# Requirements Elicitation

The following sections contain a Use Case model of the SOS system. Section 4.1, Use Case Analysis, collects 30 uses cases describing interactions between the system and its users. Following that, Section 4.2, contains the Use Case Diagrams giving an UML description of the Use Cases in Section 4.1.

## Use Case Analysis

This section includes a description of the user types participating in the system in Section 4.1.1, SOS Actors, and each use case in full in the subsections of Section 4.1.2, Use Cases.

### SSL Actors

The actors participating in the system are:

* User – any individual using the website, including ones without a registered account.
* Member – any user with a registered account who belongs to an organization.
* Organizer – any member of an organization with leadership and/or administrative privileges on that organization.
* Admin – a privilege user with system-wide powers and access

### Use Cases

Each of the following subsections presents a Use Case describing a feature of the SOS system. These refer to the actors involved (see Section 4.1.1) and describe a step-by-step interaction between these actors and the system. They also include support information as well as usability, reliability, performance, supportability, and implementation constraints.

#### Create Event

**Use Case ID:** SOS1

**Use Case Level:** User Goal

**Details:**

* **Actor:** Organizer
* **Pre-conditions:**
  1. Organizer has successfully logged onto the system.
  2. Organizer is assigned to an Organization.
  3. Organizer has Event Creation privileges
* **Description:**
  1. Use case begins when Organizer clicks on **Create Event** on the administration page of their organization.
  2. The system shall prompt the Organizer with an **Event Creation** menu, which shall present them with a template for data entry.
  3. The Organizer shall enter the following data:
     + **Event Name**
     + **Event Date and Duration**
     + **Event Location**
     + **Event Description** (Optional)
     + **Event Type** (Defaults to Normal Event)
     + **Event Visibility** (Defaults to Visible)
  4. The Organizer shall complete the Event Creation by selecting the **publish** button.
  5. The system shall notify the Organizer that the event was published correctly.
  6. End case ends when the system receives the Event specifications, generates an **unique event id** and publishes the Event according to the given specifications.
* **Post-conditions:**
  1. An event has been published by the Organizer representing the Organization according to the specifications given.

**Alternative Courses of Action**

1. In step D.4, the Organizer has the option to **cancel** the Event Creation.
2. In step D.4, the Organizer has the option to **schedule** the Event Creation for a future date.
3. In step D.4, the Organizer has the option to **save without publishing** the Event Creation to complete at a later date.
4. In step D.5, if any of the required fields are blank, the system shall notify the Organizer and request an entry to the appropriate fields.

**Extensions:**

1. Create Special Event – Defines an Event with a **Type** other than Normal Event, which adds new fields to the Event Creation view and alters how the event is published.

**Exceptions:**

1. The event database is not active.
2. The event creation view is not active.

**Concurrent Uses:**

None

**Related Use Cases:**

None

**Decision Support**

**Frequency:** On average 3 Events are created per Organization weekly.

**Criticality:** High. The most basic and central activity of the whole system is Event Creation.

**Risk:** Medium. Implementation does not require any complex specialized knowledge.

**Constraints:**

* Usability
  1. No previous training or knowledge.
  2. Tutorial or Help frame should be provided.
  3. Organizer should take less than 10 minutes to create an event.
* Reliability
  1. Mean Time to Failure – 5% failure monthly is acceptable.
  2. Availability
     + Downtime for Login Back-up – 30 minutes in a 24-hour period.
     + Downtime for Maintenance – 1 hour in a 2 weeks period.
* Performance
  1. The form should be sent and saved within 10 seconds.
  2. The system should be able to handle 50 requests in 1 minute.
* Supportability
  1. The Event Creation should be supported by Chrome, Mozilla, and IE.
* Implementation
  1. The implementation shall use JS React for front-end, and Java-based software for back-end.

**Modification History**

**Owner:** Armando J. Ochoa

**Initiation date:** 09/01/2019

**Date last modified:** 09/15/2019

#### Grant Organizer Role

**Use Case ID:** SOS2

**Use Case Level:** User Goal

**Details:**

* **Actor:** Organizer
* **Pre-conditions:**
  1. Target Member belongs to the current organization.
  2. Target Member does not have Organizer status on the current organization.
  3. Organizer has power to give other people Organizer status.
* **Description:**
  1. Use case begins when the Organizer clicks on the **Add Organizer** tab on the organization management view.
  2. The system shall prompt the Organizer with an **Invitation Menu**, which shall present them with a template for data entry.
  3. The Organizer shall enter the following data:
     + **Member ID** (Either a name, or selectable from a drop-down menu with the list of organization members).
     + **Organizer Title** (Optional)
     + **Powers and Privileges** (From a list of pre-set privileges).
  4. The Organizer shall finish adding an organizer by selecting the **complete** button.
  5. The system shall notify the Organizer that the Member’s privilege and status has been changed correctly.
  6. Use case ends when the system changes the Member’s status in its database and the Member has been notified.
* **Post-conditions:**
  1. The status of the target Member has been changed, and he or she has received new privileges on the given organization.
  2. The list of Organizers in the Organization has been updated.
  3. The Member has been notified of the update.

**Alternative Courses of Action**

1. In step D.3, if the Organizer attempts to set a privilege that they themselves do not have, then the system shall notify them that they lack the required privileges (e.g., an Organizer without Event Creation privileges cannot invite another Organizer with Event Creation privileges).
2. In step D.4, the Organizer has the option to **cancel** the invitation.
3. In step D.5, if any of the required fields are blank, the system shall notify the Organizer and request an entry to the appropriate fields.

**Extensions:**

None

**Exceptions:**

1. Incorrect input in step D.3 (such as a non-existent Member ID) shall cause an exception and trigger a notification to the Organizer.

