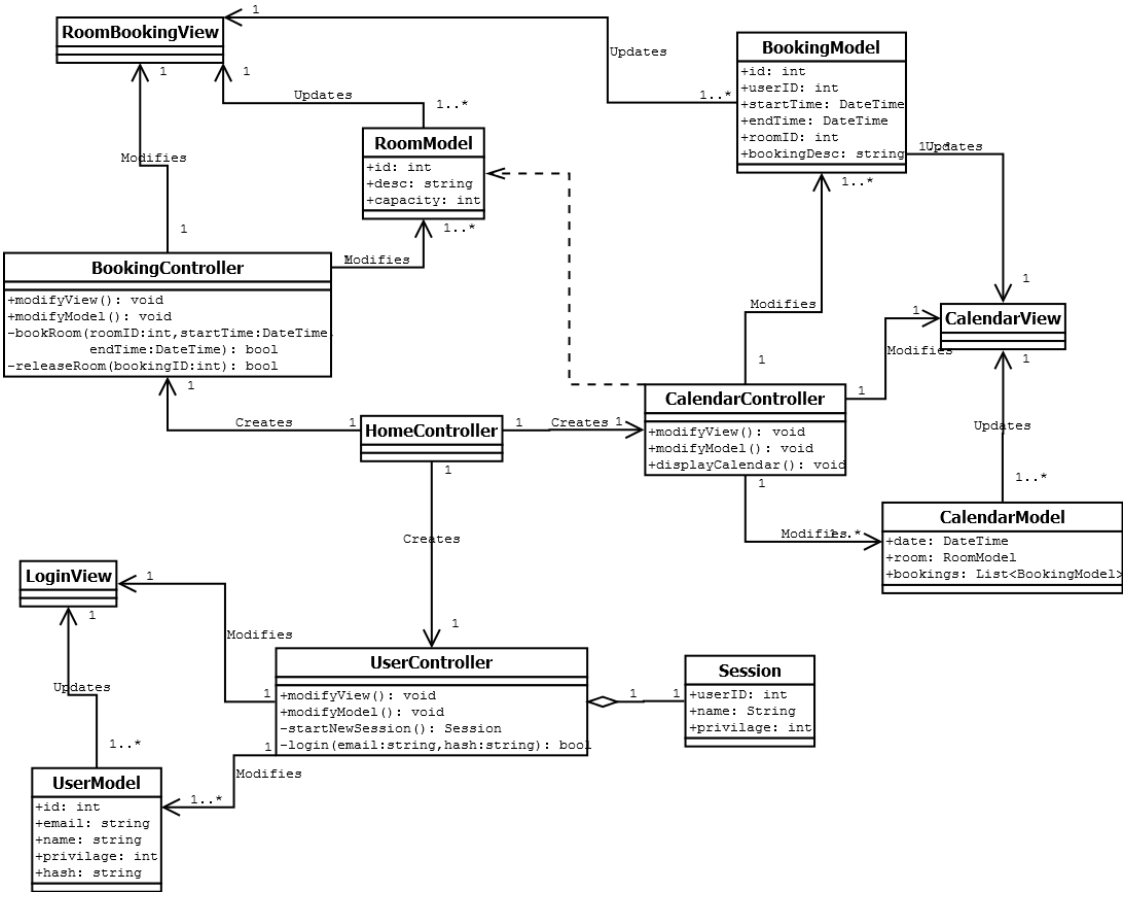
Class Diagram

Group Members - Guy Coccimiglio Justin Rhude Michael Thomas

Revision History

Version	Date	Description	Author
First Revision	Mar 14, 2017	Final version.	Justin Rhude



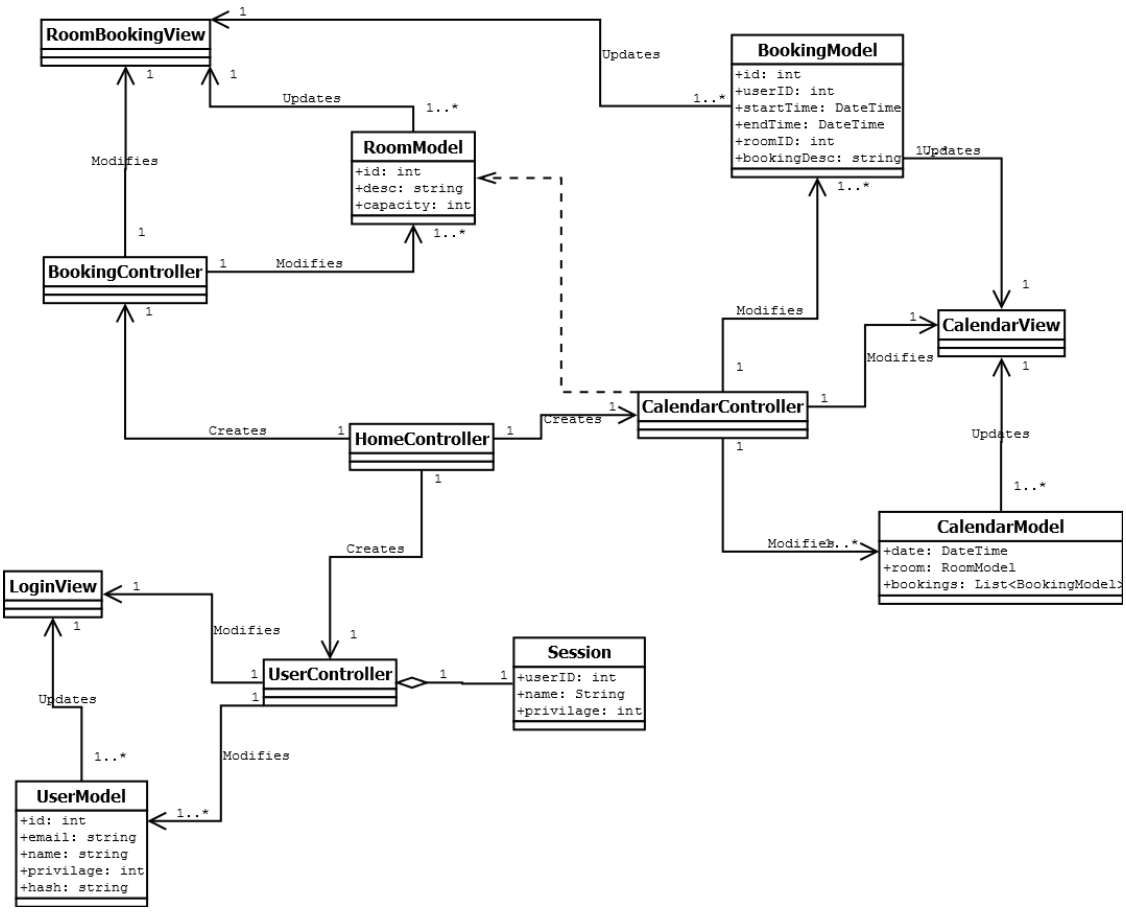
Domain Model

Group Members - Guy Coccimiglio Justin Rhude Michael Thomas

Revision History

Version	Date	Description	Author
Inception draft	Feb 17, 2017	First draft.	Justin Rhude
Second Revision	Mar 12, 2017	Final version.	Justin Rhude

The Model



Glossary

Group Members - Guy Coccimiglio Justin Rhude Michael Thomas

Revision History

Version	Date	Description	Author
Inception draft	Feb 12, 2017	First draft.	Justin Rhude
Second revision	Mar 23, 2017	Final version.	Justin Rhude

Terms

General

- Booking - Setting a room to the booked state at a specific time and date.
- Viewing - Checking the state of a room at a given time and date.
- Reserving - Setting a room to the booked state at a specific time and date.
- Priority - If a conflict arises, priority determines who gets the time slot. Admins have priority over Faculty, who have priority over Students.

Actors

- Student - A person that is enrolled in classes at the university.
- Faculty - A person that is employed by the university.
- Admin - A person that has the ability to create users and override the reservation of rooms.

ASP.net Terms

- Model - Class that stores the data from the database table with matching data-fields.
- View - Class that creates the visual interface.
- Controller - Manipulates the model and view classes to create the functionality of the app.

Class Terms

- BookingView - View that displays the form to book a room.
- BookingController - Controller that adds, alters, and removes bookings.
- BookingModel - Model that stores the data from the bookings table in the database.
- Time Slot - The specific time chunk for booking a room. Seperated into 1h 30min intervals to coincide with classes.
- CalendarView - View that displays the calendar information in a visual way to the user.
- CalendarController - Controller that converts the data from the CalendarModel into a format usable to the CalendarView.
- CalendarModel - Model that stores the data from the Room table of the database.
- LoginView - View that displays the form for a user to login.
- UserController - Controller that creates and destroys user sessions.
- UserModel - Model that stores the data from the users table in the database.
- Session - When a user logs in, the UserController creates a Session that allows that user to perform actions.

Operation Contracts

Group Members Guy Coccimiglio Justin Rhude Michael Thomas

Revision History

Version	Date	Description	Author
Inception draft	Feb 3, 2017	First draft.	Guy Coccimiglio

Contract C-01: startSession

Operation:
startSession(userID: int)
Cross References:
Use Cases: Book room, view bookings
Preconditions:
User has identified with student card or manual login
Postconditions:
A new Session instance was created

Contract C-02: requestBookRoom

Operation:
requestBookRoom(requestCode: int)
Cross References:
Use Cases: Book room, release room, view bookings
Preconditions:
A session is active for the user.
The requestCode is valid
Postconditions:
A RequestResponse is created
The RequestResponse is initialized with responseCode = CONFIRM

Contract C-03: enterRoomInfo

Operation:
enterRoomInfo(roomName: string, roomNumber: int)
Cross References:
Use Cases: Book room, view bookings
Preconditions:
Request has been confirmed.
Postconditions:
System has identified the Room[] required

Contract C-04: enterDateInfo

Operation:
enterDateInfo (date: Date)
Cross References:
Use Cases: Book room, view bookings
Preconditions:
Request has been confirmed.
Postconditions:
System has identified the Date[] required

Contract C-05: releaseCurrentRoom

Operation:
releaseCurrentRoom (roomName: string, roomNumber: int)
Cross References:
Use Cases: Release room
Preconditions:
Current room has been selected by user and identified by the system
Postconditions:
Booking information is removed from the instance of Room
The Booking object is destroyed
The system begins broadcasting the release

Contract C-06: broadcastReleaseRoom

Operation:
broadcastReleaseRoom(roomName: string, roomNumber: int, timeslot: Timeslot)
Cross References:
Use Cases: Release room
Preconditions:
Booking object has been destroyed in one session for this room in the given timeslot
Postconditions:
A new Response instance is created
The response is sent to all active sessions

Contract C-07: lockRoom

Operation: lockRoom(id: int)
Cross References:
Use Cases: Book room
Preconditions:
Booking request has been confirmed
Postconditions:
A new Response instance is created
The response is initialized with lock roomCode
The response is sent to all active sessions
The room is marked unavailable to other sessions

Contract C-08: terminateSession

Operation: terminateSession (userID: int)
Cross References:
Use Cases: Book room, view bookings, release room
Preconditions:
The user has completed all desired tasks
User is logged out
Postconditions:
The session instance is destroyed
The session instance is removed from sessionManager

