chronos

Concordia University
Department of Computer Science
and Software Engineering

Software Architecture and Design II
SOEN 344 S --- 2017

Project Chronos
Software Requirements Specification

Team members	
Name	**Student ID**
An Ran Chen	27277385
Christiano Bianchet	27039573
Philippe Kuret	27392680
Alexander Rosser	27543069
Saif Mahabub	27392974
Adriel Fabella	27466005
Youness Tawil	27013353
Amr Mourad	26795331

Table of Contents

List of Figures

List of Tables

1. 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.

1.1 Purpose

The SRS is a documentation artifact that specifies all requirements the system has to meet. This includes both functional (user goals) and non-functional 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 non-functional requirements.

1.2 Scope

The software product to be produced is Chronos, an online college conference room reservation system. The system should maintain a directory of rooms to be reserved at different time slots by users registered with the facility. 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.

1.2.1 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.

1.2.2 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 can only be reserved by one user at a time.

This requirement has been modified from its original specification for reasons of technical feasibility. Due to the complications of modern web server technologies, there is no straightforward way to implement mutual exclusion for page load connections. A user could select a time slot (thereby “locking” it), and simply close their browser window. As server connections are closed immediately after page load, the server is stuck with a locked time slot with no knowledge of the client having abandoned it. This would require implementing expirations for the locks, which is outside of the scope of this application. Additionally, simply halting a user’s incoming connection to view a presently locked room, without implementing a technology such as socket-driven communication, would cause the user’s browser to hang indefinitely and provide a negative experience.

The requirement will instead be partially satisfied by gracefully displaying an error message telling the user that the room was already reserved in the case of an attempted reservation of an already booked room, instead of locking a certain room from all readers.

1.2.3 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, the application places the user on a waiting list for that room and time, with a limited capacity of 5. This allows wait-listed users to obtain their desired reservation upon cancellation from the user currently occupying it. The application does so by removing the former user from the reservation slot, 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 for a given time slot and all of them are currently occupied, the user can reserve multiple rooms for the same time 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, cancel a reservation or if the time reserved for has passed. Additionally, a capstone student who wishes to be put on the waitlist will be given priority in the list over non-capstone students.

1.2.4 Recurring Reservations

The application allows a user to book the same room on a repeated time slot for a maximum of 3 weeks. When creating a reservation, the user can choose to repeat the same reservation for a chosen number of following weeks. If the user deletes a reservation, the application will ask the user if they want to delete other recurring reservations from the same set. If they accept, all their reservations with the same room and time slot for different weeks will be deleted.

1.2.5 Maximum Reservations

The application keeps track of the number of reservations made per user. A user may only reserve a maximum 3 hours per week. The user will be denied a reservation if it causes them to surpass this limit.

1.3 Glossary of Terms

Table 1: 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
Equipment	Computers (3) and Projectors (3) that are made available to a user

1.4 References

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

2. 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. The project should be completed within the given timeframe and delivered on the deadline, April 4th, explicitly noted in the project description.

2.1 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 cancellations does not depend on someone behind the system entering the reservations. The algorithm verifies for availability before confirming any reservations.

2.2 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 with or without an equipment 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 or to close the window which in this case will automatically log the user out.

2.3 User Characteristics

The targeted users of this system are the engineering students 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.

2.4 Constraints

There are a few constraints placed on the system. The software is required to allow a multitude of students to view the room reservations simultaneously. The system must also provide safety for all write functions. The system also has the following restrictions; a total of five rooms must be available in the system on a 24-hour basis, a user may reserve a maximum of 3 hours a week, a user may create a repeated reservation for no more than 3 weeks straight, capstone students are awarded priority for reservations and lastly a reservation may only be granted to a user if both the room and equipment requested are available.

2.5 Assumptions and Dependencies

In the case of this software based application, there is no need of any other applications besides a web browser and a populated database. It is assumed that wherever the user is trying to reserve a classroom, they will have access to a 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.

3. 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 non-functional requirements will be explained in natural language.

3.1 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/anrchen/soen344.

3.2 Functionality documentation

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

3.2.1 Functional Requirements

Commented [1]: Need to update in table

Commented [2]: was the table updated?

Further information about use cases is explained in section 3.2.4.