**Concurrent Uses:**

None

**Related Use Cases:**

None

**Decision Support**

**Frequency:** On average, 2 or 3 times per month per organization.

**Criticality:** High. This is basic element of the system and is required for good usability.

**Risk:** Medium. Implementation does not require any complex specialized knowledge.

**Constraints:**

* Usability
  1. No previous training or knowledge.
  2. Tutorial or Help frame should be provided.
  3. Organizer should take less than 10 minutes to complete the invitation.
* Reliability
  1. Mean Time to Failure – 1% failure yearly is acceptable.
  2. Availability – 30 minutes in a 24-hour period for backup and maintenance.
* Performance
  1. Privilege Checks should be done within 2 seconds.
  2. The system should handle 20 privilege checks in 1 minute.
* Supportability
  1. Should be supported by all browsers.
* Implementation
  1. Using Java-based software for back-end.

**Modification History**

**Owner:** Armando J. Ochoa

**Initiation date:** 09/01/2019

**Date last modified:** 09/15/2019

#### Earn Points by Attending an Event

**Use Case ID:** SOS3

**Use Case Level:** User Goal

**Details:**

* **Actor:** Member
* **Pre-conditions:**
  1. Member has successfully logged onto the system.
  2. Member belongs to an organization.
  3. Member is participating in the organization’s points ranking.
* **Description:**
  1. Use case begins when the Member is marked as attending an Event.
  2. The system shall check the Event log to see if the Member is already marked as having attended in this Event.
  3. The system shall note the Member’s participation on the Event log.
  4. The system shall note the Member’s participation on the Member’s page.
  5. The system shall award the Member a certain amount of points, as defined by the Event specifications.
  6. The system shall update the Organization’s ranking to reflect the new points.
  7. The case ends once the system notifies the Member that his or her point ranking has changed, by how much, and what his or her new ranking on the Organization is.
* **Relevant requirements:**

None

* **Post-conditions:**
  1. The Event log has been updated with the Member’s participation.
  2. The Member’s points towards the organization has been updated.
  3. The Organization ranking has been updated with the Member’s new points.
* **Alternative Courses of Action:**
  1. In steps D.2, if the Member’s participation is already in the Event log, then all of the following steps are ignored. The Member is notified that he or she has already participated in the Event.

**Extensions:**

None.

**Exceptions:**

1. The Event log, Organization, and Ranking are not accessible or active. In which case the Member shall be notified of the error and told his or her points will not be counted.

**Concurrent Uses:**

None

**Related Use Cases:**

SOS4 – Attending an Event   
 SOS9 – Member Ranking

**Decision Support**

**Frequency:** On average, 15-30 participants per Event, with an average of 3 Events per Organization created weekly.

**Criticality:** Medium. The point and ranking systems are an optional functionality that not everybody will use, and that is subordinate to other systems.

**Risk:** Medium. Implementation does not require any complex specialized knowledge.

**Constraints:**

* Usability
  1. No previous training or knowledge. The system should respond without user interaction after the attendance is completed.
* Reliability
  1. Meant Time to Failure: 5% failure monthly is acceptable.
* Performance
  1. The system should be able to handle 20 requests in 1 minute.
  2. The system should update the Event, Member, and Organization logs within 2 seconds.
* Supportability
  1. Point earning should be supported by Chrome, Mozilla, and IE.
* Implementation
  1. The implementation shall use JS React for front-end, and Java-based software for back-end, as well as SQL for database management.

**Modification History**

**Owner:** Armando J. Ochoa

**Initiation date:** 09/01/2019

**Date last modified:** 09/15/2019

#### Attending an Event

**Use Case Level:** User Goal

**Details:**

* **Actor:** Member
* **Pre-conditions:**
  1. Member has an account in our application.
  2. Member is successfully logged into the application.
  3. Member is part of a club and is attending an event hosted by said club.
* **Description:**

**Trigger:**

1. Use case begins when member clicks on the events tab.
2. The system shall provide the member with a sorted list of events that the user has signed up for.
3. The member will click on the event that they are currently attending.
4. The system shall provide the member with a description of the event as well as a button that says “I’m here!”
5. The user shall click on the “I’m here” button.
6. The system shall process the request for the click.
7. Use case ends when the system notifies the user that their attendance at the event was noted.

* **Relevant requirements:**

None

* **Post-conditions:**

1. The attendance request is saved in the system, along with arrival time.
2. The member is awarded a certain amount of points for attending the event.

**Alternative Courses of Action**:

1. In step D.10 the “I’m here” button will only appear if the user is at the location where the event is occuring.
2. In step D.8 the sorted list provided by to the user can be sorted by date the event will take place on or by club name.

**Exceptions:**

1. If the member tries to click the I’m here button 15 minutes before the event is ending they will not get credit for attending the event.

**Concurrent Use Cases:**

None.

**Related Use Cases:**

None.

**Decision Support**

**Frequency:** On average 100 attendance requests are made weekly by the club leader.

**Criticality:** High. Allows the member to notify their club that they are active in their organization.

**Risk:** High. Implementing this use case requires web-based technology and GPS tracking.

**Constraints:**

* Usability:
  1. No previous training time.
  2. On average the user should take 2 minutes to complete the notification request to the system.
* Reliability
  1. Mean time to failure – 5% failures for every month of operation is acceptable.
  2. Availability – Down time for Login Back-up 30 minutes in a 24 hour period.
* Performance
  1. Request should be sent and saved within 6 seconds.
  2. System should be able to handle 1000 request in 1 minute.
* Supportability
  1. The Event Creation should be supported by Chrome, Mozilla, and IE.
* Implementation
  1. The implementation shall use JS React for front-end, and Java-based software for back-end.

**Modification History**

**Owner:** Anthony Sanchez-Ayra

**Initiation date:** 09/04/2019

**Date last modified:** 09/15/2019

#### Ensure User Access

**Use Case ID:** SOS5

**Use Case Level:** Security.

**Details:**

* **Actor:** User
* **Pre-conditions:**
  1. User has privileged access to an Event, Organization, or Member Profile page.
  2. User is logged in.
* **Description:**
  1. Use case begins when the User clicks on an Event, Organization or Member Profile page.
  2. The system requests the User status and privileges.
  3. The system checks that status and privileges against the set requirements to see the Event, Organization, or Member Profile.
  4. The case ends when the privileged Event, Organization, or Member Profile view is presented to the User.
* **Relevant requirements:**

None

* **Post-conditions:**
  1. The User’s view has been changed to the appropriate Event, Organization, or Member Profile view. Privileged view might include editing, deleting, or seeing privileged information.
* **Alternative Courses of Action:**
  1. In step D.3, if the User status and privileges are not adequate to view the Event, Organization, or Member Profile page, then they are denied access or presented with a non-privileged view.