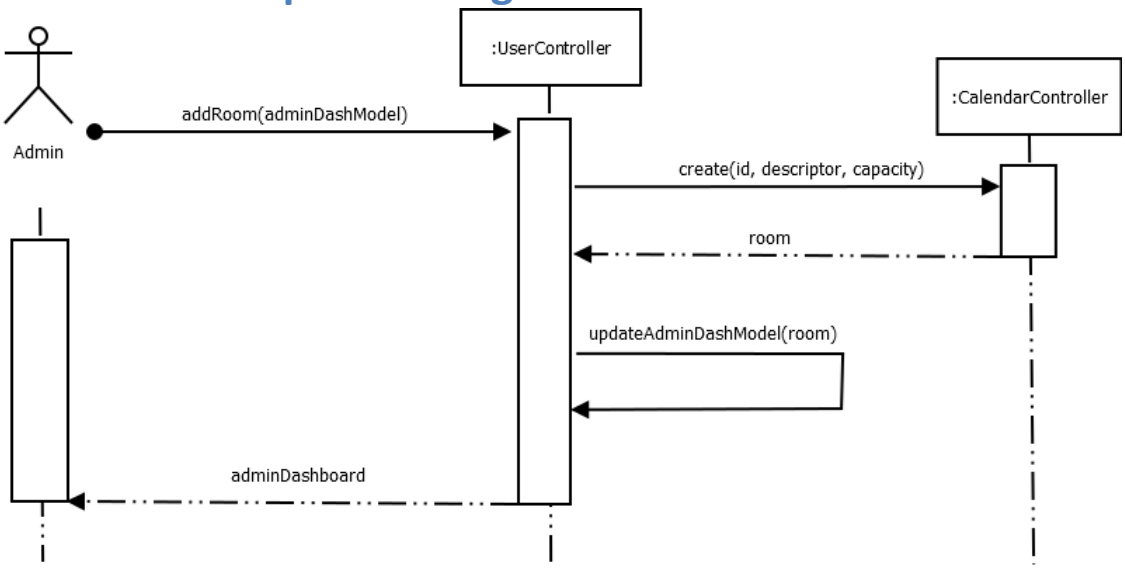
Sequence Diagrams

Group Members - Guy Coccimiglio Justin Rhude Michael Thomas

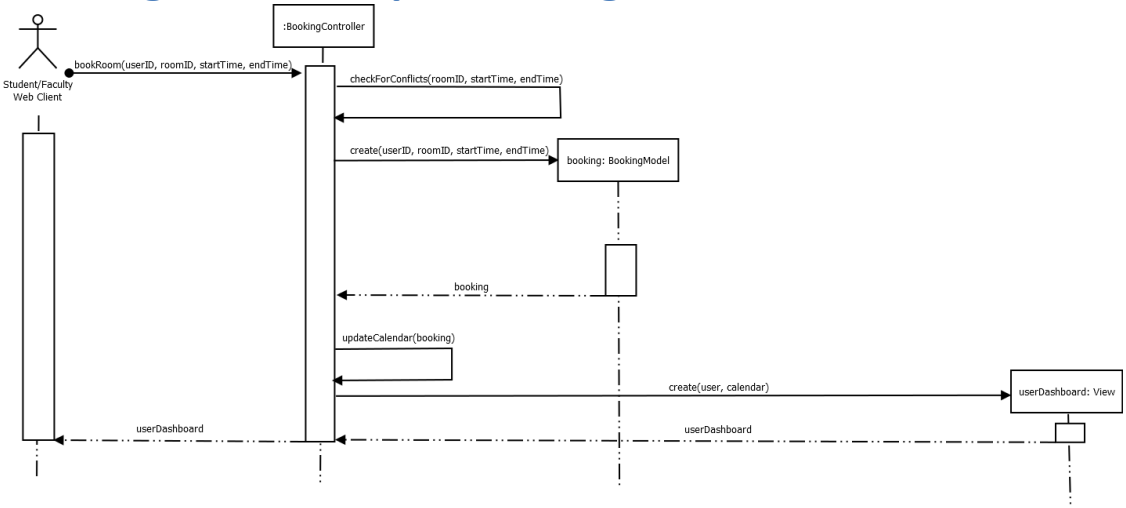
Revision History

Version	Date	Description	Author
First Revision	Mar 15, 2017	Final Version.	Guy Coccimiglio

