05.1: | showModifyForm |
| **Operation** | showModifyForm(resID) |
| **Cross Reference** | UC05: Modify Reservation |
| **Preconditions** | * User is logged in * User is viewing their Reservation for a room-timeslot |
| **Post conditions** | * None |

Table 14: Contract for modifyReservation

|  |  |
| --- | --- |
| Contract CO05.2: | modifyReservation |
| **Operation** | modifyReservation(resID) |
| **Cross Reference** | UC05: Modify Reservation |
| **Preconditions** | * User is viewing the reservation’s modification form |
| **Post conditions** | * Reservation attributes have been modified |

## View Reservation

User can access an existing reservation from the calendar or from a list of their current reservations. Doing so will display relevant information to that specific reservation.

### View Reservation System Sequence Diagram

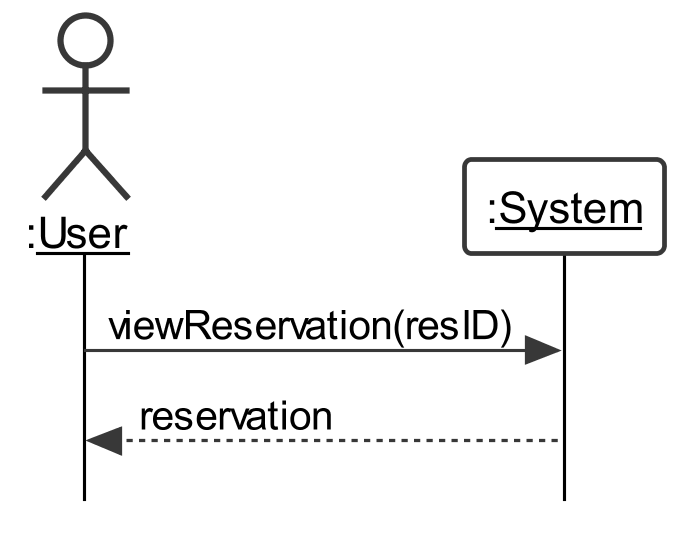


Figure 9: System Sequence Diagram for View Reservation

### View Reservation Contracts

Table 15: Contract for View Reservation

|  |  |
| --- | --- |
| Contract CO06.1: | viewReservation |
| **Operation** | viewReservation(resID) |
| **Cross Reference** | UC06: View Reservation |
| **Preconditions** | * User is logged in * Reservation to be viewed belongs to the user |
| **Post conditions** | * None |

## View Reservation List

A user requests to view all their reservations. The system retrieves the user’s room-timeslot reservations and presents them to the user.

### View Reservation System Sequence Diagram

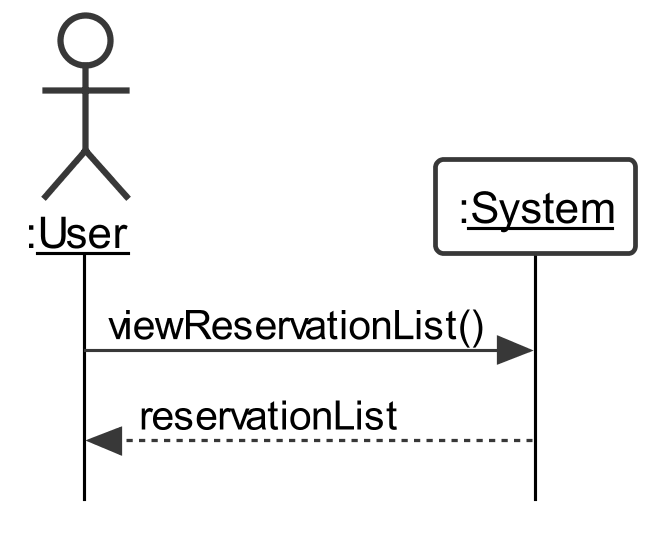


Figure 10: System Sequence Diagram for View Reservation List

### View Reservation Contracts

Table 16: Contract for View Reservation List

|  |  |
| --- | --- |
| Contract CO07.1: | viewReservationList |
| **Operation** | viewReservationList() |
| **Cross Reference** | UC07: View Reservation List |
| **Preconditions** | * User is logged in |
| **Post conditions** | * None |

## Cancel Reservation

The user is viewing a reservation associated with their account. The user indicates to the system that they wish to cancel the reservation. The system removes the reservation associated with the user’s account for that room and time slot. If there is another user associated with the waiting list for that room and time slot, the system creates a reservation (only if equipment is available) and associates it to that user’s account for the canceled room and time slot. The wait-listed user is no longer associated with the waiting list for that room and time slot.

### Cancel Reservation System Sequence Diagram

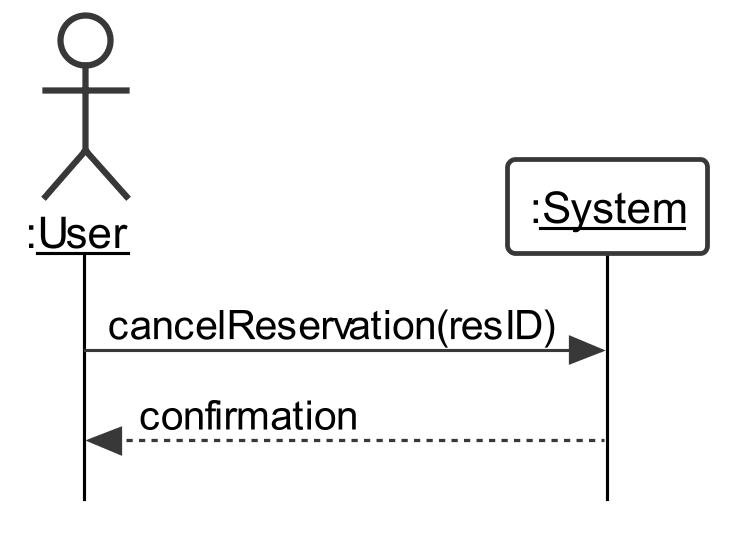


Figure 11: System Sequence Diagram for Cancel Reservation

### Cancel Reservation Contracts

Table 17: Contract for Cancel Reservation

|  |  |
| --- | --- |
| Contract CO08.1: | cancelReservation |
| **Operation** | cancelReservation(resID) |
| **Cross Reference** | UC08: Cancel Reservation |
| **Preconditions** | * User is logged in * The reservation belongs to the user * User is viewing the reservation |
| **Post conditions** | * Associations between User and the reservation has been deleted * Associations between Calendar and the reservation has been deleted |