**Extensions:**

None.

**Exceptions:**

None.

**Concurrent Uses:**

None

**Related Use Cases:**

None

**Decision Support**

**Frequency:** On average, 20 attempts per day.

**Criticality:** High. The system should ensure correct access and privileges.

**Risk:** Medium. This is a standard security measure that does not require a lot of work to implement.

**Constraints:**

* Usability
  1. User must be aware of their privileges and what actions those privileges permit.
* Reliability
  1. Mean Time to Failure – 1% failure yearly is acceptable.
  2. Availability – 30 minutes in a 24-hour period for backup and maintenance.
* Performance
  1. Privilege Checks should be done within 2 seconds.
  2. The system should handle 20 privilege checks in 1 minute.
* Supportability
  1. Should be supported by all browsers.
* Implementation
  1. Using Java-based software for back-end.

**Modification History**

**Owner:** Armando J. Ochoa

**Initiation date:** 09/01/2019

**Date last modified:** 09/15/2019

#### Ensure User Profile Privacy

**Use Case ID:** SOS6

**Use Case Level:** Security.

**Details:**

* **Actor:** User
* **Pre-conditions:**
  1. The target User has profile information set to private or with restricted access.
* **Description:**
  1. Use case begins when the User attempts to view the private information belonging to the target User (e.g., a private feed, or a private membership, or ranking).
  2. The system shall check the target User’s privacy settings.
  3. The system shall check the User’s privileges.
  4. The system shall check the User against the target User’s whitelist.
  5. The case ends when the system rejects the User and present him or her with a standard page indicating that the page is private.
* **Relevant requirements:**

None

* **Post-conditions:**
  1. The system has presented the User with an adequate view of the profile.
  2. The system has logged the Misuser’s attempt to see the target Member’s data.
* **Alternative Courses of Action:**
  1. In step D.2, if the privacy settings are not **private**, then system shall provide access.
  2. In step D.3., if the User privileges allow it, then the system shall give access (i.e., the User is an **admin** or has similar privileges).
  3. In step D.4, if the User is in the target User’s whitelist, then the system shall provide them access.

**Extensions:**

None.

**Exceptions:**

None.

**Concurrent Uses:**

None

**Related Use Cases:**

SOS7 – Set Private Accounts

**Decision Support**

**Frequency:** On average, 20 attempts per day.

**Criticality:** Medium. The system should not allow Misusers to easily access non-privileged pages, but implementing private Member, Organization, and Event pages is a secondary objective to the main functionality of the system.

**Risk:** Medium. This is a standard security measure that does not require a lot of work to implement.

**Constraints:**

* Usability
  1. User must be aware of their privileges and what actions those privileges permit.
* Reliability
  1. Mean Time to Failure – 1% failure yearly is acceptable.
  2. Availability – 30 minutes in a 24-hour period for backup and maintenance.
* Performance
  1. Privilege Checks should be done within 2 seconds.
  2. The system should handle 20 privilege checks in 1 minute.
* Supportability
  1. Should be supported by all browsers.
* Implementation
  1. Using Java-based software for back-end.

**Modification History**

**Owner:** Armando J. Ochoa

**Initiation date:** 09/01/2019

**Date last modified:** 09/15/2019

#### Set Private Account

TODO: Ask Kian to redo this.

#### Sharing

**Use Case ID:** SSL8

**Use Case Level:** User Goal

**Details:**

* **Actor:** Member
* **Pre-conditions:**
  1. Member has successfully logged onto the system.
* **Description:**
  1. Use case begins when clicks on the **Share** link on an Event or Organization.
  2. The system shall prompt a menu with several sharing options, including:
     + Share with Other Member
     + Share with Facebook
     + Share with Twitter
     + Share with Email
     + Copy URL to Clipboard
  3. The user can decide how to share the Event or Organization by clicking on the corresponding choice.
  4. The system shares the Event or Organization.
  5. The case ends once the system notifies the Member that it has shared the Event or Organization according to his or her choice.
* **Relevant requirements:**

None

* **Post-conditions:**

None

* **Alternative Courses of Action:**
  1. In step D.3, the Member can click on **cancel** or outside of the menu to cancel the sharing.
  2. In step D.3, if the Member choose to Share with Other Member, then the system shall prompt another menu asking for the recipient User’s username.

**Extensions:**

None.

**Exceptions:**

None.

**Concurrent Uses:**

None

**Related Use Cases:**

None

**Decision Support**

**Frequency:** On average, events will be shared 20 to 30 times per week.

**Criticality:** Low. Not an important feature.

**Risk:** Low. Facebook, Twitter, and Email sharing are easy to implement using ready-made widgets.

**Constraints:**

* Usability
  1. No previous training or knowledge.
* Reliability
  1. Meant Time to Failure: 5% failure monthly is acceptable.
* Performance
  1. The system should be able to handle 20 requests in 1 minute.
  2. Sharing should happen instantly.
* Supportability
  1. Point earning should be supported by Chrome, Mozilla, and IE.
* Implementation
  1. The implementation shall use JS React for front-end, and Java-based software for back-end, as well as SQL for database management.

**Modification History**

**Owner:** Armando J. Ochoa

**Initiation date:** 09/01/2019

**Date last modified:** 09/01/2019

#### Member Ranking

**Use Case ID:** SOS9

**Use Case Level:** User Goal

**Details:**

* **Actor:** Member
* **Pre-conditions:**
  1. Member belongs to at least one Organization.
  2. Member enabled access to their current location via GPS.
  3. They have attended Events that gives them scores.
* **Description:**
  1. Use case begins whenever the Member is marked as attending an Event and earns points because of it.
  2. The system shall store the Member’s point total in a database, together with the Member’s information.
  3. The system shall rank the Member and all other members of his Organization based on their point score. This rank and the point total of all the members of an Organization shall be linked to in the Organization’s page.
  4. The case ends when the rankings are updated and redisplayed in the Organization’s page.