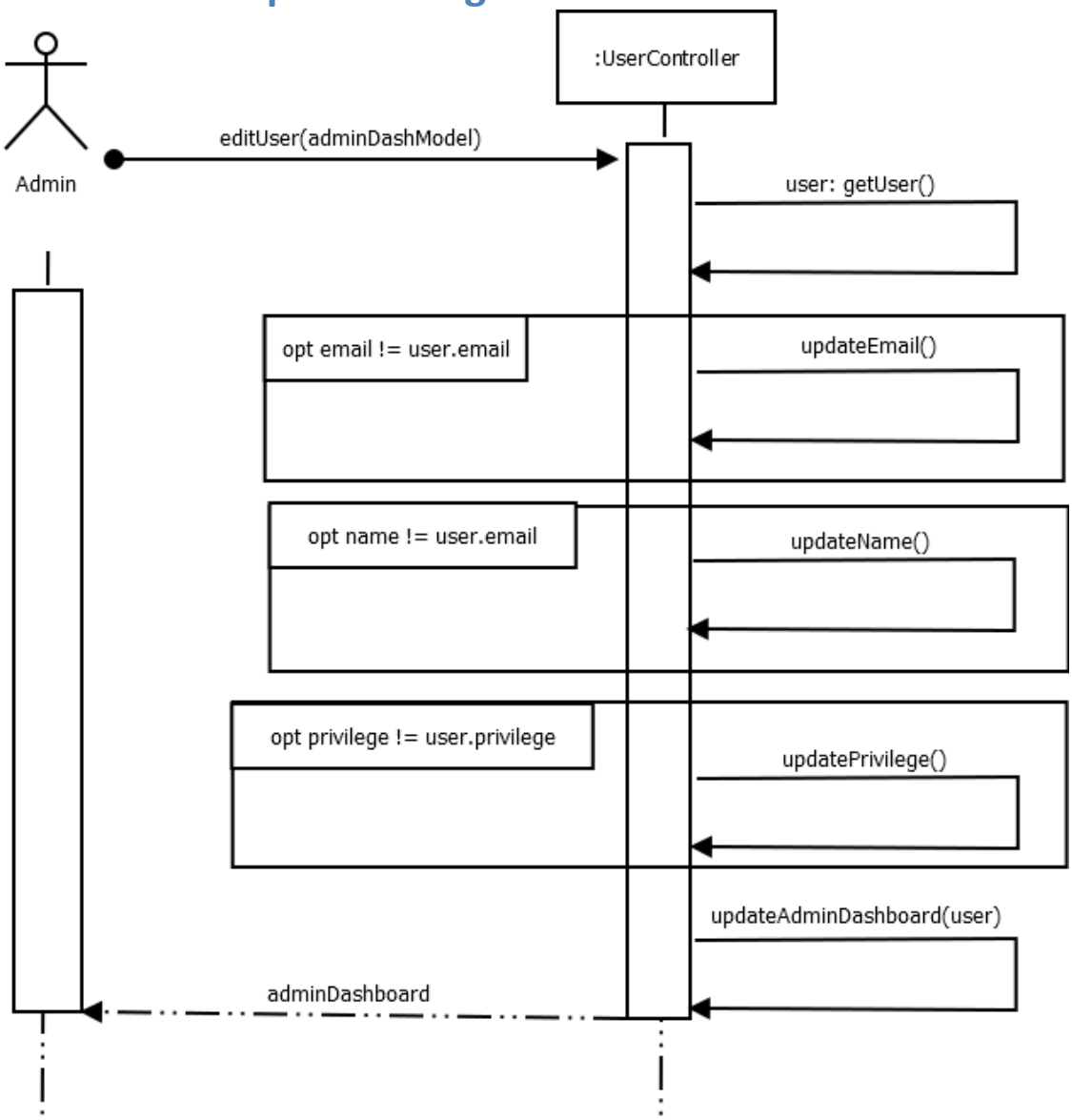
Add User - Sequence Diagram



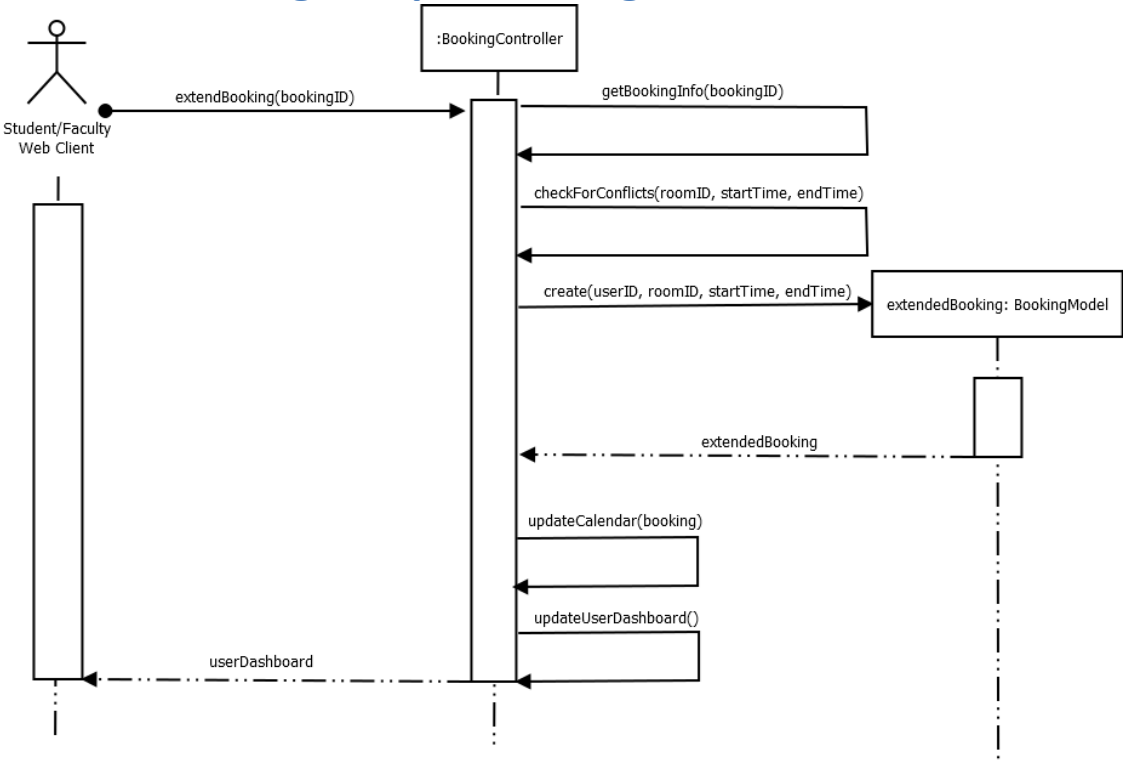
Booking Room - Sequence Diagram



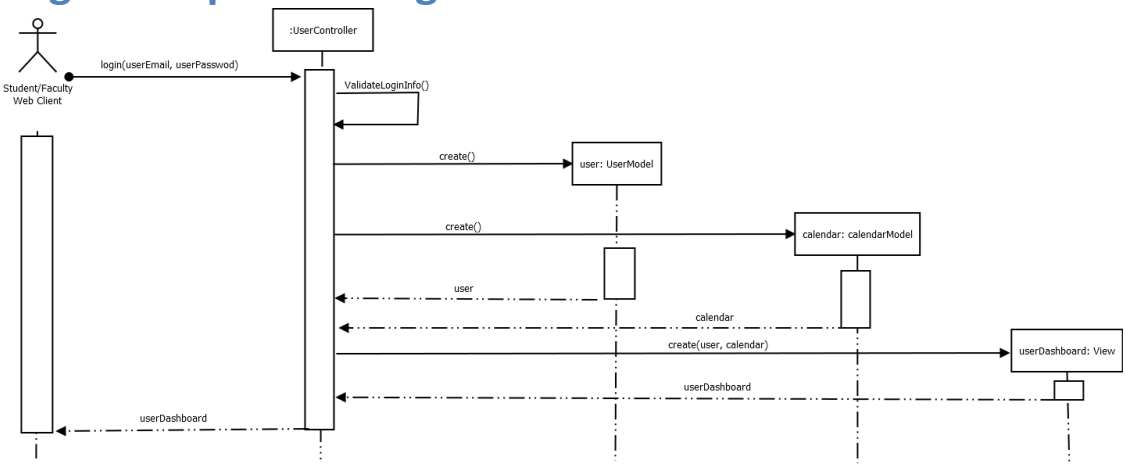
Edit User - Sequence Diagram



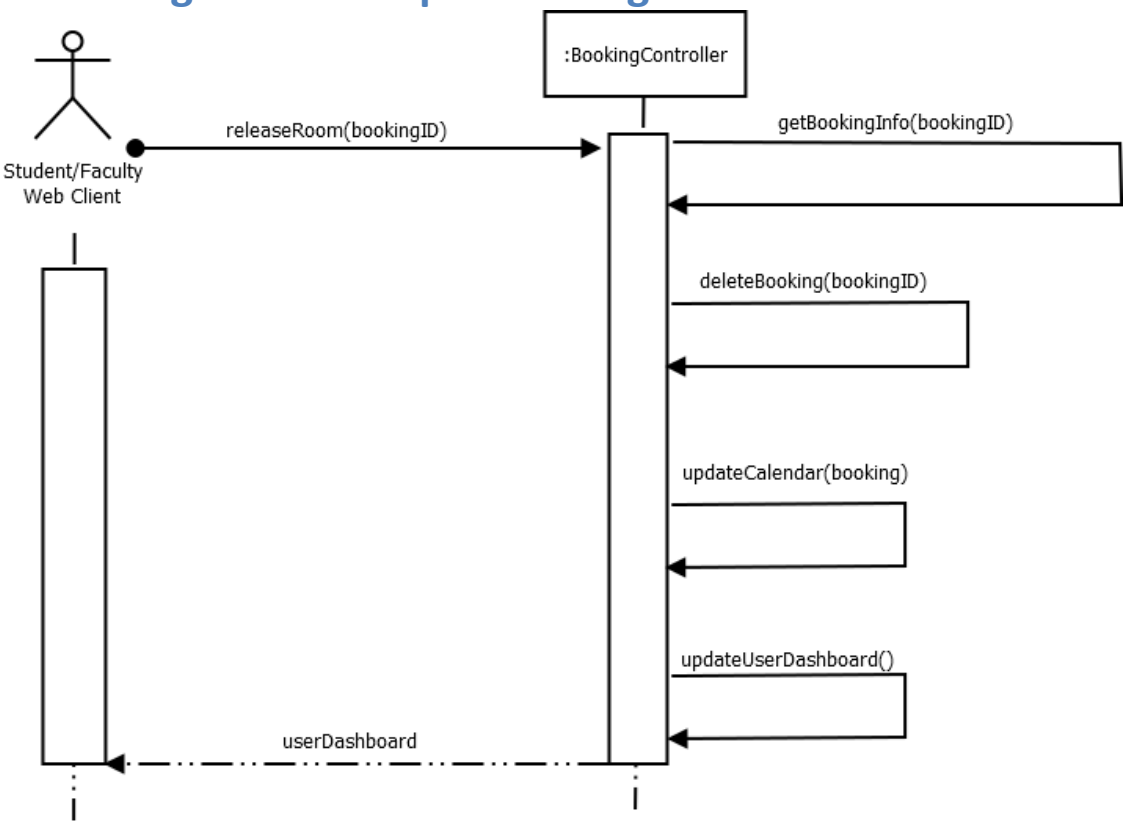
Extend Booking - Sequence Diagram



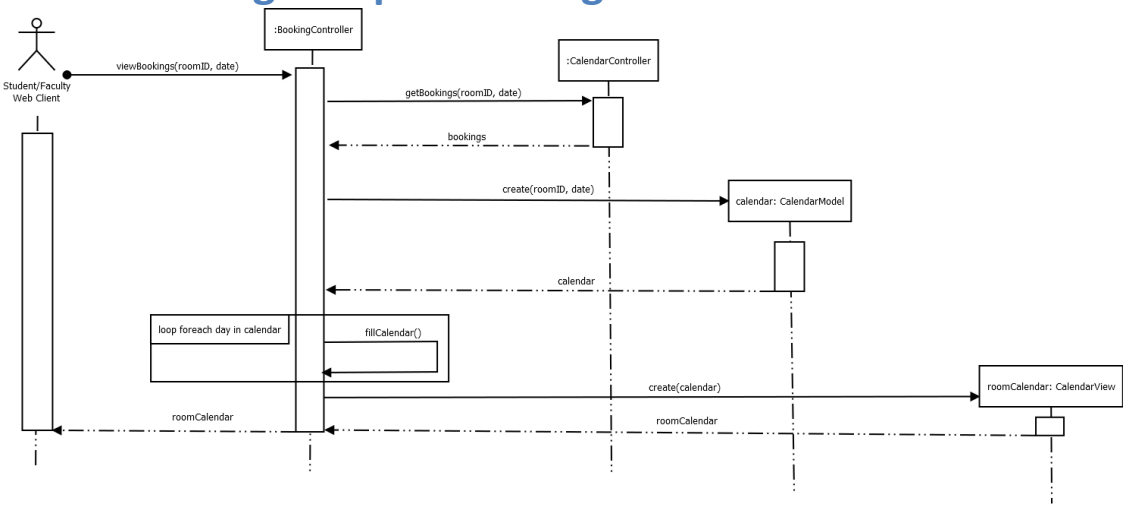
Login - Sequence Diagram



Releasing Room - Sequence Diagram



View Booking - Sequence Diagram



Supplementary Specification Document

Group Members Guy Coccimiglio Justin Rhude Michael Thomas

Revision History

Version	Date	Description	Author
Inception draft	Feb 25, 2017	First draft.	Michael Thomas

Introduction

This document is the repository for all the requirements not captured in the use cases; specifically non-functional requirements.

Functionality

Logging and Error Handling

Errors handling will be automated, specifically booking conflict resolution. This will be done to reduce human interaction and human error possibilities.

Security

User information will be fully secure in the system, such as login usernames and passwords.

Usability

Human Factors

The customer will be able to interact with the system through a calendar GUI which will - Allow for ease of use: click on the date you wish to book, select the time, etc. - Have text that is easy to read; big enough for those who have trouble reading small text. - Use a color scheme that is pleasant to the eye and doesn't make the text hard to read.

Reliability

The system will have live updating of booking information, which makes the information the customer views completely accurate, as well as reduces booking conflicts that could arise.

Performance

The customer will want to be able to book a room, release a room, or view information as fast as possible. The system will allow the user to complete these simple tasks in a very short amount of time. Target time is less than a minute.

Supportability

The system uses a web based interface, allowing it to be 100% cross platform.

Interfaces

Noteworthy Hardware and Interfaces

- Any modern device with internet access (computers, cellphones, etc.)
- Terminals outside of rooms.

Software Interfaces

- Will use a calendar GUI, web based interface that will allow for many devices to use it. This includes touch-based as well as mouse and keyboard interactivity.

