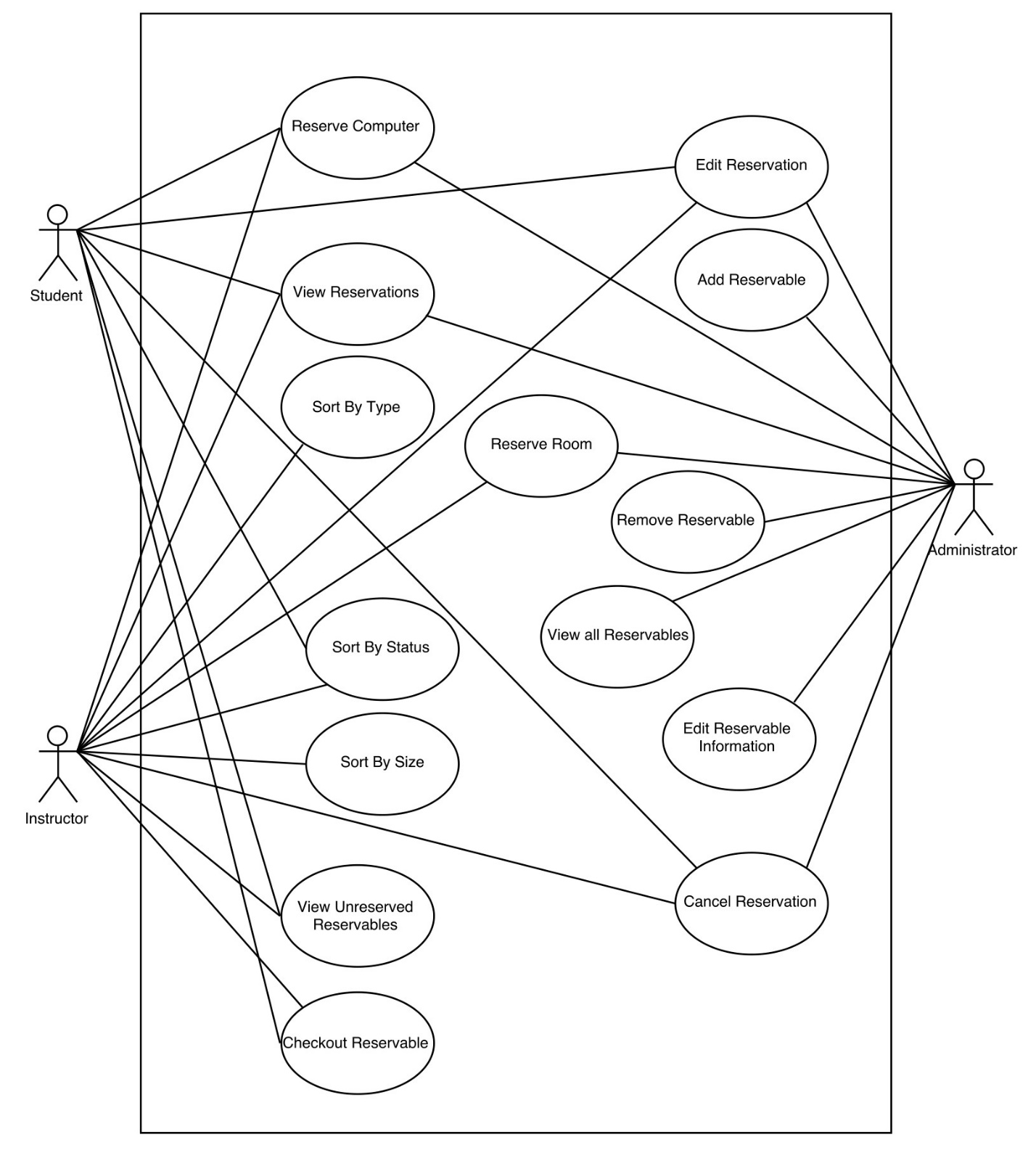
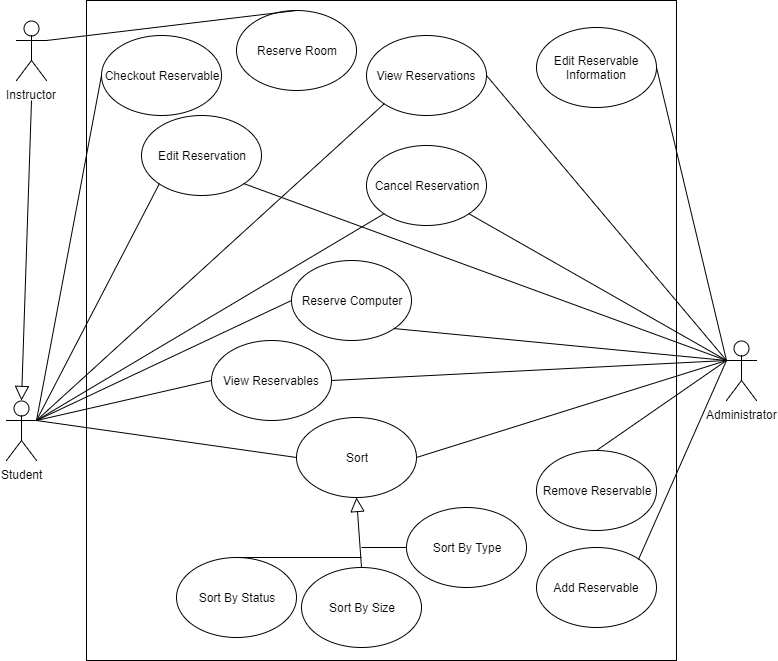
Requirements