**Relevant requirements:**

None

**Post-conditions:**

None.

**Alternative Courses of Action:**

None

**Extensions:**

None.

**Exceptions:**

None.

**Concurrent Uses:**

None

**Related Use Cases:**

SOS3 – Earn Points by Attending and Event

**Decision Support**

**Frequency:** On Average, 30 members per Organization will be reporting attendance to Events

**Criticality:** Medium. The point and ranking systems are an optional functionality that not everybody will use, and that is subordinate to other systems.

**Risk:** Medium. Implementation requires specialized knowledge, but GPS and Geolocation Services are available in most web browsers (Desktop and Mobile).

**Constraints:**

* Usability
  1. User must be aware of their privileges and what actions those privileges permit.
* Reliability
  1. Mean Time to Failure – 1% failure yearly is acceptable.
  2. Availability – 30 minutes in a 24-hour period for backup and maintenance.
* Performance
  1. Privilege Checks should be done within 2 seconds.
  2. The system should handle 20 privilege checks in 1 minute.
* Supportability
  1. Should be supported by all browsers.
* Implementation
  1. Using Java-based software for back-end.

**Modification History**

**Owner:** Kian Maroofi

**Initiation date:** 09/10/2019

**Date last modified:** 09/15/2019

#### Access Events by Location

**Use Case ID:** SOS10

**Use Case Level:** User Goal

**Details:**

* **Actor:** User
* **Pre-conditions:**
  1. User is logged into the system.
* **Description:**
  1. Use case begins when the User goes to the Events page or the Home page on the website.
  2. The system shall ask for accessing to the current location of the User by GPS.
  3. The system shall verify that User gave access to their location.
  4. The system shall find events within a defined proximity range of the User’s location.
  5. The system shall update the Event map component to center on the User’s location.
  6. The case ends when the system modifies the Event feed to prioritize Events within range of the User’s location, and when the Event map component is updated to the User’s location.
* **Relevant requirements:**

None

* **Post-conditions:**
  1. The User’s location is tracked on the system, and several Events are marked as within range.
  2. The Map component is updated to center on the User’s location.
* **Alternative Courses of Action:**
  1. In step D.2, if the User has agreed to share location before, or if it has a permanent flag to share location in his or her profile, then it this step is ignored, and the system jumps directly to D.4
  2. In step D.3, if the User declines access, then the system shall ignore User location when presenting the Events.
  3. In step D.4, if location is not enabled, the system shall present all Events of the Organization.
  4. In step D.5, if location is not enabled, the system shall center on a system-wide default position.

**Extensions:**

None.

**Exceptions:**

None.

**Concurrent Uses:**

None

**Related Use Cases:**

None

**Decision Support**

**Frequency:** On average, users access the Home and Event pages 5 to 10 times daily.

**Criticality:** Medium, geolocation of events is an optional functionality that not everybody will use, and that is subordinate to other systems.

**Risk:** Medium. Medium. Implementation requires specialized knowledge, but GPS and Geolocation Services are available in most web browsers (Desktop and Mobile).

**Constraints:**

* Usability
  1. User must be aware of their privileges and what actions those privileges permit.
* Reliability
  1. Mean Time to Failure – 1% failure yearly is acceptable.
  2. Availability – 30 minutes in a 24-hour period for backup and maintenance.
* Performance
  1. Privilege Checks should be done within 2 seconds.
  2. The system should handle 20 privilege checks in 1 minute.
* Supportability
  1. Should be supported by all browsers.
* Implementation
  1. Using Java-based software for back-end.

**Modification History**

**Owner:** Kian Maroofi

**Initiation date:** 09/10/2019

**Date last modified:** 09/15/2019

#### Attendance Tracker

TODO: Redo this one.

#### Set up Two Factor Authentication (2FA)

**Use Case ID:** SOS12

**Use Case Level:** Security.

**Details:**

* **Actor:** User
* **Pre-conditions:**
  1. User have made an account on the web app already, and is not logged in.
* **Description:**
  1. Use case begins when the User clicks on **Enable 2 Factor Authentication** in their profile, under the security tab.
  2. The system shall generate a 2FA seed and save it to its database.
  3. The system shall ask the User to connect either Google Authenticator or such services using the generated seed.
  4. The system checks that authenticator service is successfully connected to their account on the website by asking for a generated 2FA code on the authenticator service.
  5. The case ends when the system confirms the link to the authentication service and notifies the User that 2FA has been enabled.
* **Relevant requirements:**

None

* **Post-conditions:**
  1. The User needs to provide 2FA generated codes every time they are trying to log in to their account on the website.
  2. The 2FA Seed for the User is stored in the system’s database.
  3. 2FA Authentication is marked as Enabled in the User’s profile.
* **Alternative Courses of Action:**

None

**Extensions:**

None.

**Exceptions:**

None.

**Concurrent Uses:**

None

**Related Use Cases:**

None

**Decision Support**

**Frequency:** On average, 5 attempts per day.

**Criticality:** High. The system should ensure correct access and privileges.

**Risk:** High. This is a standard security measure that does not require a lot of work to implement, including integration of authenticator applications such as Duo, Google Authenticator or SMS.

**Constraints:**

* Usability
  1. User must be aware of their privileges and what actions those privileges permit.
* Reliability
  1. Mean Time to Failure – 1% failure yearly is acceptable.
  2. Availability – 30 minutes in a 24-hour period for backup and maintenance.
* Performance
  1. Privilege Checks should be done within 2 seconds.
  2. The system should handle 20 privilege checks in 1 minute.
* Supportability
  1. Should be supported by all browsers.
* Implementation
  1. Using Java-based software for back-end.