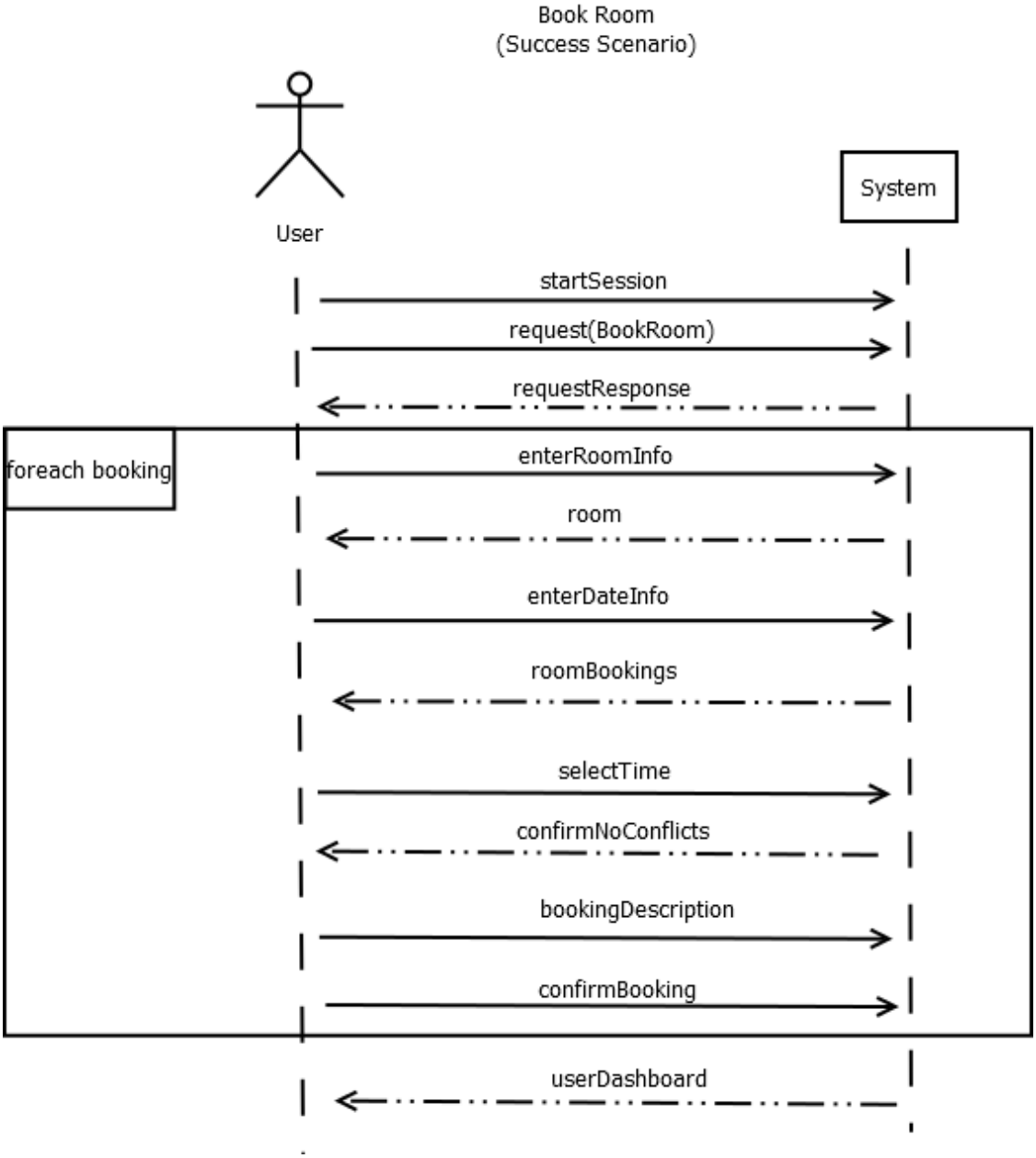
System Sequence Diagrams

Group Members - Guy Coccimiglio Justin Rhude Michael Thomas

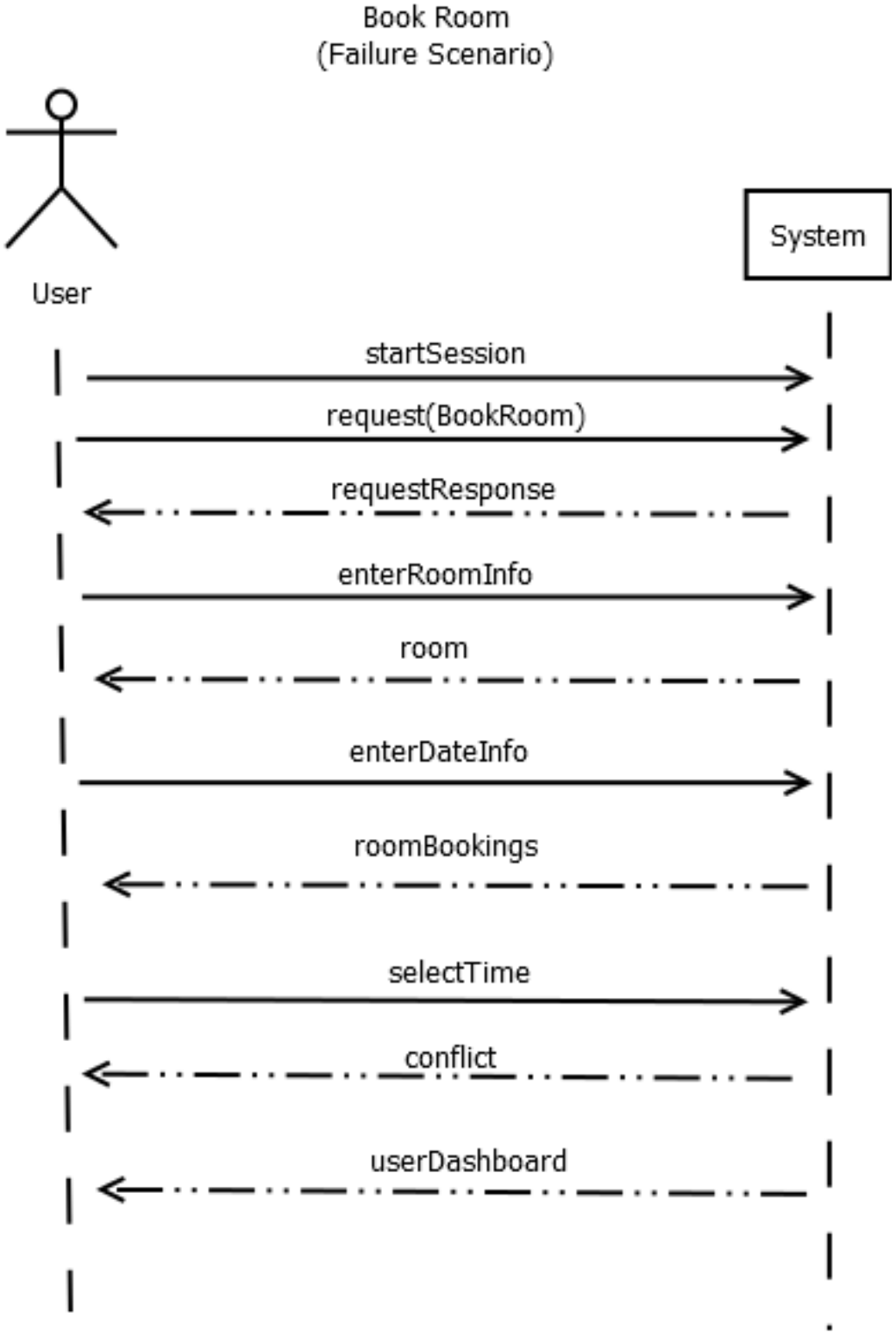
Revision History

Version	Date	Description	Author
Inception draft	Feb 17, 2017	First draft.	Guy Coccimiglio
Second Revision	Mar 15, 2017	Final Version.	Justin Rhude

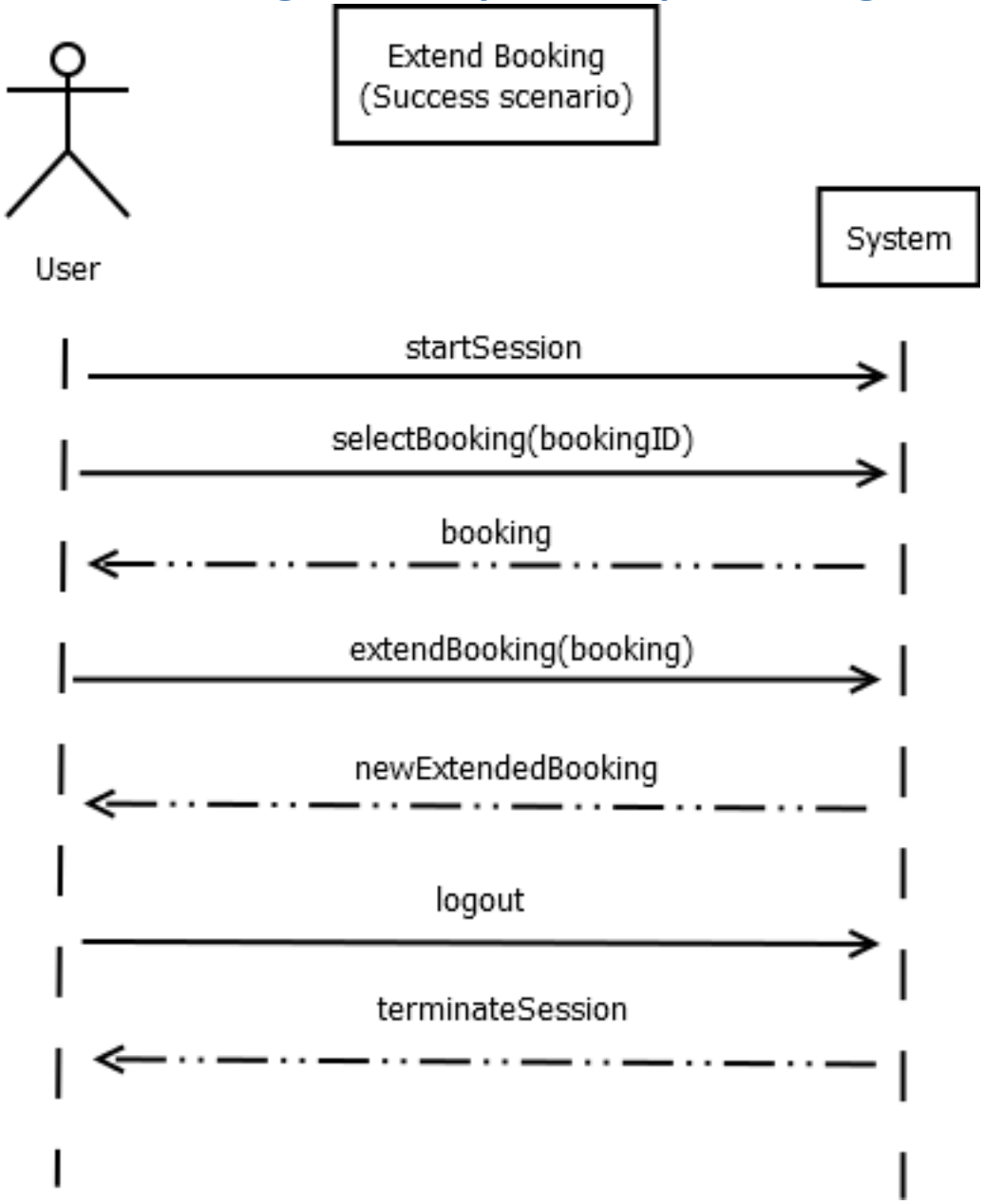
Booking Room - System Sequence Diagram



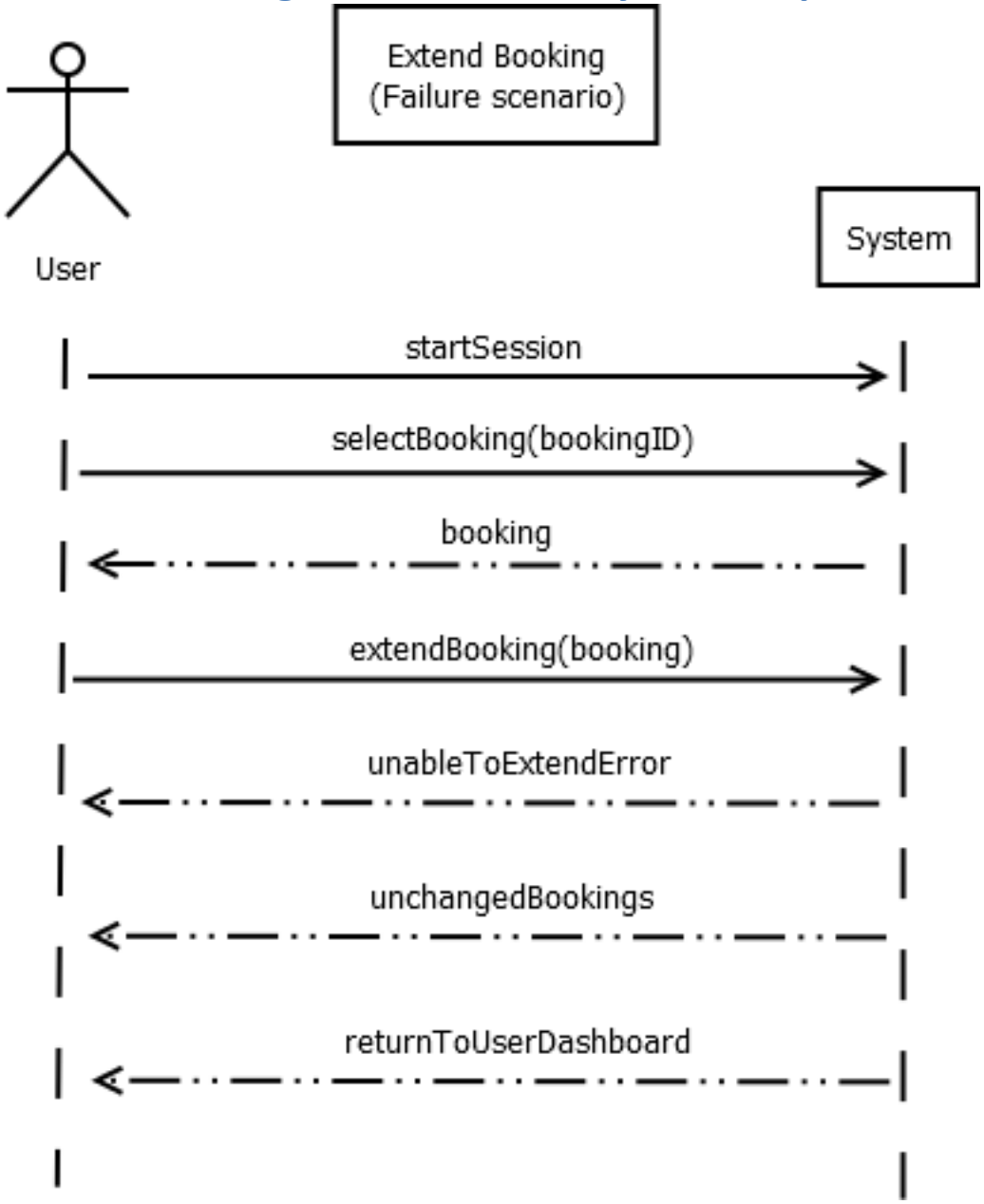
Booking Room Failure - System Sequence Diagram



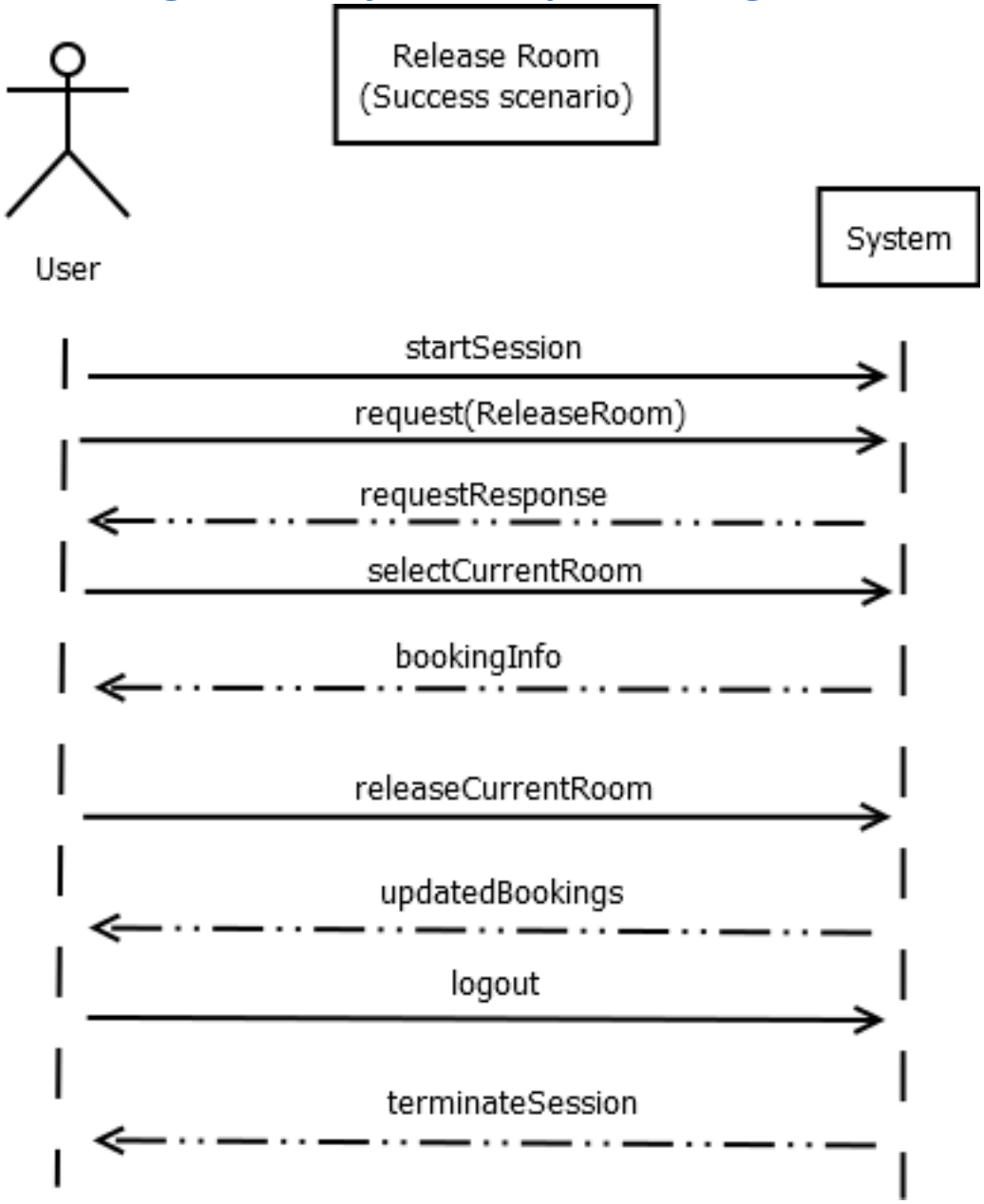
Extend Booking Room - System Sequence Diagram