Table 2: Functional Requirements

Functional Requirements	Relevant Use Cases
The system will allow the user to securely log in.	Use case 01
The system will allow the user to log out.	Use case 02
The system will allow the user to view the status in all the rooms.	Use case 03
The system will allow the user to reserve a maximum of 10 time slots a week.	Use case 04
If a user on a wait list obtains the reservation, the system removes the user from all other wait lists with identical time slots.	Use cases 04 & 08

The system will allow only one user at a time to add, modify or cancel a reservation on a specific room.	Use cases 04, 05 & 08
The system will allow the user to view information regarding an existing reservation.	Use case 06
The system will allow the user to view a list of reservations associated with their account.	Use case 07
The system will allow the user to reserve a maximum of 3 hours a week.	Use case 04
The system will allow the user to request equipment while reserving a room.	Use case 04
The system will give priority for reservations to users that are capstone student.	Use cases 04, 05 & 08

3.2.2 Actor Goal List documentation

Table 3: Actor Goal List

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

3.2.3 Use Case View documentation

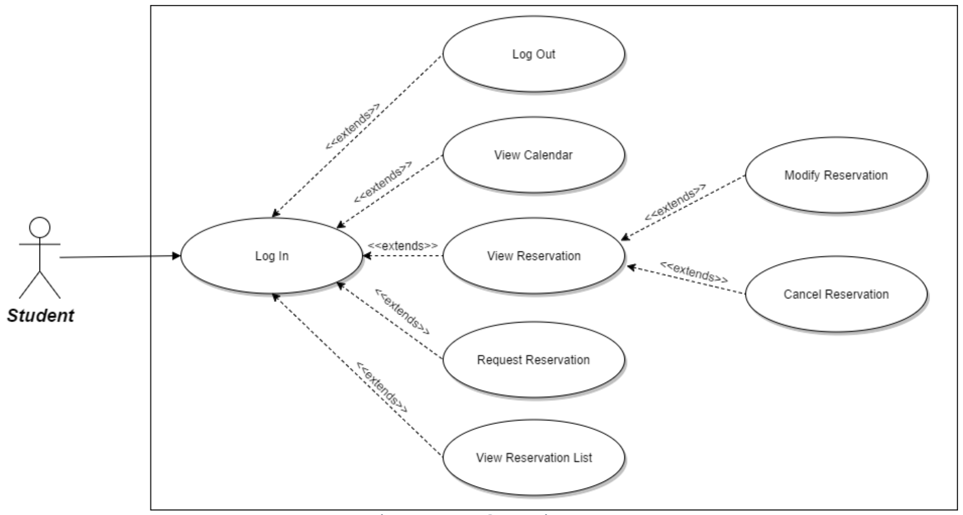


Figure 1: Use Case Diagram

3.2.4 Use Case Descriptions

3.2.4.1 Log In

Table 4: Use Case for Log In

Use Case ID:	**UC01**		
Use Case Name:	Log In		
Created By:	Steve Ferreira	**Last Updated By:**	Alexander Rosser
Date Created:	October 14, 2016	**Date Last Updated:**	February 20, 2017
Actors:	● User		
Goal:	User logs into system using their ID and password		
Summary:	Users with an account logs in, establishing a connection with the system. from this point until they manually logout.		
Preconditions:	● User is not logged in ● User has an account		
Postconditions:	● User is logged into 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 authenticates information and provides a successful login message		
Minimum guarantee:	None.		
Notes and Issues:	None.		

3.2.4.2 Log Out

Table 5: Use Case for Log Out

Use Case ID:	UC02		
Use Case Name:	Log Out		
Created By:	Steve Ferreira	Last Updated By:	Alexander Rosser
Date Created:	October 14, 2016	Date Last Updated:	February 20, 2017
Actors:	● User		
Goal:	The user is logged out of the system.		
Summary:	Logged in user selects to log out and is disconnect from the system.		
Preconditions:	● User is logged in		
Postconditions:	● 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.		