**Modification History**

**Owner:** Kian Maroofi

**Initiation date:** 09/10/2019

**Date last modified:** 09/15/2019

#### Kick Privileges

**Use Case ID:** SOS13

**Use Case Level:** Privileges

**Details:**

* **Actor:** Organizer.
* **Pre-conditions:**
  1. Organizer has successfully logged onto the system.
  2. The application is open.
  3. There is at least one member part of the organization.
* **Description:**

1. Use case begins when Organizer clicks on the member management tab.
2. The system shall provide the Organizer with a list of members that are sorted.
3. The Organizer will click on the member that they want to kick out.
4. The Organizer will then click on the kick button in the member description.
5. The club adminstrator will provide a short description to the member why they are being kicked from their club.
6. The club adminstrator will send the request by selecting the send button.
7. The system shall notify the Organizer if the request was submitted correctly.
8. Use case ends when the system will removes the member from the club.

* **Relevant requirements:**

None

* **Post-conditions:**

1. The request to kick the member is saved by the system.
2. When the kicked member logs in they will receive a message notifying why they have been kicked from said club.

**Alternative Courses of Action**

1. In step D.6 (step 6 of Description section) the user has the option to cancel the kick request.
2. In step D.5 if the description is left blank the system will provide the user with a message to give a short reason why the member is being kicked.
3. In step D.2 the list of users can be sorted alphabetically or by ranking.

**Exceptions:**

1. There are no members in the club to kick.

**Related Use Cases:** None.

**Decision Support**

**Frequency:** On average 50 kick requests are made monthly by Organizer.

**Criticality:** High. Allows the club adminstrator to kick inactive members to make space for other people that will contribute to their organization.

**Risk:** Medium. Implementing this use case requires web-based technology.

**Constraints:**

* Usability:
  1. No previous training time.
  2. On average the user should take 2 minutes to complete the kick request to the system.
* Reliability
  1. Mean time to failure – 5% failures for every month of operation is acceptable.
  2. Availability – Down time for Login Back-up 30 minutes in a 24 hour period.
* Performance
  1. Request should be sent and saved within 6 seconds.
  2. System should be able to handle 100 request in 1 minute.
* Supportability
  1. The Event Creation should be supported by Chrome, Mozilla, and IE.
* Implementation
  1. The implementation shall use JS React for front-end, and Java-based software for back-end.

**Modification History**

**Owner:** Anthony Sanchez-Ayra

**Initiation date:** 09/03/2019

**Date last modified:** 09/15/2019

#### Promotion Privileges

**Use Case ID:** SOS14

**Use Case Level:** User Privileges

**Details:**

* **Actor:** Organizer.
* **Pre-conditions:**
  1. Organizer has successfully logged onto the system.
  2. The application is open.
  3. There is at least one member part of the organization.
* **Description:**

1. Use case begins when Organizer clicks on the member management tab.
2. The system shall provide the Organizer with a list of members that are sorted.
3. The Organizer will click on the member that they want to promote.
4. The Organizer will then click on the promote button in the member description.
5. The club adminstrator will provide a short congratulatory message for the user receiving the promotion.
6. The club adminstrator will send the request by selecting the send button.
7. The system shall notify the Organizer if the request was submitted correctly.
8. Use case ends when the system promotes the member of the club to the desired position.

* **Relevant requirements:**

None

* **Post-conditions:**

1. The request to promote the member is saved by the system.
2. When the promoted member logs in they will receive a message notifying them of their promotion.

**Alternative Courses of Action**

1. In step D.6 (step 6 of Description section) the user has the option to cancel the promotion request.
2. In step D.5 if the congratulatory message is left blank the system will provide the user with a message to give a short reason why the member is being promoted.
3. In step D.2 the list of users can be sorted alphabetically or by ranking.

**Exceptions:**

1. There are no members in the club to promoted.
2. The Organizer attempts to promote the leader of the club.
3. The Organizer attempts to promote a member to a position that has already been filled within the organization.

**Concurrent Use Cases:**

None.

**Related Use Cases:**

None.

**Decision Support**

**Frequency:** On average 10 promotion requests are made monthly by Organizer.

**Criticality:** High. Allows the club adminstrator to promote active members to make sure that the club can runs operations smoothly.

**Risk:** Medium. Implementing this use case requires web-based technology.

**Constraints:**

* Usability:
  1. No previous training time.
  2. On average the user should take 2 minutes to complete the promotion request to the system.
* Reliability
  1. Mean time to failure – 5% failures for every month of operation is acceptable.
  2. Availability – Down time for Login Back-up 30 minutes in a 24 hour period.
* Performance
  1. Request should be sent and saved within 6 seconds.
  2. System should be able to handle 100 request in 1 minute.
* Supportability
  1. The Event Creation should be supported by Chrome, Mozilla, and IE.
* Implementation
  1. The implementation shall use JS React for front-end, and Java-based software for back-end.

**Modification History**

**Owner:** Anthony Sanchez-Ayra

**Initiation date:** 09/03/2019

**Date last modified:** 09/15/2019

#### Notifications

**Use Case ID:** SOS15

**Use Case Level:** High-Level

**Details:**

* **Actor:** Member.
* **Pre-conditions:**
  1. Member has an account in the system.
  2. Member is part of at least one organization and is subscribed to events.
* **Description:**

1. Use case begins when member clicks on the clubs tab.
2. The system shall provide the member with a set of cards that represent the clubs that they are a part of.
3. The member will click on the club that they want to obtain notifications for.
4. The member will click on get event news button on the club description page.
5. The system shall notify the member that the request was submitted correctly.
6. Use case ends when the system allows the user to receive notifications for events of the club.

* **Relevant requirements:**

None

* **Post-conditions:**

1. The request to receive notifications from the club is saved in the system.

**Alternative Courses of Action**:

None.

**Exceptions:**

None.

**Concurrent Use Cases:**

None.

**Related Use Cases:**

None.