Initial Use Case Diagram



Reservation System

Revised Use Case Diagram



Reservation System

|  |
| --- |
| Name: Sort by status |
| Actors: Student, Instructor, administrator |
| Trigger: User wants to sort the reservations by status |
| Preconditions: None |
| Post conditions: list is sorted by status |
| Success Scenario:   1. Users requests the reservation information to be sorted by status 2. System sorts the list based on the reserve status of the rooms 3. System displays the list to user |
| Alternatives flows: None |

|  |
| --- |
| Name: Sort by Size |
| Actors: Student, Instructor, administrator |
| Trigger: User wants to sort the reservations by Size |
| Preconditions: None |
| Post conditions: list is sorted by Size |
| Success Scenario:   1. Users requests the reservation information to be sorted by Size 2. System sorts the list based on the reserve status of the rooms 3. System displays the list to user |
| Alternatives flows: None |

|  |
| --- |
| Name: Sort by Type |
| Actors: Student, Instructor, administrator |
| Trigger: User wants to sort the reservations by Type |
| Preconditions: None |
| Post conditions: list is sorted by Type |
| Success Scenario:   1. Users requests the reservation information to be sorted by Type 2. System sorts the list based on the reserve status of the rooms 3. System displays the list to user |
| Alternatives flows: None |

|  |
| --- |
| Name: Cancel reservations |
| Actors: Student, Instructor, Administrator |
| Trigger: User cancels a reservation |
| Preconditions: User has a reservation |
| Post conditions: Computer’s Status is updated to reservable |
| Success Scenario:   1. User selects a reservation they made and cancels the reservation, 2. Changing the status to reservable during the student’s original reservation. |
| Alternative Flows: User Cannot cancel a reservation because they don’t have a reservation |

|  |
| --- |
| Name: View reservations |
| Actors: User, Instructor, Administrator |
| Trigger: User selects Checkout reservable |
| Preconditions: User selected Checkout reservable |
| Post conditions: Reservation list is displayed for Student |
| Success Scenario:   1. Reservation list is displayed to user |
| Alternative Flows: System has no rooms to view |

|  |
| --- |
| Reserve Room |
| Instructor |
| Instructor willing to use classrooms |
| Instructor requesting to reserve classrooms |
| System updates the status of reservation |
| 1. Instructor choose to request reserve for classrooms 2. System checks if there are available classrooms the selected timeslot 3. System shows the list of available classrooms with number of seats 4. Instructor choose the building and room that are available 5. System adds and update the reservation to the system |
| 2.1. There are available classrooms on selected timeslot  2.2. There are no available classrooms on selected timeslot |

|  |
| --- |
| Reserve Computer |
| Student, Instructor |
| Students or instructor willing to use the lab computers |
| Student or Instructor requesting to reserve the lab computers |
| System updates the status of reservation |
| 1. Instructors or students choose to request reserve for computer 2. System checks if there are available lab computers for specific timeslot 3. System shows the list of available lab computers 4. Instructors or students choose the lab and computer they want to reserve 5. System adds and update the reservation to the system |
| 2.1. There are available classrooms on selected timeslot  2.2. There are no available lab computers on selected timeslot |