3.2.4.3 View Calendar

Table 6: Use Case for View Calendar

Use Case ID:	UC03		
Use Case Name:	View Calendar		
Created By:	Steve Ferreira	Last Updated By:	Alexander Rosser
Date Created:	October 14, 2016	Date Last Updated:	February 20, 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.		
Postconditions:	● None		
Basic flow:	1. User indicates they wish to view the calendar 2. System redirects the user to calendar page		
Minimum guarantee:	Calendar page is displayed to the user		
Notes and Issues:	Oct 14,2016: Calendar view should show that rooms are not available		

3.2.4.4 Request Reservation

Table 7: Use Case for Create Reservation

Use Case ID:	UC04		
Use Case Name:	Request Reservation		
Created By:	Angelo Pengue	Last Updated By:	Saif Mahabub
Date Created:	October 16, 2016	Date Last Updated:	April 2, 2017
Actors:	User		
Goal:	A user requests to reserve a room for a specific time slot 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, the user has not reached their maximum allowed number of time slots per week and the desired 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.
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 ● The desired equipment is available
Postconditions:	● 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 for some number of weeks 4. System returns the result of the request
Alternate 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 4. System indicates that the room has already reached the maximum number of reservations
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.

3.2.4.5 Modify Reservation

Table 8: Use Case for Modify Reservation

Use Case ID:	**UC05**		
Use Case Name:	Modify Reservation		
Created By:	Steve Ferreira	**Last Updated By:**	Alexander Rosser
Date Created:	November 7, 2016	**Date Last Updated:**	February 20, 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' description. The reservation is updated with the modified information.		
Preconditions:	● User is logged in ● User has a reservation ● User is viewing a reservation		
Postconditions:	● Reservation is 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 updates the reservation		
Alternate flow:	None		

Minimum guarantee:	Changes to the reservation will be saved
Notes and Issues:	

3.2.4.6 View Reservation

Table 9: Use Case for View Reservation

Use Case ID:	**UC06**		
Use Case Name:	View Reservation		
Created By:	Angelo Pengue	**Last Updated By:**	Alexander Rosser
Date Created:	October 14, 2016	**Date Last Updated:**	February 20, 2017
Actors:	● User		
Goal:	A user views information regarding an 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 ● Reservation belongs to the User		
Postconditions:	● 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:	Time slot is displayed to the user		
Notes and Issues:	October 14,2016:Time slot view should show when the room has a waiting list for that time slot.		

3.2.4.7 View Reservation List

Table 10: Use Case for View Reservation List

Use Case ID:	**UC07**		
Use Case Name:	View Reservation List		
Created By:	Angelo Pengue	**Last Updated By:**	Angelo Pengue
Date Created:	October 16, 2016	**Date Last Updated:**	October 16, 2016
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 to view room reservations 2. System presents room reservations associated with the user’s account		
Minimum guarantee:	System presents room reservations associated with user’s account		
Notes and Issues:	October 16, 2016: Display a message to user if their reservation list is empty		

3.2.4.8 Cancel Reservation

Table 11: Use Case for Cancel Reservation

Use Case ID:	**UC08**		
Use Case Name:	Cancel Reservation		
Created By:	Angelo Pengue	**Last Updated By:**	Saif Mahabub
Date Created:	October 16, 2016	**Date Last Updated:**	April 2, 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 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		
Postconditions:	● Reservation for that room/timeslot and/or equipment 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:	Capstone users are given priority in the waiting list upon creating a reservation. Therefore, if they are position 1 in the waiting list when a reservation is cancelled they are automatically associated to the room they were wait listed in.		

3.3 Non-functional Requirements

The following sections describe all non-functional requirements the Chronos system is expected to meet.

3.3.1 Reliability documentation

The room booking service is a relatively simple application. Due to its low complexity, the system is expected to have a 100% uptime during reservation hours. Furthermore, the system shall serve at least 1000 concurrent users without jeopardizing uptime. Finally, the database shall keep current reservations saved without any data being unexpectedly wiped.

3.3.2 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.

3.3.3 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.

3.3.4 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.

3.3.5 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.

3.3.6 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. Plus, the integration of an AOP framework is required to migrate the interceptor pattern. (A detailed analysis on the migration is discussed in SAD - section 5. AOP Migration of Interceptor Pattern.

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. Finally, Travis CI is used as a continuous integration server and will automatically deploy all changes made to the source code onto a live testing server, which shall be used by all team members to aid in testing and validation. Travis will also generate a set of class and method documentation from the source code, which shall be available to the team and serve as a reference.

3.3.7 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.

3.4 Purchased Components documentation

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

3.4.1 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.

3.4.2 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**.**

4. 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.