**Decision Support**

**Frequency:** On average 30 notification requests are made daily by the member.

**Criticality:** High. Allows the member to know when the organization that they are a part of is conducting events.

**Risk:** Medium. Implementing this use case requires web-based technology.

**Constraints:**

* Usability:
  1. No previous training time.
  2. On average the user should take 2 minutes to complete the notification request to the system.
* Reliability
  1. Mean time to failure – 5% failures for every month of operation is acceptable.
  2. Availability – Down time for Login Back-up 30 minutes in a 24 hour period.
* Performance
  1. Request should be sent and saved within 6 seconds.
  2. System should be able to handle 100 request in 1 minute.
* Supportability
  1. The Event Creation should be supported by Chrome, Mozilla, and IE.
* Implementation
  1. The implementation shall use JS React for front-end, and Java-based software for back-end.

**Modification History**

**Owner:** Anthony Sanchez-Ayra

**Initiation date:** 09/03/2019

**Date last modified:** 09/15/2019

#### Create Organization

**Use Case ID:** SOS16

**Use Case Level:** High-Level

**Details:**

* **Actor:** Organizer
* **Pre-conditions:**
  1. Organizer has an account in our application.
  2. Organizer is successfully logged into the application.
* **Description:**

1. Use case begins when Organizer clicks on the clubs tab in their current page (home page for example) and the homepage refreshes and provides the Organizer with the Club page.
2. The club page shall provide the Organizer with a set of cards that represent the clubs that they are a part of and a Create Club option.
3. The Organizer will click on the Create Club option.
4. The club page shall provide the Organizer with a form to fill out, asking for the following details:
   1. Club Name
   2. Club Description
   3. Requirements for joining
   4. Privacy of the club (whether its open to others or not).
5. The system shall notify the Organizer that the request was submitted correctly by showing a notification in the Club page.
6. Use case ends when the club page the displays the new club that the Organizer has created a new club.

* **Relevant requirements:**

None

* **Post-conditions:**

1. The request to create a club is stored in the system.
2. The club is shown to members depending on its privacy settings.

**Alternative Courses of Action**:

1. In step D.4 the user has the option to cancel the creation of their club.
2. In step D.5 if any of the fields are left blank the system will provide the user with a message to fill in all the fields.
3. In step D.5 the system shall ask the user to confirm if they would like to create a club.

**Exceptions:**

1. If the Organizer tries to make a club that already exists then they will get an error message.

**Concurrent Use Cases:**

None.

**Related Use Cases:**

None.

**Decision Support**

**Frequency:** On average 20 club creation requests are made monthly by the Organizer.

**Criticality:** High. Allows the Organizer to create a club which allows new communities to grow around campus.

**Risk:** Medium. Implementing this use case requires web-based technology.

**Constraints:**

* Usability:
  1. No previous training time.
  2. On average the user should take 2 minutes to complete the notification request to the system.
* Reliability
  1. Mean time to failure – 5% failures for every month of operation is acceptable.
  2. Availability – Down time for Login Back-up 30 minutes in a 24 hour period.
* Performance
  1. Request should be sent and saved within 6 seconds.
  2. System should be able to handle 200 request in 1 minute.
* Supportability
  1. The Event Creation should be supported by Chrome, Mozilla, and IE.
* Implementation
  1. The implementation shall use JS React for front-end, and Java-based software for back-end.

**Modification History**

**Owner:** Anthony Sanchez-Ayra

**Initiation date:** 09/04/2019

**Date last modified:** 09/15/2019

#### Cancel an Event

**Use Case ID:** SOS17

**Use Case Level:** User Goal

**Details:**

* **Actor:** Organizer
* **Pre-conditions:**
  1. Organizer has an account in our application.
  2. Organizer is successfully logged into the application.
  3. Organizer is part of a club.
* **Description:**

1. Use case begins when organizer clicks on the event that they want to cancel.
2. The system shall redirect the organizer to the Event Description view, which shall present them with a button labeled cancel event.
3. The organizer will click on the cancel event button.
4. The organizer will click yes on the validation message displayed by the system.
5. The system shall notify the organizer that the event was cancelled.
6. End case ends when the system removes the event from being viewed.

* **Relevant requirements:**

None

* **Post-conditions:**

1. The cancellation request is saved in the system.
2. The system notifies all users that subscribed to the event that it has been cancelled.

**Alternative Courses of Action**:

1. In step D.3 the system will prompt the organizer with a validation message to confirm that they actually want to cancel the event.

**Exceptions:**

1. The database is not active.
2. The Event Description view is not active.
3. The validation message is not active.

**Concurrent Use Cases:**

None.

**Related Use Cases:**

None.

**Decision Support**

**Frequency:** On average 5 cancellation requests are made weekly by the organizer.

**Criticality:** High. Allows the organizer to cancel an event whenever necessary.

**Risk:** High. Implementing this use case requires web-based technology.

**Constraints:**

* Usability:
  1. No previous training time.
  2. On average the user should take 2 minutes to complete the notification request to the system.
* Reliability
  1. Mean time to failure – 5% failures for every month of operation is acceptable.
  2. Availability – Down time for Login Back-up 30 minutes in a 24 hour period.
* Performance
  1. Request should be sent and saved within 6 seconds.
  2. System should be able to handle 100 request in 1 minute.
* Supportability
  1. The Event Creation should be supported by Chrome, Mozilla, and IE.
* Implementation
  1. The implementation shall use JS React for front-end, and Java-based software for back-end.

**Modification History**

**Owner:** Anthony Sanchez-Ayra

**Initiation date:** 09/04/2019

**Date last modified:** 09/15/2019

#### TBD

#### TBD

#### TBD

#### TBD

#### TBD

#### TBD

#### TBD

#### Filter Events

**Use Case ID:** SOS25

**Use Case Level:** User Goal

**Details:**

* **Actor:** User
* **Pre-conditions:**
  1. User is logged into the site.
* **Description:**

1. Use case begins when the user clicks on the “events” tab.
2. The user clicks on “find events”.
3. The system displays a list of tags (potlucks, volunteering, social events, etc).
4. The user selects one or more of their desired tags.
5. Use case ends when the system automatically updates the page with a list of events relevant to the selected tags.

