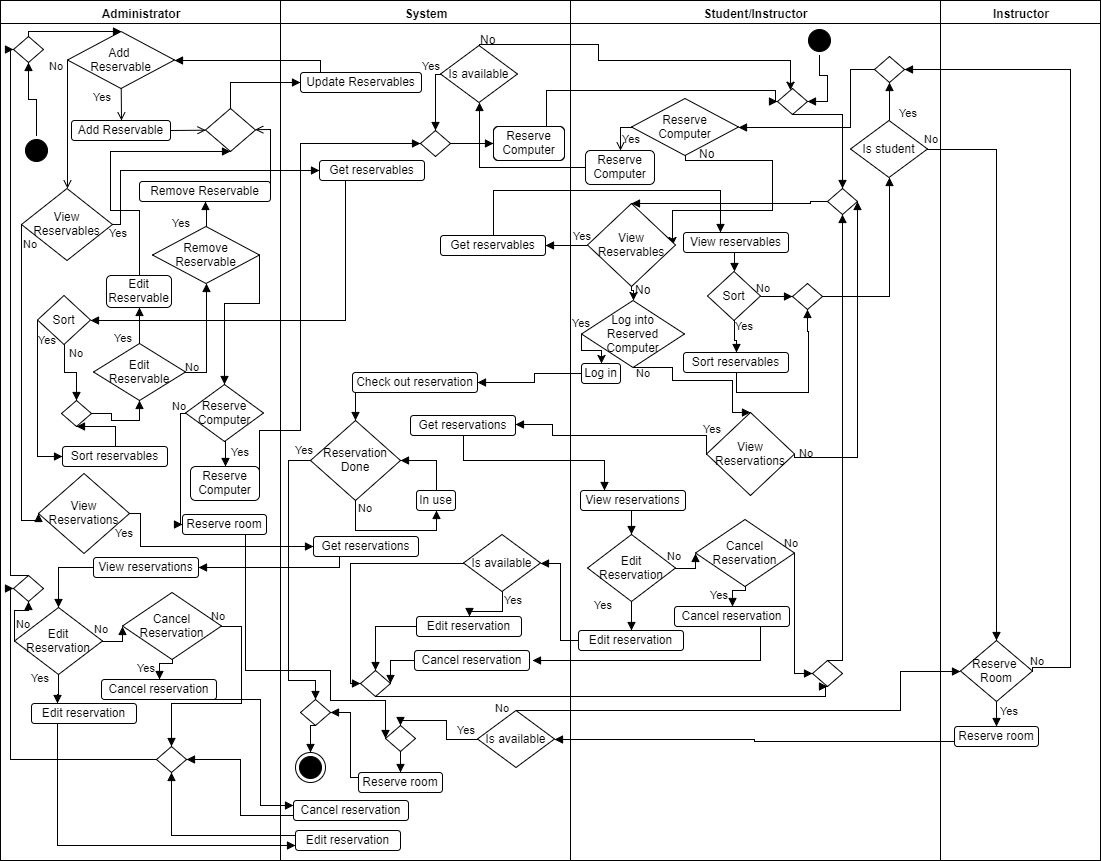
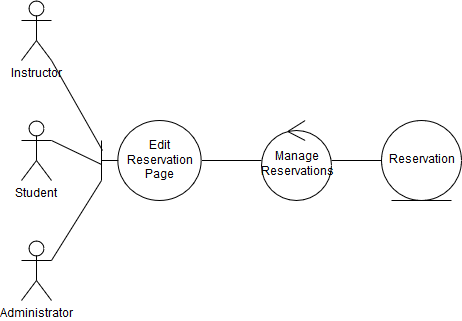
**Activity Diagram:**

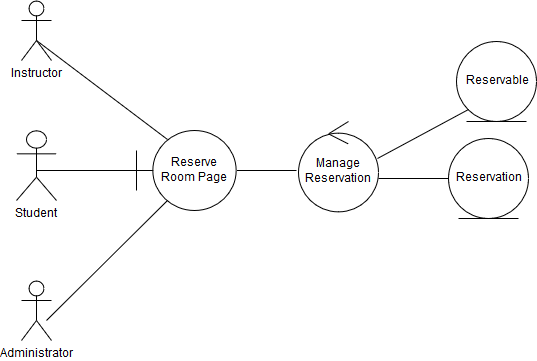


**Robustness Diagrams:**

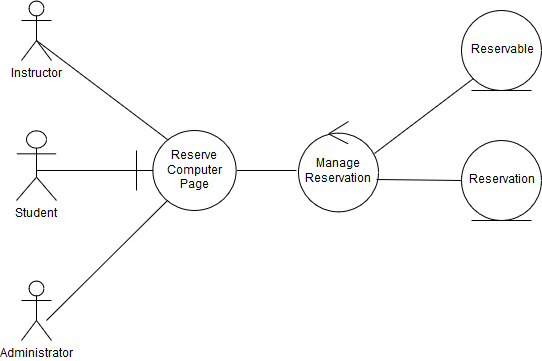
View Reservations:



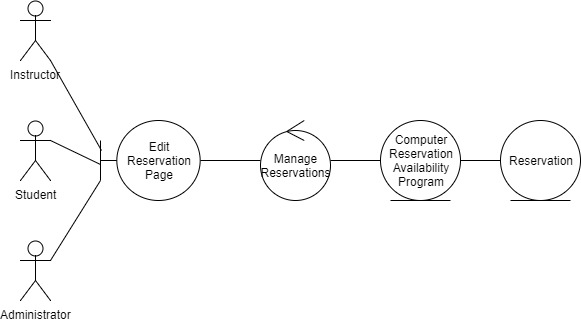
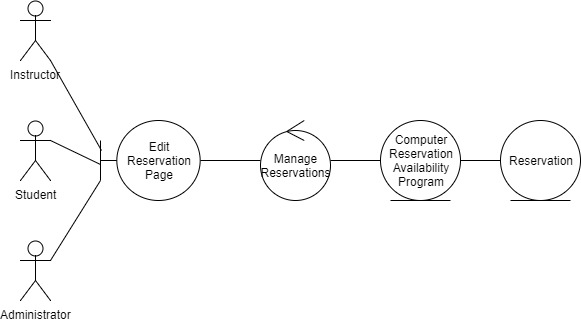
Reserve Room:



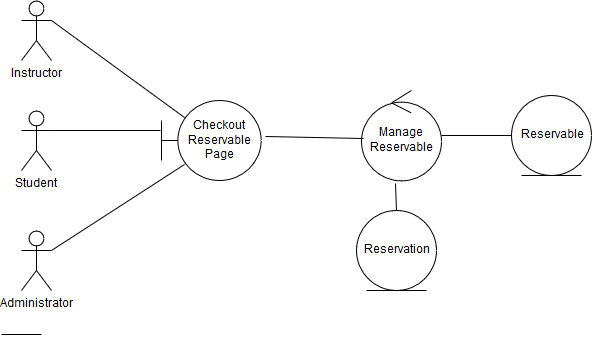
Reserve Computer:



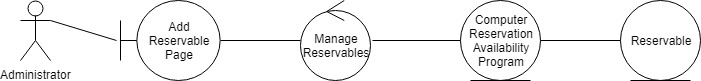
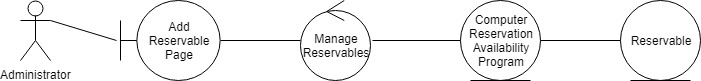
Edit Reservation:



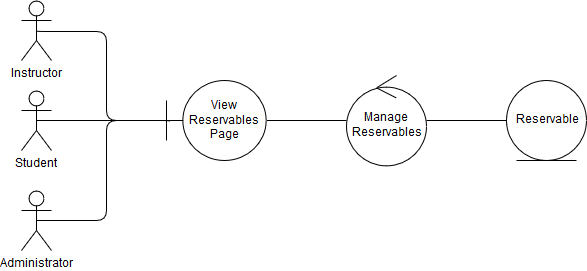
Checkout Reservable:



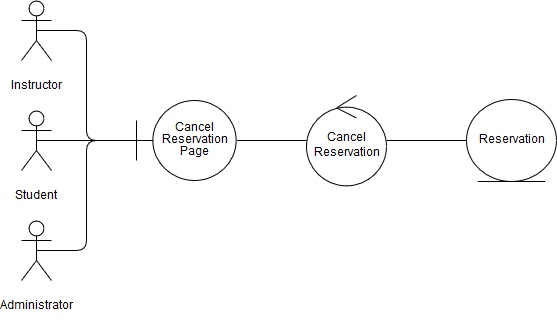
Add Reservable:



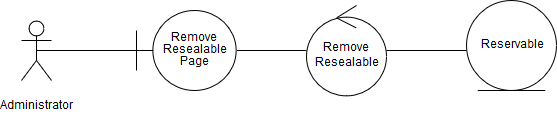
View Reservables:



Cancel Reservation:



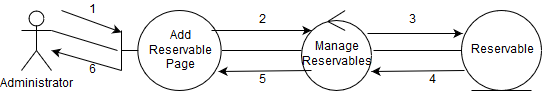
Remove Reservable:

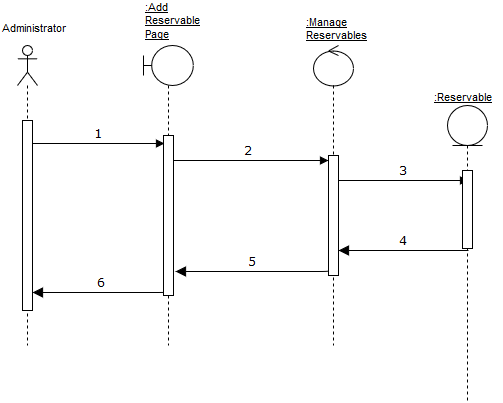


**Collaboration/Sequence Diagrams:**

Add Reservable:

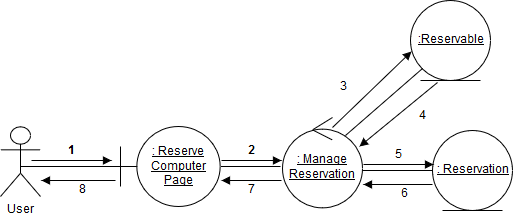
1. Provide reservable information
2. Pass reservable information
3. Create reservable
4. Return reservable
5. Reservable added
6. Reservable added

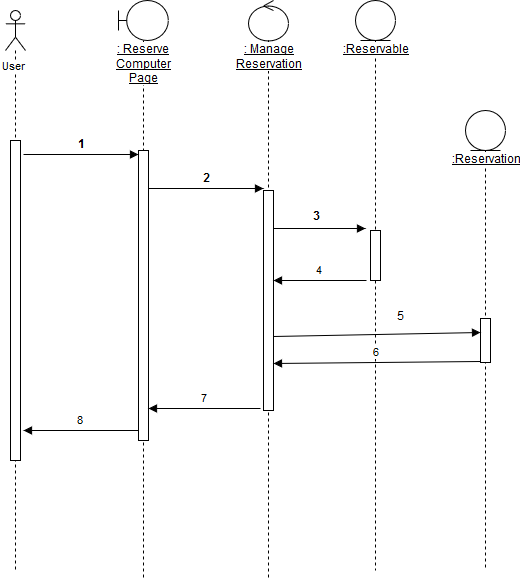




Reserve Computer:

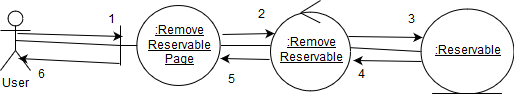
1. Request to reserve a computer
2. Pass reservation info
3. Create Reservable
4. Return Reservable
5. Create Reservation
6. Return Reservation
7. Reservation Added
8. Reservation Added