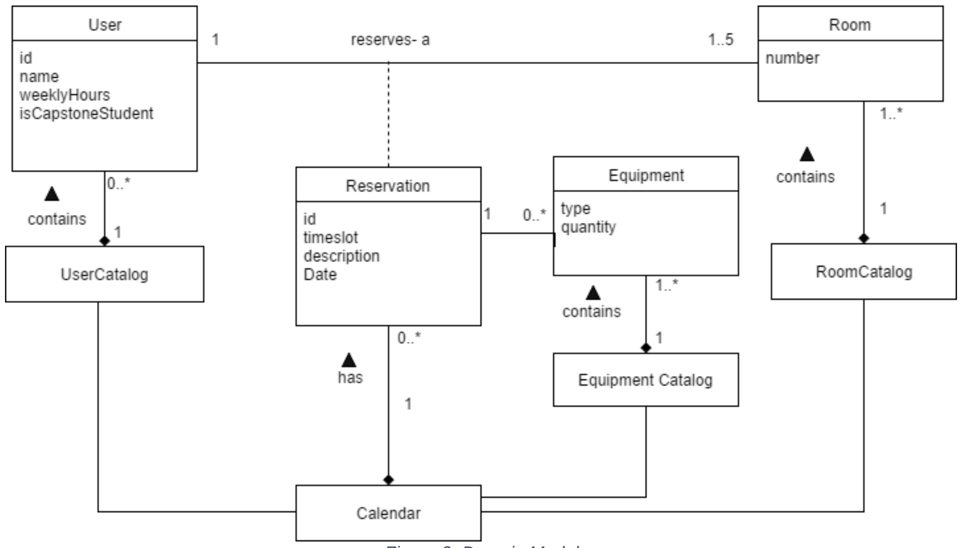


Figure 2: Domain Model

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.

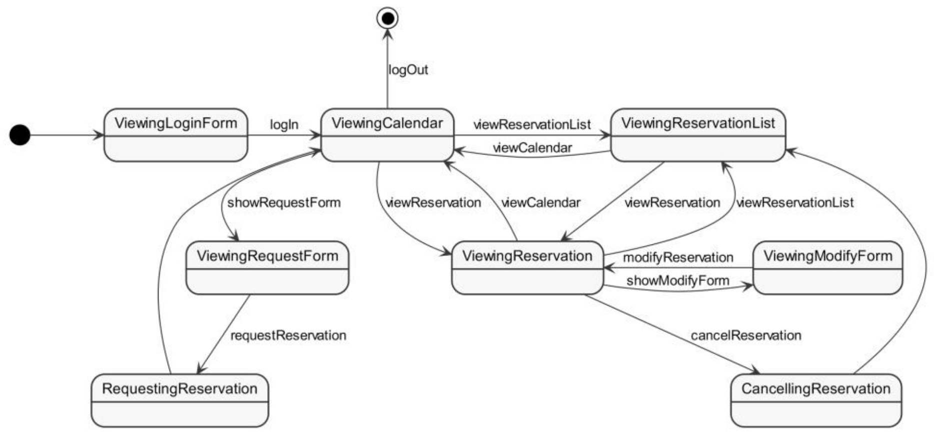


Figure 3: System State Diagram

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.