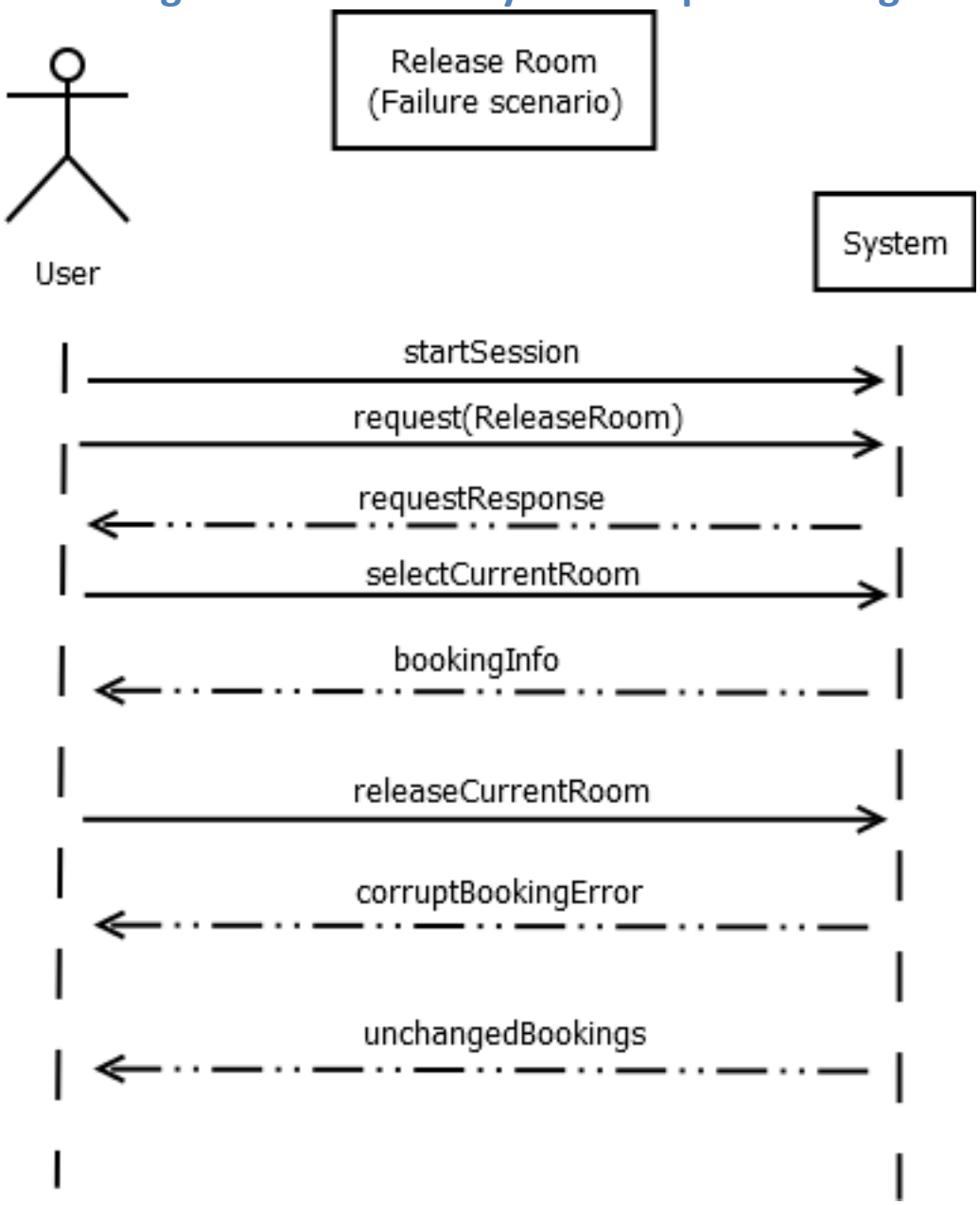
Extend Booking Room Failure - System Sequence Diagram



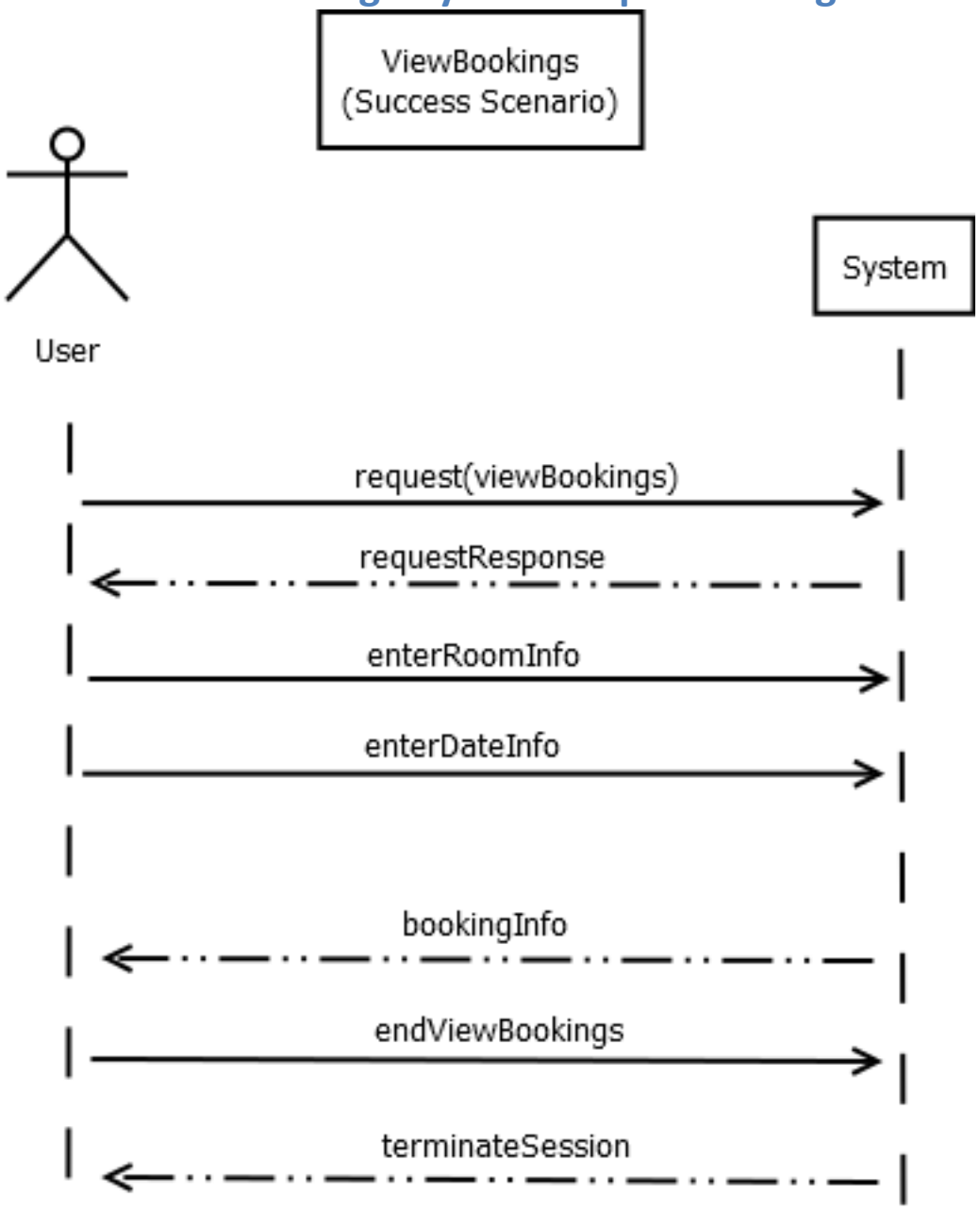
Releasing Room - System Sequence Diagram



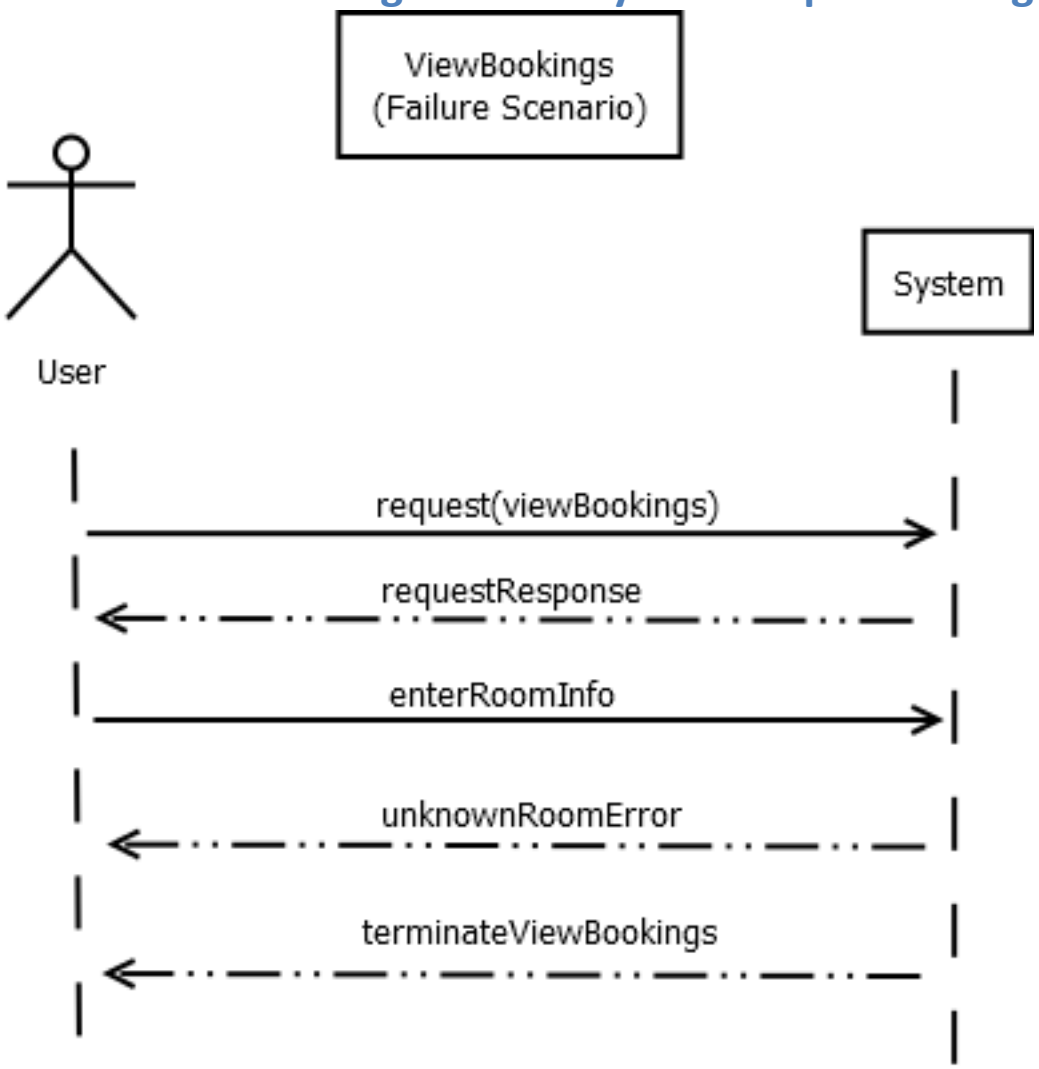
Releasing Room Failure - System Sequence Diagram