* **Relevant requirements:**

None

* **Post-conditions:**

1. The relevant events are made viewable.

**Alternative Courses of Action**:

1. In step D.5, the user has the option to unselect and reselect tags.

**Exceptions:**

1. The find event button is not active.
2. The user does not select any tags.

**Concurrent Use Cases:**

None.

**Related Use Cases:**

None.

**Decision Support**

**Frequency:** On average 100 requests are made daily.

**Criticality:** High. Allows the user to find events they may be interested in.

**Risk:** Low. Implementing this use case doesn’t require specialized knowledge nor using it requires any sensitive information from the user.

**Constraints:**

* Usability:
  1. No previous training time, no explicit instructions required.
  2. Should take about 30 seconds for the average user to complete the use case.
* Reliability
  1. Mean time to failure – 5% failures for every month of operation is acceptable.
  2. Availability – Down time for Login Back-up 30 minutes in a 24 hour period.
* Performance
  1. The page should be updated in real time as the user clicks on each tag.
* Supportability
  1. The Event Creation should be supported by Chrome, Mozilla, and IE.
* Implementation
  1. The implementation shall use JS React for front-end, and Java-based software for back-end.

**Modification History**

**Owner:** Teriq Douglas

**Initiation date:** 09/06/2019

**Date last modified:** 09/16/2019

#### Invite User from Roster

**Use Case ID:** SOS26

**Use Case Level:** User Goal

**Details:**

* **Actor:** Organizer
* **Pre-conditions:**
  1. The Organizer is logged into site.
  2. The Organizer has the adequate privileges within the Organization.
* **Description:**

1. Use case begins when the organizer clicks “*My Organization*”.
2. Then organizer clicks “*View Roster*”.
3. The system shall shows a list of current members registered on the site.
4. The organizer clicks “*Invite Member*”.
5. The system shall ask for the organizer to input the member’s email.
6. The organizer clicks “*Submit*”.
7. The system shall ask the Organizer for comfirmation.
8. The organizer clicks “*Confirm*”.
9. The system shall send an invitation email to the member.
10. Use case ends when the system displays the message “*Invitation Sent*”.

* **Relevant requirements:**

None

* **Post-conditions:**

1. The relevant events are made viewable.

**Alternative Courses of Action**:

1. In step D.8, the organizer click “*Cancel*”, cancelling the request.

**Exceptions:**

1. Incorrect email.
2. The submit and/or remove button is not active.

**Concurrent Use Cases:**

None.

**Related Use Cases:**

SOS27 – Removing User from Roster

**Decision Support**

**Frequency:** About 5000 roster changes are made daily.

**Criticality:** High. Allows the organizer to have a stable view of their roster.

**Risk:** Low. Implementing this use case doesn’t require any complex knowledge.

**Constraints:**

* Usability:
  1. Might require light training.
  2. One help frame on the Help page provided.
  3. On average the user should take 1 minute to update their roster.
* Reliability
  1. Mean time to failure – 1% failures for every month of operation is acceptable.
  2. Availability – Down time for Login Back-up 30 minutes in a 24 hour period.
* Performance
  1. System should be able to handle 100 requests in 1 minute.
* Supportability
  1. The Event Creation should be supported by Chrome, Mozilla, and IE.
* Implementation
  1. The implementation shall use JS React for front-end, and Java-based software for back-end.

**Modification History**

**Owner:** Teriq Douglas

**Initiation date:** 09/06/2019

**Date last modified:** 09/16/2019

#### Remove User from Roster

**Use Case ID:** SOS27

**Use Case Level:** User Goal

**Details:**

* **Actor:** Organizer
* **Pre-conditions:**
  1. The Organizer is logged into site.
  2. The Organizer has the adequate privileges within the Organization.
* **Description:**

1. Use case begins when the organizer clicks “*My Organization*”.
2. Then Organizer clicks “*View Roster*”.
3. The system shall shows a list of current members registered on the site.
4. The Organizer clicks “*Remove Member*”.
5. The system shall ask the Organizer for a member’s name or email.
6. The Organizer clicks “*Submit*”.
7. The system shall ask the Organizer for comfirmation.
8. The Organizer clicks “*Confirm*”.
9. The system shall remove the member from the organization.
10. Use case ends when the system displays the message “*Invitation Sent*”.

* **Relevant requirements:**

None

* **Post-conditions:**

1. The relevant events are made viewable.

**Alternative Courses of Action**:

1. In step D.8, the organizer click “*Cancel*”, cancelling the request.

**Exceptions:**

1. Incorrect email.
2. The submit and/or remove button is not active.

**Concurrent Use Cases:**

None.

**Related Use Cases:**

SOS26 – Invite User from Roster

SOS13 – Kick Privileges

**Decision Support**

**Frequency:** About 5000 roster changes are made daily.

**Criticality:** High. Allows the organizer to have a stable view of their roster.

**Risk:** Low. Implementing this use case doesn’t require any complex knowledge.

**Constraints:**

* Usability:
  1. Might require light training.
  2. One help frame on the Help page provided.
  3. On average the user should take 1 minute to update their roster.
* Reliability
  1. Mean time to failure – 1% failures for every month of operation is acceptable.
  2. Availability – Down time for Login Back-up 30 minutes in a 24 hour period.
* Performance
  1. System should be able to handle 100 requests in 1 minute.
* Supportability
  1. The Event Creation should be supported by Chrome, Mozilla, and IE.
* Implementation
  1. The implementation shall use JS React for front-end, and Java-based software for back-end.

**Modification History**

**Owner:** Teriq Douglas

**Initiation date:** 09/06/2019

**Date last modified:** 09/16/2019

#### User RSVP

**Use Case ID:** SOS28

**Use Case Level:** User Goal

**Details:**

* **Actor:** User
* **Pre-conditions:**
  1. The Organizer is logged into site.
* **Description:**

1. Use case begins when the User finds clicks on “*RSVP*” on an Event.
2. The system shall display a description of the Event which includes the date, time, location, and a list of rules.
3. The User must click on “*Confirm*”to confirm the RSVP.
4. Use case ends when the system shows a success message.