|  |
| --- |
| Name: Edit Reservable Information |
| Actors: Administrator |
| Trigger: Administrator selects a reservable and chooses to edit it |
| Preconditions: There is a reservable selected that is preexisting in the system |
| Postconditions: The attributes and information of the reservable are changed |
| Success Scenario:   1. Administrator selects edit reservable 2. System uses <<include: View all Reservables>> 3. Administrator chooses reservable to edit 4. System “unlocks” attributes of that reservable 5. Administrator changes attributes and selects to save changes 6. System changes and saves attributes of the selected reservable 7. System displays list of reservables |
| Alternative Flows:  There are no reservables in the system  Administrator decides not to save changes and cancels edit |

|  |
| --- |
| Name: Remove Reservable |
| Actors: Administrator |
| Trigger: Administrator selects a reservable and chooses to remove it |
| Preconditions: There is a reservable selected that is preexisting in the system |
| Postconditions: The selected reservable is no longer in the system as a reservable |
| Success Scenario:   1. Administrator selects remove reservable 2. System uses <<include: View all Reservables>> 3. Administrator chooses reservable to remove 4. System removes reservable from list of reservables 5. System displays list of reservables with the deleted reservable no longer available |
| Alternative Flows: There are no reservables in the system |

|  |
| --- |
| Name: Add Reservable |
| Actors: Administrator |
| Trigger: Administrator chooses to add a reservable |
| Preconditions: Reservables screen is active |
| Postconditions: A new reservable is added to the list of reservables |
| Success Scenario:   1. Administrator chooses to add a reservable 2. System creates a new reservable 3. Administrator fills in attributes of the reservable and selects save 4. System saves reservable attributes and adds reservable to list of reservables 5. System displays list of reservables |
| Alternative Flows: Administrator decides not to add reservable to system and cancels add |

|  |
| --- |
| Name: View Reservables |
| Actors: User, Instructor, Administrator |
| Trigger: User decides to look at the list of reservables |
| Preconditions: System is running |
| Postconditions: Reservables list is displayed |
| Success Scenario:   1. System calls view all reservables 2. System retrieves list of reservables 3. System Displays name of reservables in a list to user |
| Alternative Flows: None |

|  |
| --- |
| Name: Checkout Reservable |
| Actors: Student, Instructor |
| Trigger: User logs in to check out the reservable |
| Preconditions: The user has an account |
| Post conditions: Reservation has been removed from the user’s list, if it was there |
| Success Scenario:  1. User selects reservable to check out  2. User checks out reservable  3. System removes reservable from the user’s list |
| Alternative Flows:  1.1 If there are no reservables in the user’s list, then the system returns to the previous state |

|  |
| --- |
| Name: Edit Reservation |
| Actors: Student, Instructor, Administrator |
| Trigger: User selects to edit a reservation |
| Preconditions: The user has an account |
| Post conditions: The user has the same number of reservations as before the use case was performed |
| Success Scenario:  1. The system performs <include: View Reservations>  2. The user selects a reservation to edit  3. The system displays the reservation  4. The user selects to change the reservation room  5. The system preforms <include: View Unreserved Reservables>  6. The user selects the new reservation  7. The system adds the old reservation back to the list, and removes the newly selected one |
| Alternative Flows:  4.1 The user only changes the time, in which case the reservation time is modified, and no more steps are performed.  6.1 The user cancels the edit, in which case the system returns to its previous state |

We divided the work as evenly as we could. First we met up and worked on the use case diagrams together in the lab. Then we split up the specification cards evenly giving everyone 2-4 cards.

Levon Swenson – Sort by status, Sort by Size, Sort by type

Evan Gjerde – Edit Reservable Information, Remove Reservable, Add Reservable, View all Reservables