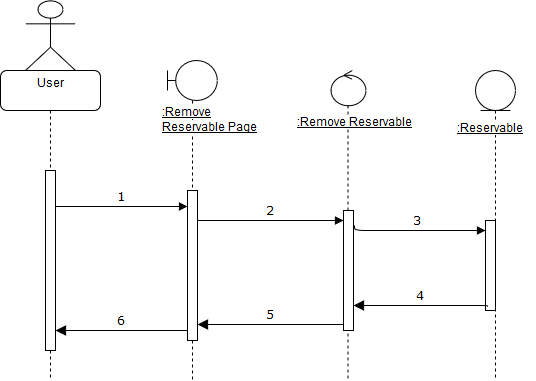




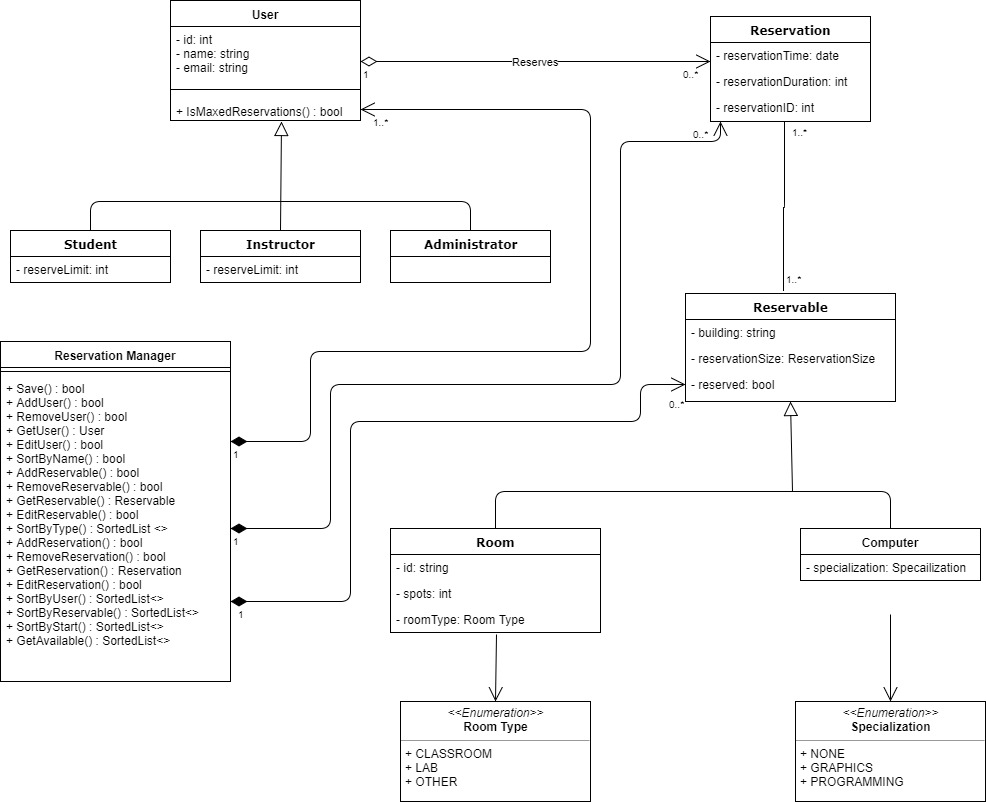
Remove Reservable:

1. Select the reservable to remove
2. Pass reservable information
3. Delete reservable
4. Reservable deleted
5. Successfully removed
6. Successfully removed





**Class Diagram:**



**Detailed Design:**

AddUser

If (adding a student)

Add new student to users

If (adding an instructor)

Add a new instructor to users

If (adding an administrator)

Add a new instructor to users

Return true

RemoveUser

If (users includes selected user)

Remove user from users

Return true

Else

Return false

GetUser

Search users for specified ID

If (ID is found)

Return user

Else

Return null

SortByName

Create sorted list of users with the user’s name as the key

Return the list

AddReservable

If (adding a computer)

Add new computer to reservables

If (adding a room)

Add a new room to reservables

Return true

RemoveReservable

If (reservables includes selected reservable)

If (reservable is a room)

Remove all computers in room from reservables

Remove reservable from reservables

Return true

Else

Return false

GetReservable

Search reservables for specified reservable

If (reservable is found)

Return reservable

Else

Return null

SortByType

Create sorted list of reservables with the reservable’s type as the key

Return the list

AddReservation

Add new reservation to reservations

Return true

RemoveReservation

If (reservations includes selected reservation)

Remove reservation from reservations

Return true

Else

Return false

GetReservation

Search reservations for specified reservable

If (reservation is found)

Return reservation

Else

Return null

SortByUser

Create sorted list of reservations with the user ID who made reservation as the key

Return the list

SortByReservable

Create sorted list of reservation with the reservation’s reservable as the key

Return the list

SortByStart

Create sorted list of reservation with the reservation’s start time as the key

Return the list

CheckAvailability

Foreach (reservation)

If (reservation is for specified reservable)

If (reservation time conflicts with new reservation time)

Return false

Return true

**Teamwork Dynamics:**

The weekly meetings were set at times everyone was available. This eliminated conflicting schedules allowing everyone to show up to each meeting. It also helped us have a more consistent work flow from week to week.

The GitHub repository allowed everyone to have access to the files at any time. This eliminated the issue our group had dividing up the work.

The workload was distributed evenly,

Levon Swenson, Junsu Jeong – Worked on detailed design with pseudocode.

Evan Gjerde – Worked on the class diagram.

Benjamin Mehn – Worked on organizing this word document refining all previous diagrams based on the feedbacks from last assignments.

Alex Bisbach – Worked on the activity diagram