* **Relevant requirements:**

None

* **Post-conditions:**

1. The system adds the user to the guest list.
2. The system adds the event to the user’s list of attending events.

**Alternative Courses of Action**:

1. In step D.3, the User can cancel the RSVP by clicking on “*Cancel*”.

**Exceptions:**

1. Max number of guests reached.

**Concurrent Use Cases:**

None.

**Related Use Cases:**

None.

**Decision Support**

**Frequency:** On average 500 RSVPs are made daily.

**Criticality:** Medium. Allows the user to formally attend events created on campus as long as they agree with the terms set by the hosts.

**Risk:** Low. Implementing this use case doesn’t require any complex knowledge.

**Constraints:**

* Usability:
  1. Requires no training.
  2. On average the user should take 20 seconds to perform an RSVP.
* Reliability
  1. Mean time to failure – 1% failures for every month of operation is acceptable.
  2. Availability – Down time for Login Back-up 30 minutes in a 24 hour period.
* Performance
  1. RSVP requests should be processed within 5 seconds.
  2. The system shall be consistent when handling RSVP requests.
* Supportability
  1. The Event Creation should be supported by Chrome, Mozilla, and IE.
* Implementation
  1. The implementation shall use JS React for front-end, and Java-based software for back-end.

**Modification History**

**Owner:** Teriq Douglas

**Initiation date:** 09/06/2019

**Date last modified:** 09/16/2019

#### Unauthorized Organization Management

**Use Case ID:** SOS29

**Use Case Level:** User Goal

**Details:**

* **Actor:** User
* **Pre-conditions:**
  1. The User is logged into site.
  2. The User does not have privileges (or Organizer status) on the target Organization.
* **Description:**

1. Use case begins when the User access target Organization’s page.
2. The system shall check for the User’s privileges on that Organization.
3. Use case ends when the system displays the Organization profile, which includes a description and contact information and excludes “*View Roster*” as well as other privileged views of the Organization.

* **Relevant requirements:**

None

* **Post-conditions:**

1. The view of the website has changed to the target Organization’s page.

**Alternative Courses of Action**:

None

**Exceptions:**

None

**Concurrent Use Cases:**

None.

**Related Use Cases:**

None.

**Decision Support**

**Frequency:** The act of viewing an organization’s profile will occur on average 1000 times daily.

**Criticality:** High. Prevents unauthorized changes in an organization’s roster.

**Risk:** Medium. Implementing this use case doesn’t requires some specialized knowledge about privilege control.

**Constraints:**

* Usability:
  1. Requires no training.
  2. On average the user should take less than 5 seconds to locate and click on the Organization page. It should also not take longer than 1 minutes to realize that the view is different when not logged as an Organizer.
* Reliability
  1. Mean time to failure – 1% failures for every month of operation is acceptable.
  2. Availability – Down time for Login Back-up 30 minutes in a 24 hour period.
* Performance
  1. Should be able to produce results within 3 seconds.
* Supportability
  1. The Event Creation should be supported by Chrome, Mozilla, and IE.
* Implementation
  1. The implementation shall use JS React for front-end, and Java-based software for back-end.

**Modification History**

**Owner:** Teriq Douglas

**Initiation date:** 09/06/2019

**Date last modified:** 09/16/2019

#### Unauthorized Event Creation

**Use Case ID:** SOS30

**Use Case Level:** User Goal

**Details:**

* **Actor:** User
* **Pre-conditions:**
  1. The User is logged into the site.
  2. The User does not have privileges (or Organizer status) on the target organization.
* **Description:**

1. Use case begins when the user clicks “*My Organizations*” assuming the user belongs to an organization.
2. The system shall display a list of organizations the user belongs to.
3. The user selects their desired organization.
4. The system shall check the privileges of the User relating to the chosen administration.
5. Use case ends when system displays the profile page omitting the “*Schedule*” button and other managirial views.

* **Relevant requirements:**

None

* **Post-conditions:**

1. The view of the website has changed to the target organization’s page.

**Alternative Courses of Action**:

1. In step D.2, if they user does not belong to any organization, when they click on “my organization” the system will display a message saying that they do not belong to one.

**Exceptions:**

None

**Concurrent Use Cases:**

None.

**Related Use Cases:**

None.

**Decision Support**

**Frequency:** On average, up to 3000 requests daily.

**Criticality:** High. Prevents unauthorized event creation.

**Risk:** Medium. Implementing this use case doesn’t requires some specialized knowledge about privilege control.

**Constraints:**

* Usability:
  1. Requires no training.
  2. On average the user should take less than 5 seconds to locate and click on the Organization. It should also not take longer than 1 minutes to realize that the view is different when not logged as an Organizer.
* Reliability
  1. Mean time to failure – 1% failures for every month of operation is acceptable.
  2. Availability – Down time for Login Back-up 30 minutes in a 24 hour period.
* Performance
  1. Should be able to produce results within 3 seconds.
* Supportability
  1. The Event Creation should be supported by Chrome, Mozilla, and IE.
* Implementation
  1. The implementation shall use JS React for front-end, and Java-based software for back-end.

**Modification History**

**Owner:** Teriq Douglas

**Initiation date:** 09/06/2019

**Date last modified:** 09/16/2019

## Use Case Diagrams

This section contains the Use Case Diagrams giving an UML description of the Use Cases in the previous section. Section 4.2.1, Full Use Case Diagram, contains several UML Use Case diagrams describing the planned system, which includes the 30 Use Cases collected in Section 4.1. Following that, Section 4.2.2., Implemented Use Case Diagram contains a UML Use Case diagram describing the Use Cases that are currently implemented in the current version of the SOS system. This amounts to 10 Use Cases from Section 4.1.

### Full Use Case Diagram

PLACEHOLDER

### Implemented Use Case Diagram

PLACEHOLDER