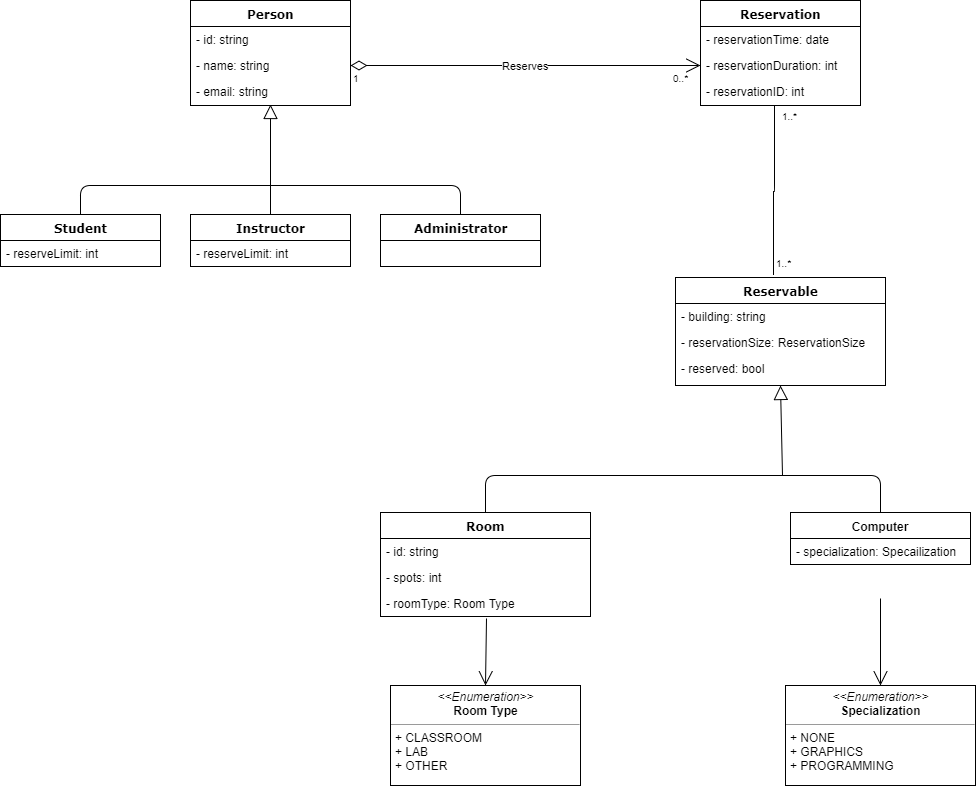
Junsu Jeong – Reserve Room, Reserve Computer

Benjamin Mehn – Cancel Reservations, View Reservations

Alex Bisbach – Checkout Reservable, Edit Reservation, View Unreserved Reservables, Rework according to feedback

1. Apply noun extraction to identify entity classes. You may also want to extract entity classes by examining scenarios of use cases. Please draw a class diagram, specify relationships among classes and for each class identify core attributes. (9 points)

Nouns: instructor, reservation, room, class, Computer, student, administrator, time, building, ID.



1. Create CRC cards for all your entity classes and use them to test and refine your class diagram (4 points)

|  |  |
| --- | --- |
| Student | |
| Manages his or her computer reservation.  View reservation status of computers and classrooms.  Create reservation for computer.  Cancel his or her reservation. | Reservation |

|  |  |
| --- | --- |
| Instructor | |
| Manages his or her reservation of computer or classroom.  View reservation status of computers and classrooms.  Create reservation for computer or classroom.  Cancel his or her reservation. | Reservation |

|  |  |
| --- | --- |
| Administrator | |
| Manage all the reservations.  View all the reservations.  Add rooms and computers to the list.  Remove rooms and computer from the list.  Manage the list of students and Instructors. | Reservation |

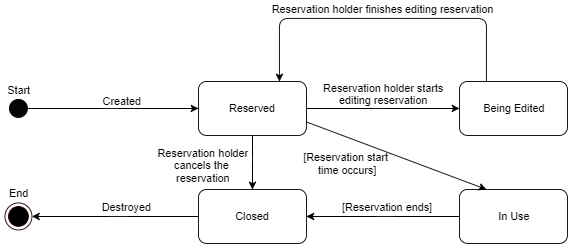
|  |  |
| --- | --- |
| Reservation | |
| Maintain reserved computer and classroom data.  Maintain reservation data  Maintain the available computer and classroom list by building and date/time.  Maintain the status of computer and classroom based on reservation time and duration. | Person  Reservation |

|  |  |
| --- | --- |
| Room | |
| Maintain the available times.  Maintain room data.  Check reservation availability.  Check reservation privileges.  Maintain a list of the reservations on the reservable | Reservation |