4.1 Log In

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

4.1.1 System Sequence Diagram

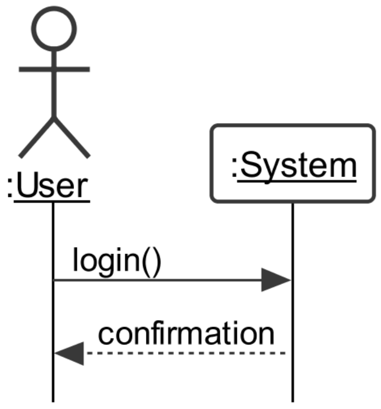


Figure 4: System Sequence Diagram for Log In

4.2 Log out

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

4.2.1 System Sequence Diagram

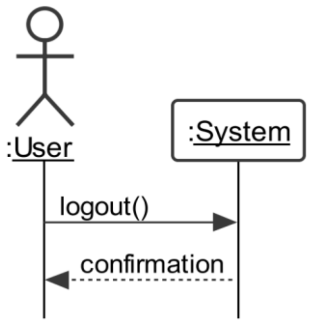


Figure 5: System Sequence Diagram for Logout

4.3 View Calendar

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

4.3.1 System Sequence Diagram

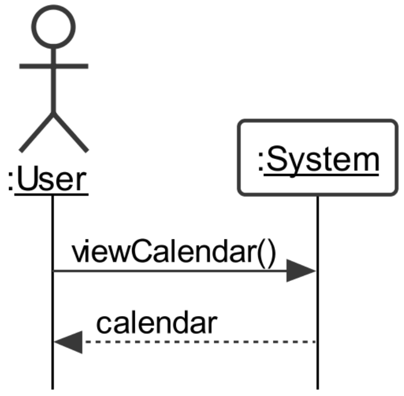


Figure 6: System Sequence Diagram for View Calendar

4.3.2 Contracts

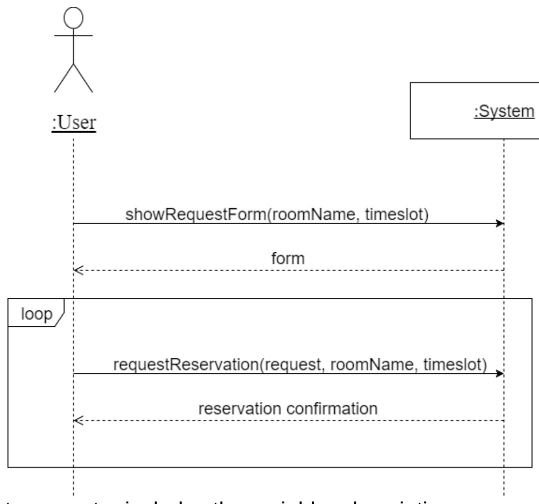
Table 12: Contract for View Calendar

Contract CO03.1:	viewCalendar
Operation	viewCalendar()
Cross Reference	UC03: View Calendar
Preconditions	● The user is logged in
Postconditions	● None

4.4 Request Reservation

A user selects an available time slot from the calendar view. If 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 is successfully created. The order in which reservations are created determines who currently holds the active reservation and who is on the waiting list.

4.4.1 System Sequence Diagram


:User
:System
showRequestForm(roomName, timeslot)
form
loop
requestReservation(request, roomName, timeslot)
reservation confirmation

(request parameter includes the variables description, recur and equipment)

Figure 7: System Sequence Diagram for Request Reservation

4.4.2 Contracts

Table 13: Contract for Show Request Form

Contract CO04.1:	showRequestForm
Operation	showRequestForm(roomName, 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 14: Contract for Request Reservation

Contract CO04.2:	requestReservation
Operation	requestReservation(request, roomName, timeslot)
Cross Reference	UC04: Request Reservation
Preconditions	● User is viewing the reservation request form
Postconditions	● An instance of Reservation r has been created for the time slot for the specified number of weeks ● The created instances of Reservation have been associated with User

	● The created instances of Reservation have been associated with Calendar

4.5 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 updated with the modified information.

4.5.1 System Sequence Diagram

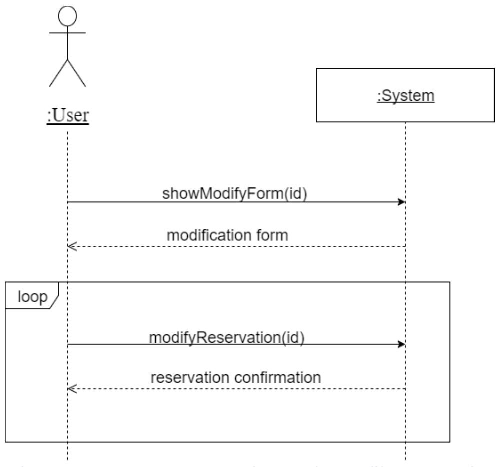


Figure 8: System Sequence Diagram for Modify Reservation

4.5.2 Contracts

Table 15: Contract for showModifyForm

<u>Contract</u> CO05.1:	**showModifyForm**
Operation	showModifyForm(id)
Cross Reference	UC05: Modify Reservation
Preconditions	● User is logged in ● Reservation belongs to the user ● User is viewing the calendar
Postconditions	● None

Table 16: Contract for modifyReservation

<u>Contract</u> CO05.2:	**modifyReservation**
Operation	modifyReservation(id)

Cross Reference	UC05: Modify Reservation
Preconditions	● User is viewing the reservation modification form
Postconditions	● The attribute description from the instance Reservation has been modified.

