Initial Use Case Diagram



Revised Use Case Diagram



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

|  |
| --- |
| Name: View reservations |
| Actors: Student |
| Trigger: Student selects Checkout resealable |
| Preconditions: Student selected Checkout resealable |
| 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 all Reservables |
| Actors: System |
| Trigger: Remove or Edit reservable is selected |
| 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 |

We divided the work as evenly as we could. First we we met up and worked on them together in the lab. Then we split up the specification cards evenly giving everyone 2-3 cards.