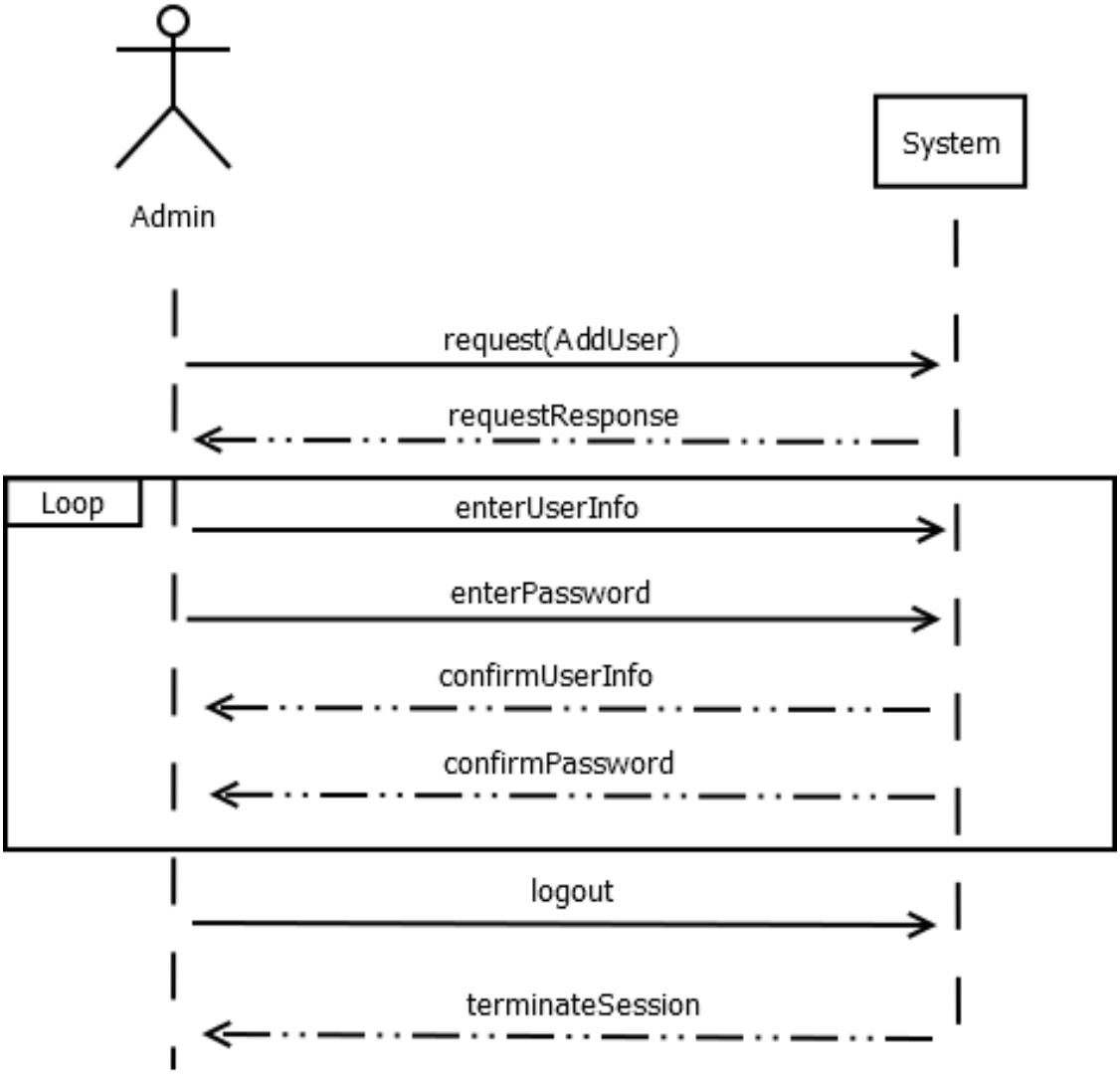
View Room Booking - System Sequence Diagram



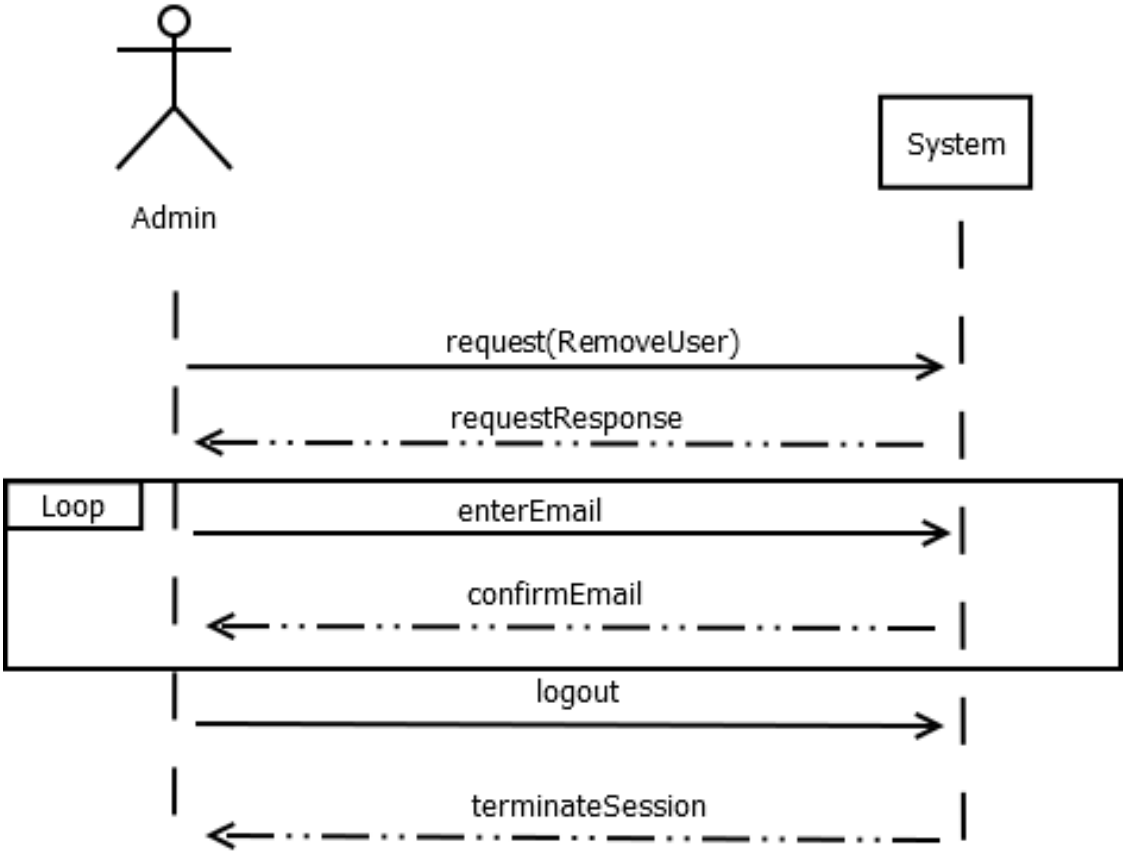
View Room Booking Failure - System Sequence Diagram



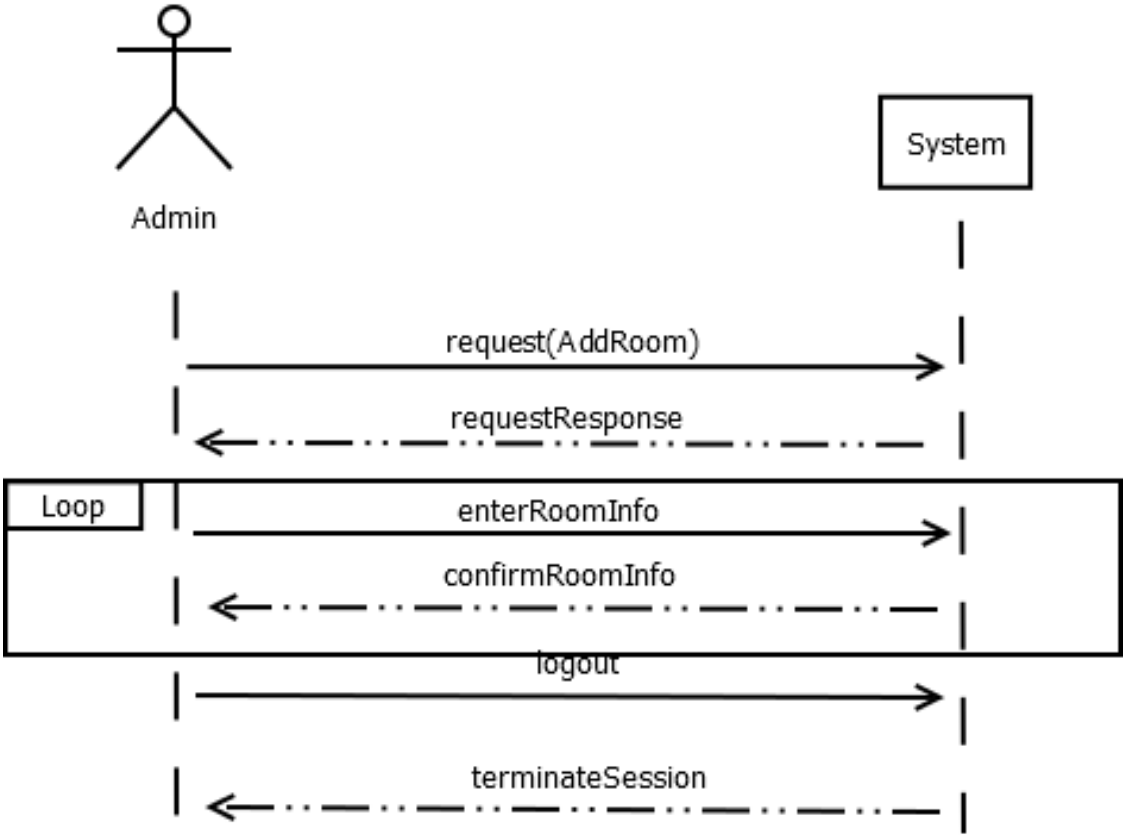
Add User - System Sequence Diagram



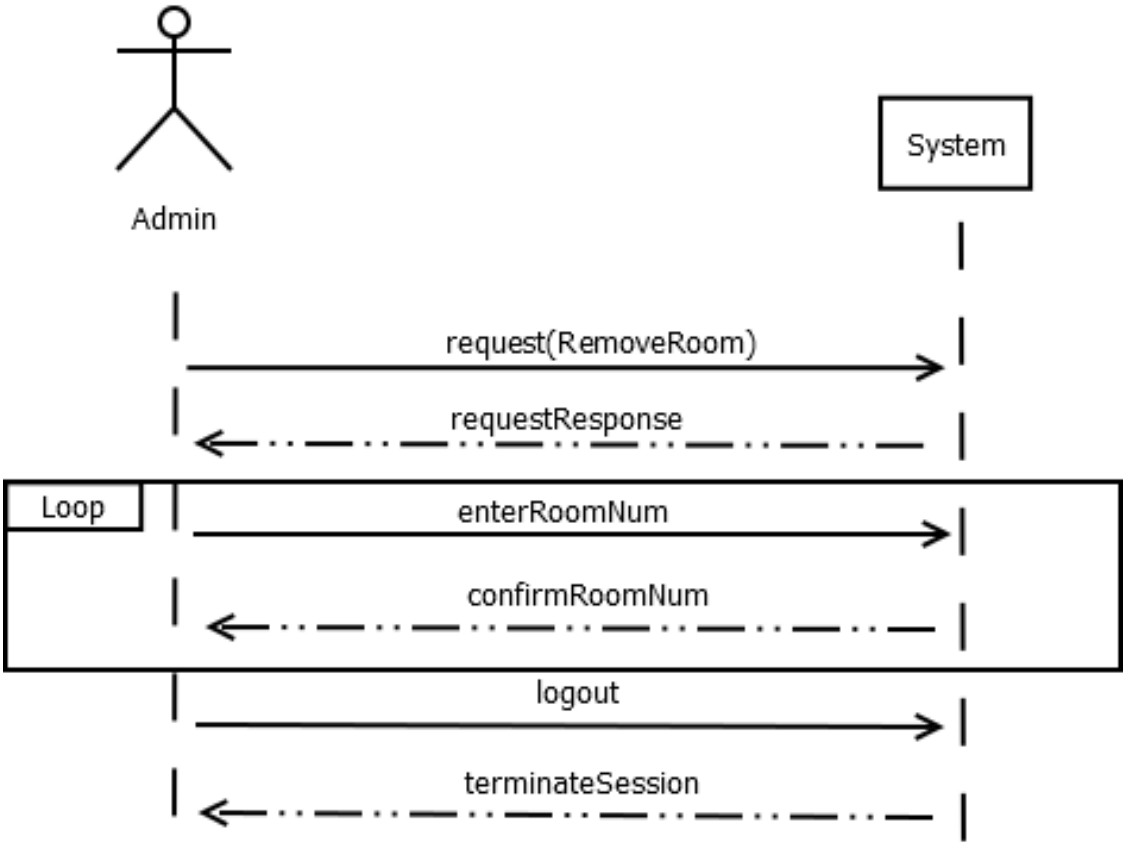
Remove User - System Sequence Diagram



Add Room - System Sequence Diagram



Remove Room - System Sequence Diagram



Testing Document

Revision History

Version	Date	Description	Author
Testing Phase	Feb 12, 2017	First draft.	Michael Thomas