4.6 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.

4.6.1 System Sequence Diagram

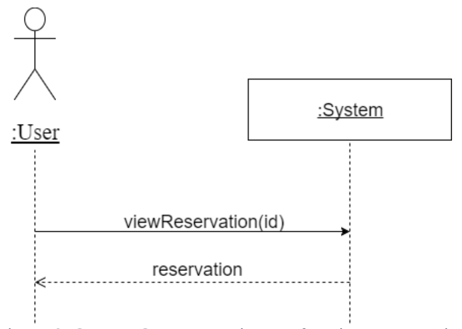


Figure 9: System Sequence Diagram for View Reservation

4.6.2 Contracts

Table 17: Contract for View Reservation

Contract CO06.1:	**viewReservation**
Operation	viewReservation(id)
Cross Reference	UC06: View Reservation
Preconditions	● User is logged in ● Reservation to be viewed belongs to the user
Postconditions	● None

4.7 View Reservation List

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

4.7.1 System Sequence Diagram

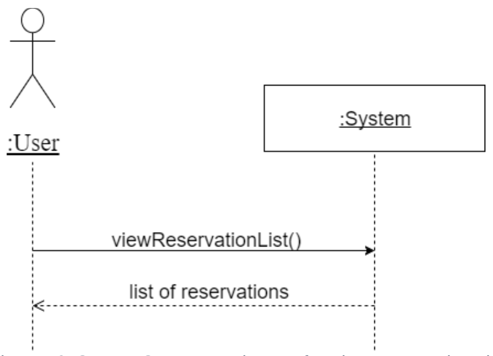


Figure 10: System Sequence Diagram for View Reservation List

4.7.2 Contracts

Table 18: Contract for View Reservation List

Contract CO07.1:	viewReservationList
Operation	viewReservationList()
Cross Reference	UC07: View Reservation List
Preconditions	● User is logged in
Postconditions	● None

4.8 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 (regular waiting list or waiting list pending availability of an equipment) for that room and/or time slot, the system creates a reservation 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.

4.8.1 System Sequence Diagram

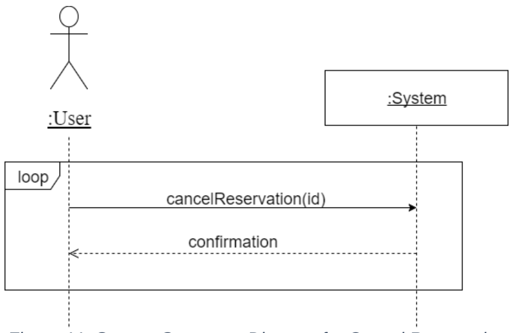


Figure 11: System Sequence Diagram for Cancel Reservation

4.8.2 Contracts

Table 19: Contract for Cancel Reservation

Contract CO08.1:	cancelReservation
Operation	cancelReservation(id)
Cross Reference	UC08: Cancel Reservation
Preconditions	● User is logged in ● The reservation belongs to the user ● User is viewing the reservation
Postconditions	● Associations between User and the reservation, along with all its recurrences, have been removed ● Associations created between Equipment and ReservationMapper ● Attribute *waitlisted* modification for Reservation instances with other users

5. Test Report

The testing report covers testing on the Table Data Gateway files of the application. Unit tests were made to ensure that there is a proper communication between the third-party system, which is the MySQL database. Also, one test was made for the Identity Map to ensure that the system stores the data that is received from the database.

5.1 Method Coverage

Table 20: Method coverage

Class Name	Method Coverage (%)
UserTDG.php	29% (2/7)
RoomTDG.php	50% (½)
ReservationSessionTDG.php	75% (¾)
EquipmentTDG.php	50% (½)
EquipmentIdentityMap	100% (3/3)

5.2 Unit Test Summary

Table 21: Unit test summary

Total number of unit tests	13

5.3 Unit Test Report

Table 22: Test suite for Class RoomTDG.php

TEST CASE 1: Method**: find (string room)**		
Tests that a room is successfully retrieved upon the input of a valid room number.		
Input	**Expected Output**	**Result**