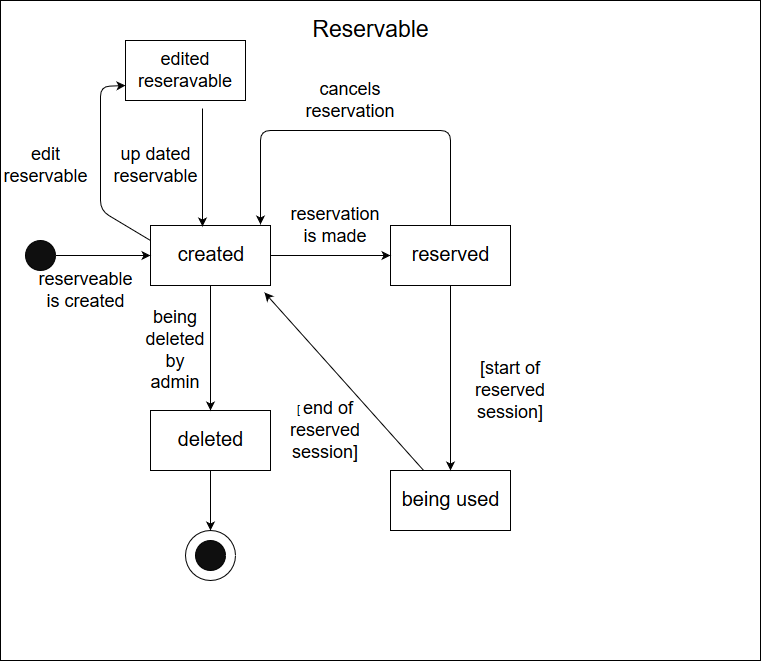
|  |  |
| --- | --- |
| Computer | |
| Maintain the available times.  Maintain computer data.  Check reservation availability.  Maintain a list of the reservations on the reservable | Reservation |

1. Choose at least two core entity classes. Determine operations performed by or to each class and present the information in the form of a State Chart. (4 points)

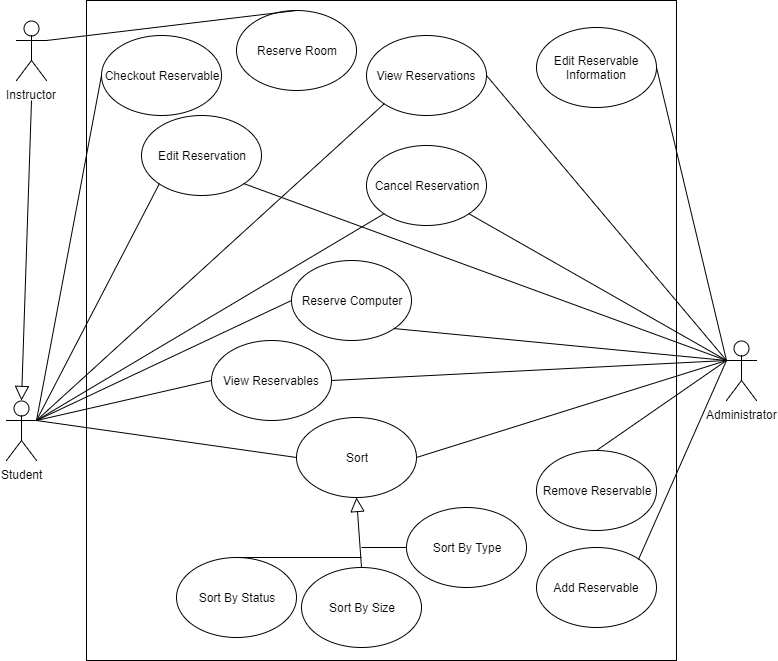
Reservation Class



Reservable Class



1. Complete your use case specifications if you have some parts left unfinished (triggers, preconditions, postconditions) in your last group assignment. If you have made changes to your use case diagram and specifications, please also submit your refined artifacts along with this submission. (2 points)



1. At this moment, you should have already decided the platform and the programming language for your system. Please briefly explain the reason of your choice. (1 point)

We decided to use C# as the language and we currently plan to make it a windows desktop application. Several of the people in the group have the most experience programing in this language. Also, most of the people in the group haven’t programmed an application for any environment other than for Windows.

The workload was distributed as follows; Evan helped with the class diagram, helped with CRC cards, and applied noun extraction. Junsu made CRC Cards and compiled the document with all the parts. Alex made the Class diagram and edited the Use case diagram according to the comments. Levon and Ben did the state charts and helped edit the use case diagram. Everyone helped complete the use case specifications.