Functionality Tested

The functionality we chose to test was booking rooms. We decided that it would be the best thing to test since it is the most important part of the application. The software is meant to allow users to book rooms as its primary function, so it would be the essential function to test and get running perfectly.

Partitioning Testing

We chose to restrict the majority of user input throughout our application. The nature of web based programming furthers the need to us to be very cautious with what inputs we allow to be entered into the site. With that said we will provide the following equivalence classes for our partitioning testing. These equivalence classes represent valid and invalid inputs to the variable fields that we provide for the user. The first step to booking a room is logging in. Our first test case analyzes the login process

Test Case 1 (LG1)

A valid email and password is required for logging in. Here we will test the functionality of an invalid email. The email used is invalid but the password is valid for jon@test.com.

Email:	Password
jonny@test.com	testHash123

Test Case 2 (LG2)

Here we will test the functionality of an invalid password with a valid email.

Email:	Password
jon@test.com	garbagePassword

Test Case 3 (LG3)

The third login test combines both a valid email and valid login to ensure they work properly together

Email:	Password
jon@test.com	testHash123

The following test cases occur after a successful login. The next step is searching for a room. This can be done by searching based on parameters or by room id.

Test Case 4 (RS1)

Here we test searching for a room via room id. This test case uses an invalid room id.

Room ID
10000

Test Case 5 (RS2)

Again, we test searching for a room via room id. This test case uses an valid room id.

Room ID
201

Test Case 6 (RS3)

Here we test searching for a room via min capacity and room type. It is impossible to choose an invalid room type because we use a dropdown menu with set options all of which are acceptable, likewise you cannot enter an invalid capacity because we have a fixed ranged of inputs set. We will therefore only a search that should and shouldn't result in rooms. Here we test with inputs that should not return a room.

Capacity	Room Type
10	Bio Lab (we have no rooms of this type yet)

Test Case 7 (RS4)

Now we will test with a inputs that should return a room.

Capacity	Room Type
10	Computer Lab

After this step all input options are locked to valid options. It is impossible to perform any actions that are invalid from this point until the room is booked.

Partitioning Testing Results

Test Case	Expected Output	Actual Output
LG1	Login Error	Login Error
LG2	Login Error	Login Error
LG3	Successful Login	Successful Login
RS1	Invalid Room ID Error	Invalid Room ID Error
RS2	Successful search	Successful search
RS3	Unsuccessful search	Unsuccessful search
RS4	Successful search	Successful search

Boundary Value Testing

There are not many places where we allow inputs that can reach edge cases. However, we did come across some areas where testing near the boundaries of valid input was necessary. These include the following:

Test Case 1 (CC)

The calendar can cycle through months and years. The functionality used to implement this reaches a boundary on the first and last month of the year. So we tested changing between years on the calendar (Note this does not require any data input only interaction with UI elements by the user). For example going from December 2017 to January 2018.

Extending rooms was the other area where boundary testing was necessary. There are 2 scenarios where boundaries are reached. When the user tries to extend past 10pm – which we do not allow or when the user tries to extend a booking that has no available space after that booking.

Test Case 2 (ER1)

Here we attempt to extend the booking past 10pm.

Test Case 3 (ER2)

Here we attempt to extend into a space that is already taken.

Boundary Testing Results

Test Case	Expected Output	Actual Output
CC	Calendar month Jan 2018	Calendar month Jan 2018
ER1	Invalid booking time	Invalid booking time
ER2	Booking conflict	Booking conflict

Use Case: Booking a Room (Admin)

Group Members - Guy Coccimiglio Justin Rhude Michael Thomas

Revision History

Version	Date	Description	Author
Inception draft	Mar 17, 2017	First draft.	Justin Rhude

Scope:

Room Booking application

Level:

User Goal

Primary Actor:

Admin

Stakeholders and Interests:

- Students: Wants quick and easy access, automatic conflict resolution, and live updates.
- Faculty: Wants to be able to book rooms on a weekly basis,
- Admin: Wants an easy to use interface to manage users and room bookings, wants to be able to add new rooms easily.
- School Registrar: Wants to be able to keep track of what rooms are being used, and what rooms are free.

Preconditions:

Admin has identified with a login or student card.

Success Guarentee(or Postconditions):

Admin logs out after completing the booking, conflicts are resolved successfully, confirmation showing successful room booking is displayed.

Main Success Scenario (or Basic Flow):

1. Admin starts new room booking session.
2. Admin selects the room and the date they wish to reserve it.
3. System locks the time slot in order to prevent conflicts.
4. System retrieves the current bookings for the given room and date, and displays the information to the student.
5. Admin selects a slot to reserve and the amount of time they need it for.
6. System confirms the chosen time. If there is already a booking, system confirms that the admin wishes to override.
7. System displays a confirmation of the room being successfully booked.
8. Admin can choose to book other rooms or times using this session.
9. Admin logs out.

Use Case: Booking a Room (Faculty)

Group Members - Guy Coccimiglio Justin Rhude Michael Thomas

Revision History

Version	Date	Description	Author
Inception draft	Mar 17, 2017	First draft.	Justin Rhude
Second Iteration	Mar 21, 2017	First draft.	Guy Coccimiglio

Scope:

Room Booking application

Level:

User Goal

Primary Actor:

Faculty

Stakeholders and Interests:

- Students: Wants quick and easy access, automatic conflict resolution, and live updates.
- Faculty: Wants to be able to book rooms on a weekly basis,
- Admin: Wants an easy to use interface to manage users and room bookings, wants to be able to add new rooms easily.
- School Registrar: Wants to be able to keep track of what rooms are being used, and what rooms are free.

Preconditions:

Faculty has identified with a login or student card.

Success Guarantee(or Postconditions):

Faculty logs out after completing the booking, conflicts are resolved successfully, confirmation showing successful room booking is displayed.

Main Success Scenario (or Basic Flow):

1. Faculty starts new room booking session.
2. Faculty selects the room and the date they wish to reserve it.
3. System retrieves the current bookings for the given room and date, and displays the information to the user.
4. Faculty selects an open slot to reserve and the amount of time they need it for.
5. System confirms that there are no conflicts and books the room for the given times.
6. System displays a confirmation of the room being successfully booked.
7. Faculty can choose to book other rooms or times using this session.
8. Faculty logs out.

Main Failure Scenario:

1. User starts new room booking session.
2. User selects the room and the date they wish to reserve it.
3. System retrieves the current bookings for the given room and date, and displays the information to the user.
4. User selects an open slot to reserve and the amount of time they need it for.
5. System confirms finds a conflict with the booking.
6. System prompts user to try to book again.
7. System returns to previous booking options.
8. User retries or logs out.

Use Case: Booking a Room (Student)

Group Members - Guy Coccimiglio Justin Rhude Michael Thomas

Revision History

Version	Date	Description	Author
Inception draft	Feb 10, 2017	First draft.	Justin Rhude

Scope:

Room Booking application

Level:

User Goal

Primary Actor:

Student

Stakeholders and Interests:

- Students: Wants quick and easy access, automatic conflict resolution, and live updates.
- Faculty: Wants to be able to book rooms on a weekly basis,
- Admin: Wants an easy to use interface to manage users and room bookings, wants to be able to add new rooms easily.
- School Registrar: Wants to be able to keep track of what rooms are being used, and what rooms are free.

Preconditions:

Student has identified with a login or student card.

Success Guarentee(or Postconditions):

Student logs out after completing the booking, conflicts are resolved successfully, confirmation showing successful room booking is displayed.

Main Success Scenario (or Basic Flow):

1. Student starts new room booking session.
2. Student selects the room and the date they wish to reserve it.
3. System locks the time slot in order to prevent conflicts.
4. System retrieves the current bookings for the given room and date, and displays the information to the student.
5. Student selects an open slot to reserve and the amount of time they need it for.
6. System confirms that there are no conflicts and books the room for the given times.
7. System displays a confirmation of the room being successfully booked.
8. Student can choose to book other rooms or times using this session.
9. Student logs out.

Main Failure Scenario:

1. User starts new room booking session.
2. User selects the room and the date they wish to reserve it.
3. System retrieves the current bookings for the given room and date, and displays the information to the user.
4. User selects an open slot to reserve and the amount of time they need it for.
5. System confirms finds a conflict with the booking.
6. System prompts user to try to book again.
7. System returns to previous booking options.
8. User retries or logs out.

Use Case: Releasing a Room

Group Members - Guy Coccimiglio Justin Rhude Michael Thomas

Revision History

Version	Date	Description	Author
Iteration 2	March 20, 2017	First draft.	Guy Coccimiglio

Scope:

Room Booking application

Level:

User Goal

Primary Actor:

Student

Stakeholders and Interests:

- Student/Faculty: Wants to be able to extend a booking into the next time slot if their meeting/activity requires more time users.

Preconditions:

Student/Faculty has identified with a login or student card. Student already has a booking for the time-slot prior to the extension.

Success Guarantee(or Post-conditions):

A new booking is created with starting time equal to the end time of the previous booking and length of the standard 1.5 hours.

Main Success Scenario (or Basic Flow):

1. User logs into their account.
2. System retrieves the active bookings held by the user.
3. Student selects the desired booking from their list of active bookings.
4. Student selects to extend the booking.
5. System creates a new booking in the time slot that follows the extended booking.
6. System automatically fills in the description for the new booking.
7. System updates the calendar and booking list with the new booking.
8. User logs out.

Main Failure Scenario:

1. User logs into their account.
2. System retrieves the active bookings held by the user.
3. Student selects the desired booking from their list of active bookings.
4. Student selects to extend the booking.
5. System find a conflict with extending the booking
6. System prints error message to the user.
7. System returns the user to the original dashboard.
8. User logs out.

Use Case: Manage Users

Group Members - Guy Coccimiglio Justin Rhude Michael Thomas

Revision History

Version	Date	Description	Author
Inception draft	Mar 17, 2017	First draft.	Michael Thomas

Scope:

Room Booking application

Level:

User Goal

Primary Actor:

System Administrator.

Stakeholders and Interests:

- Student: Wants their information to be able to be edited if necessary.
- Faculty: Wants system to be able to add and remove users.
- Admin: Wants to be able to add, remove, or edit users of the system.

Preconditions:

The system administrator has logged into the system

Success Guarantee(or Post-conditions):

The user has left the manage user section of the website after making the necessary changes. The admin logs out successfully

Main Success Scenario (or Basic Flow):

1. The admin logs in.
2. The admin accesses the manage user portion of the system.
3. The admin selects either new user, edit user, or remove user.
4. If they selected new user or edit user, they input the necessary information.
5. The admin can then manage other users during the session.
6. The admin exits the manage user section.
7. The admin logs out.