H-901	An object with the room number	**PASS**
TEST CASE 2: Method**: find (string room)**		
Tests that a room is successfully retrieved upon the input of an invalid room number.		
Input	**Expected Output**	**Result**
H11111	Nothing is to be returned (null)	**PASS**
TEST CASE 3: Method**: find (string room)**		
Tests that a room is successfully retrieved upon the input of an empty string.		
Input	**Expected Output**	**Result**
Null	Nothing is to be returned (null)	**PASS**

Table 23: Test suite for Class ReservationSessionTDG.php

TEST CASE 1: Method**: makeNewSession (ReservationSession $session)**		
Tests that a session has been successfully added from the sessions table.		
Input	**Expected Output**	**Result**

('10000009', 'H-903', '2017-03-26 19:00:00')	The corresponding session has been added to the session table	**PASS**
TEST CASE 2: Method**: endSession (ReservationSession $session)**		
Tests that a session has been successfully deleted from the sessions table.		
Input	**Expected Output**	**Result**
('10000009', 'H-903', '2017-03-26 19:00:00')	The corresponding session has been removed from the sessions table	**PASS**
TEST CASE 3: Method**: checkSessionInProgress (ReservationSession $session)**		
Tests that the session that was added to the table is still active and still exists in the table.		
Input	**Expected Output**	**Result**
('10000009', 'H-903', '2017-03-26 19:00:00')	Returns **true** indicating that the **session** still exists in the table	**PASS**

Table 24: Test suite for Class EquipmentTDG.php

TEST CASE 1: Method**: find (int $equipment_id)**

Tests that an equipment object (name, amount) is successfully retrieved based on fetching its id.		
Input	**Expected Output**	**Result**
1	An equipment object will be returned	**PASS**
TEST CASE 2: Method**: find (int $equipment_id)**		
Tests that upon inserting an invalid equipment id, no object is returned.		
Input	**Expected Output**	**Result**
null	Nothing is to be returned (null)	**PASS**

Table 25: Test suite for Class UserTDG.php

TEST CASE 1: Method**: create(User $user)**		
The test ensured that the user was successfully inserted in the User table. Once the user is created we retrieve it by ID and compare it to the original user.		
Input	**Expected Output**	**Result**
new user(123,"test","password", false)	A user is added to the User table	**PASS**
TEST CASE 2: Method**: add(Equipment $equipment)**		
The Test is successful when a user's name is updated in the User table. Create a new user then update the user name by ID. Retrieve the user by ID from the database and compare to the data passed to the update method.		
Input	**Expected Output**	**Result**

new user(1234,"testUpdate","password", false)	The user is Updated in the User Table	**PASS**

Table 26: Test suite for Class EquipmentIdentityMap.php

TEST CASE 1: Method**: get(int $id))**		
The test is successful when the added data is retrieved correctly or null is returned if no data exists. . First add an equipment to memory then use the get method to retrieve it by ID and compare. Second use the get method to retrieve an equipment that is not in memory and compare with null.		
Input	**Expected Output**	**Result**
654321	An equipment Object	**PASS**
888888	null	**PASS**
TEST CASE 2: Method**: add(Equipment $equipment)**		
The Test is successful when an equipment is added to memory. Add an equipment to memory then retrieve it by Id and compare.		
Input	**Expected Output**	**Result**
new Equipment($id, "test",123456)	The equipment object is added to memory	**PASS**
TEST CASE 3: Method**: delete(Equipment $equipment**		
The Test is successful when an equipment is deleted from memory. Add an equipment to memory then remove it using the delete method. Try getting the removed equipment by using the get method and compare the result with null.		
Input	**Expected Output**	**Result**

654321	The equipment object is deleted from memory	PASS

5.4 Acceptance Test Report

The tables below provide a detailed acceptance test report considering both previous requirements and new requirements.