Use Case: Releasing a Room

Group Members - Guy Coccimiglio Justin Rhude Michael Thomas

Revision History

Version	Date	Description	Author
Inception draft	Feb 11, 2017	First draft.	Michael Thomas
Iteration 2	March 20, 2017	Second draft.	Guy Coccimiglio

Scope:

Room Booking application

Level:

User Goal

Primary Actor:

Student

Stakeholders and Interests:

- Students: Wants to be able to release a room when they are finished, and to be able to get a room if it is empty.
- Faculty: Wants room booking to be as efficient as possible; rooms that are not being used should be released to other users.
- Admin: Wants releasing a room to be as user friendly as possible.
- School Registrar: Wants users to know that releasing a room should be done when finished early.

Preconditions:

Student has identified with a login or student card. Student has finished using the room early.

Success Guarantee(or Post-conditions):

The user has left the view bookings section of the website after releasing the room they have booked. The student logs out successfully

Main Success Scenario (or Basic Flow):

1. User logs into their account.
2. System retrieves the active bookings held by the user.
3. Student selects the current booking from their list of active bookings.
4. Student selects to release the room from booking.
5. System removes the booking from the list of bookings
6. System returns the user to their dashboard
7. User logs out.

Main Failure Scenario:

1. User logs into their account.
2. System retrieves the active bookings held by the user.
3. Student selects the current booking from their list of active bookings.
4. Student selects to release the room from booking.
5. System fails to remove the booking from the list of bookings
6. System ensures original bookings are not altered
7. System returns the user to their dashboard
8. User can try again or log out.

Use Case: View Bookings

Group Members - Guy Coccimiglio Justin Rhude Michael Thomas

Revision History

Version	Date	Description	Author
Inception draft	Feb 10, 2017	First draft.	Guy Coccimiglio

Scope:

Room Booking application

Level:

User Goal

Primary Actor:

General User (Student/Faculty/Administrator)

Stakeholders and Interests:

- Students: Wants to quickly determine if a room is booked
- Faculty: Wants to quickly find a suitable empty room
- Admin: Wants to see the status of all rooms at any given time
- School Registrar: Wants to be able to keep track of what rooms are being used, and what rooms are free.

Preconditions:

The user has navigated to the booking application website

Success Guarentee(or Postconditions):

The user has left the view bookings section of the website after finding the status of the room they are interested in.

Main Success Scenario (or Basic Flow):

1. User enters the view bookings section of the site
2. User provides specific room number of query information to identify one or more rooms
3. User provides the date(s) which they are interested in viewing

4. System retrieves the current bookings for the given room(s), and displays the information to the user.
5. User can alter the query data to filter the results.
6. System terminates the session when the user exits the site.

Extensions (or Alternative Flows):

A. Spontaneous live updates of booking information may be required in the event that a different user booked one or more of the rooms that are currently being viewed by a different user.

1. User enters the view bookings section of the site
2. User provides specific room number of query information to identify one or more rooms
3. User provides the date(s) which they are interested in viewing
4. System retrieves the current bookings for the given room(s), and displays the information to the user.
5. System receives update notice
6. System updates bookings to reflect the changes
7. System informs the user that a change has occurred
8. System terminates the session when the user exits the site.

B. A user can choose to view only their own bookings when they are logged into the system.

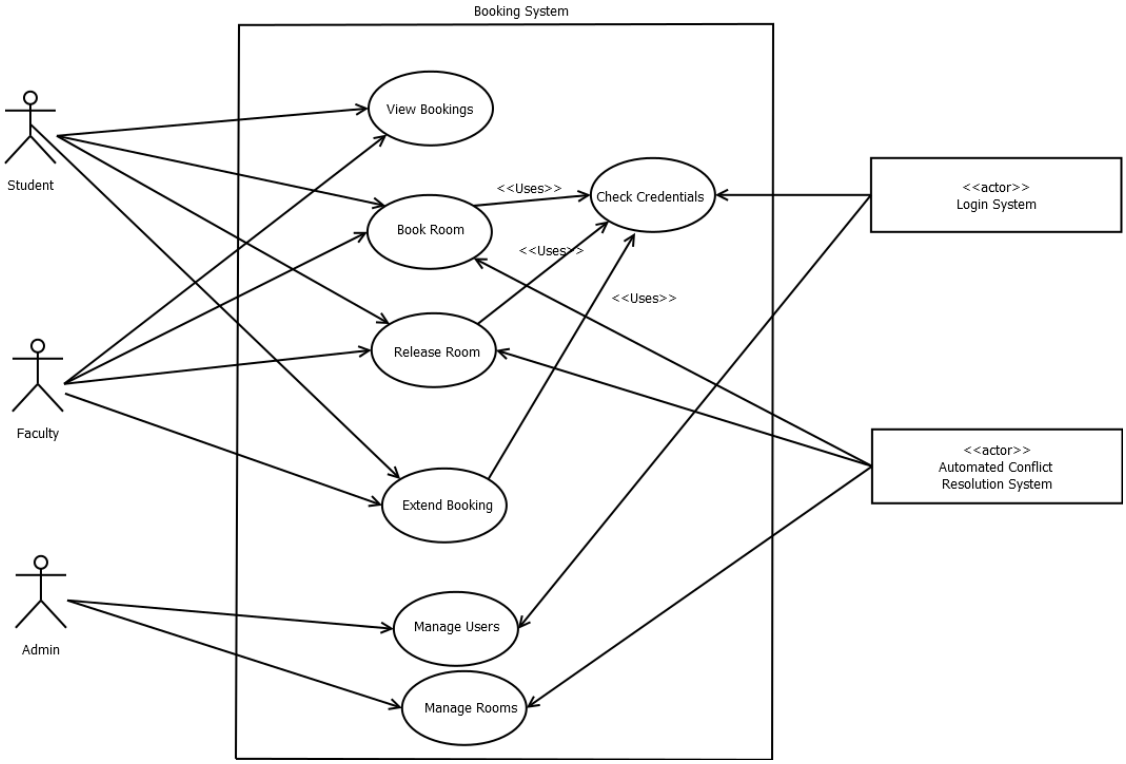
1. User enters the view bookings section of the site
2. User logs into their account.
3. User selects to view their own bookings.
4. System retrieves the bookings reserved by the user, and displays the information to the user.
5. User can alter the query data to filter the results.
6. System terminates the session when the user exits the site.

Use Case Diagram

Group Members - Guy Coccimiglio Justin Rhude Michael Thomas

Revision History

Version	Date	Description	Author
Inception draft	Feb 18, 2017	First draft.	Michael Thomas
Second Revision	Mar 20, 2017	Final Version.	Michael Thomas



Vision Document

Group Members Guy Coccimiglio Justin Rhude Michael Thomas

Revision History

Version	Date	Description	Author
Inception draft	Feb 3, 2017	First draft.	Guy Coccimiglio

Introduction

We envision a user friendly room booking application where the user interacts with a calendar GUI to easy view bookings and reserve rooms. This application will be usable on all modern web browsers and focuses on efficient and effective live updating of room booking information.

Positioning

Business Opportunity

Currently the room booking system used at our university is not easy to use. It is primarily text based and has very minimal features. It requires a helpdesk employee to manually deal with bookings and the responses to requests are not live. This is a poor system for a simple task. We believe that students and faculty would benefit from a more robust and user friendly room booking system.

Problem Statement

Room booking systems often require an immense amount of human interaction to process requests which is time consuming and costly. Moreover they rarely format the booking information in a manner that is easy to read and interact with. These systems also offer little to no hierarchy of user privileges which means they rely on some form of super user or administrator. These limitations make the system hard to integrate into a real working environment. Moreover these limitations make these room booking systems impossible to implement via a user facing terminal located at the entrance to the rooms. This is problematic for maintenance crews, general users, faculty and administrators.

Product Position Statement

Our room booking system is targeted for use by faculty and students of universities or members of a similar organization. Our system will incorporate a full web based GUI that can be integrated into the infrastructure of any modern device with internet access capabilities. We will also offer a controlled hierarchy of users to avoid bottlenecking the functions of the system with a single administrator. The room booking system will also be full automated for general users. If a room is available they can immediately book the room themselves without the need for further human intervention.

Alternatives and Competition

Our primary example for comparison is the lab booking system that currently exists at Algoma University. This system offers almost none of the features that we plan to include. It is primarily text based and requires a large amount of human interaction in order to function.

Key High-Level Goals and Problems of the Stakeholders High-Level Goal | Priority | Problems and Concerns | --- | --- | --- Live Updating of room booking information | High | Conflicts will occur if updates are not sent to all users. Live updates need to be fast and efficient to ensure low quality devices do not lag behind. User friendly web based GUI | High | We must support all major web browsers. Mobile accessibility will be a challenge. User privilege hierarchy | High | If privilege level system fails malicious users can harm the system. Bookings can be disrupted. Efficient use of rooms | Medium | Efficient use of rooms will lead to less conflicts. Automating this system will be difficult if users have restrictive preferences. Booking conflict resolution | Medium | Conflicts are possible but unlikely. We must offer a fast resolution system to deal with conflicts as soon as possible. The time constraint on this process will be a problem.
Room criteria search filtering | Low | We will need a structured description of rooms to allow for standard search criteria. Searching for nonexistent rooms will be an issue.

User-Level Goals

- Students – view booked rooms, book available rooms with low priority
- Faculty – view booked rooms, book available rooms with high priority
- Administrator – manage users, override room bookings, view booked rooms, manage bookings

User Environment

All users will interact with the system via a web client. Each privilege level will have a different set of features available after logging in. All users must log in to book rooms but anyone can view booked rooms without logging in.

Product Overview

Product Perspective

For our purposes the system will reside entirely online. The system could be integrated to work on any device with internet access. The end goal is to have the system available from user facing terminals located at the entrance of each room.

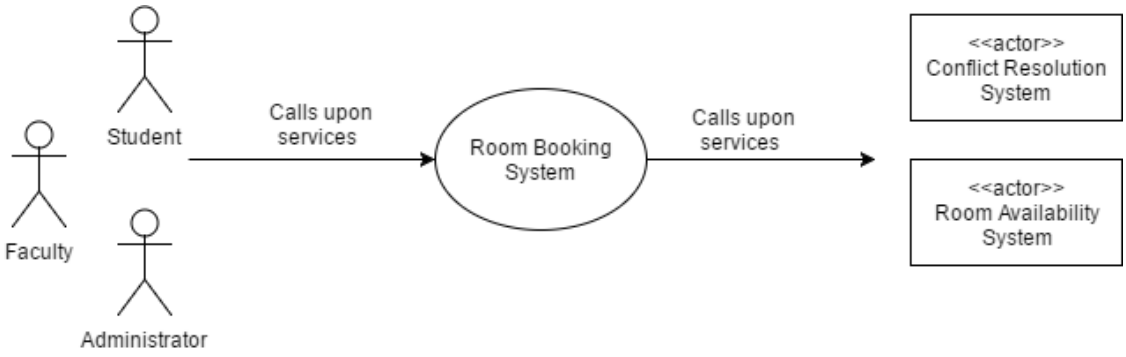


Figure vision - 1. Room Booking System context diagram

Summary of Benefits

Supporting Feature	Stakeholder Benefit
The user can interact with the system entirely via GUI	Ease of use, easy to update and maintain
Live updating of booking information	Accurate information for all users, reduces booking conflicts
Booking conflict resolution	Automated, reduces human interaction and human error possibilities
Web based interface	Nearly 100% uptime, fully cross platform

Assumptions and Dependencies

Cost and Pricing

Free and open source

Licensing and Installation

GPL - GNU General Public License

Summary of System Features

- Book rooms via web GUI
- Manage user privilege levels
- Live updating of booking information
- Booking conflict resolution
- Manage room - add new rooms, remove rooms that are no longer available for booking, update room information
- Take rooms out of bookings for maintenance

Other Requirements and Constraints

Ideally the web GUI will look and function like a calendar where users can see dates and times where they want to book rooms. This system will need 24/7 uptime and cannot be destroyed by outages at the deployment site. Outages may cause unaccounted for conflicts so a check sweep may be necessary on restarts of the system.