The following table provides tests for the basic path of the website:

Table 27: Acceptance test basic path

Step #	Execution Procedure/Input	Expected Result/Output	Passed/Failed
1. Log in	Log in to the Chronos website with a valid user ID and password.	Logged in to Chronos and Calendar page opens.	**PASS**
2. Request reservation	Click on an empty timeslot/room.	Request a reservation page opens.	**PASS**
3. Reservation information	Complete required fields and click Request.	Reservation data entered/selected is saved, calendar page opens and a message displays reservation created successfully.	**PASS**
4. Reservation list	Click My reservations link.	Reservation list page opens.	**PASS**
5. Reservation status	Select a reservation and click View.	Reservation page opens.	**PASS**
6. Modify Reservation	Click Modify.	Modify reservation page opens.	**PASS**
7. Reservation modifications	Complete required field and click Modify.	Reservation data entered is saved and a message displays reservation modified successfully.	**PASS**
8. Calendar	Click Calendar link.	Calendar page opens.	**PASS**

9. Calendar information	Input date and click Jump to date.	Calendar page opens on desired date.	**PASS**
10. Reservation status	Click on a taken timeslot/room (in green).	Reservation page opens.	**PASS**
11. Reservation cancellation	Click Cancel this and all recurring.	Reservation(s) data is deleted and a message displays reservation cancelled successfully.	**PASS**

The following table provides tests for the alternative flows such as being waitlisted, reaching max reservation per week, reaching max recurrence, waitlist is full, equipment is unavailable, session expires and closing browser before session is completed:

Table 28: Acceptance test for alternative flows

Step #	Execution Procedure/Input	Expected Result/Output	Passed/Failed
1. Log in	Log in to the Chronos website with a valid user ID and password.	Logged in to Chronos and Calendar page opens.	**PASS**
2. Request reservation	Click on a reserved timeslot/room.	Request a reservation page opens.	**PASS**
2.1 Reservation information	Complete required fields and click Request.	Reservation data entered/selected is saved, calendar page opens and a message indicating user has been put into the waitlist with the position number is displayed.	**PASS**
3. Request reservation (reservation reached for the week)	Click on an empty or reserved timeslot/room.	Request a reservation page opens.	**PASS**

3.1 Reservation information	Complete required fields and click Request.	Error message displays reservation request limit for the week has been exceeded.	**PASS**
4. Request reservation (max recurrence reached)	Click on an empty similar timeslot/room from recurring reservation after 3 weeks.	Request a reservation page opens.	**PASS**
4.1 Reservation information	Complete required fields and click Request.	Error message displays recurring reservation limit has been exceeded.	**PASS**
5. Request reservation (waitlist full)	Click on a reserved timeslot/room.	Request a reservation page opens.	**PASS**
5.1 Reservation information	Complete required fields and click Request.	Error message displays waiting list is full.	**PASS**
6. Request reservation (equipment unavailable)	Click on an empty timeslot/room.	Request a reservation page opens.	**PASS**
6.1 Reservation information	Complete required fields and click Request.	Error message displays requested equipment is not available and user is put into a waitlist pending availability of the equipment.	**PASS**
7. Request reservation (session expired)	Click on an empty timeslot/room.	Request a reservation page opens.	**PASS**
7.1 Reservation session expires	Wait more than 60 seconds.	Message displays session has expired.	**PASS**
7.2 Reservation information	Complete required fields and click Request.	Reservation data entered/selected is	**PASS**

(reservation taken by other user)		saved, calendar page opens and a message indicating user has been put into the waitlist with the position number is displayed.	
8. Request Reservation (close browser)	Click on an empty timeslot/room.	Request a reservation page opens.	**PASS**
8.1 Leaving reservation session	Close browser.	Browser closed.	**PASS**
8.2 Log in	Log in to the Chronos website with a valid user ID and password.	Logged in to Chronos and Calendar page opens.	**PASS**
8.3 Request Reservation	Click on the previous desired timeslot/room	Request a reservation page opens.	**PASS**
8.4 Reservation information	Complete required fields and click Request.	Reservation data entered/selected is saved, calendar page opens and a message displays reservation created successfully.	**PASS**

The following table provides tests scenarios considering the user being a capstone student:

Table 29: Acceptance test for capstone student

Step #	Execution Procedure/Input	Expected Result/Output	Passed/Failed
1. Log in	Log in to the Chronos website as a capstone student with a valid user ID and password.	Logged in to Chronos and Calendar page opens.	**PASS**
2. Request reservation	Click on a reserved timeslot/room.	Request a reservation page opens.	**PASS**

3. Reservation information	Complete required fields and click Request.	Reservation data entered/selected is saved. A message indicating user has been put into waitlist position 1 unless another capstone student is in position 1 then the user is put into the following position of the waitlist.	**PASS**