Student Organization System
Software Requirements Document

CEN4010 – Software Engineering
Prepared for Peter Clarke
By Team 5

Teriq Douglas
Yovanni Jones
M. Kian Maroofi
Armando J. Ochoa
Anthony Sanches-Ayra

October 1, 2019

Abstract

Student Organization System (SOS) is a web-based system meant to provide leaders and administrators of organizations a way to manage members and events. Simultaneously, it allows users to monitor the events and organizations they belong to. The SOS is developed using the Unified Software Development Process (USDP), the two first sections of which are contained in this document. The specifications of the system are captured in the form of Use Cases, forming the Use Case model of the USDP. Finally, these Use Cases are used to develop the Analysis model. This is done in the form of Unified Modeling Language diagrams, which describes both static, in the form of Class and Object diagrams, and dynamic, in the form of Sequence Diagrams, views of the system.

Table of Contents

1 Introduction

The following chapter introduces the Software Requirements Document (SRD) with the main goal of explaining the ideas and concepts behind the Student Organization System (SOS) project.

The purpose of this System Requirements Document (SRD) is to define the requirements of the SOS system, and to act as a basis for a more detailed Design Document (DD). These requirements, compiled in the form of Use Cases, describe the interactions between the potential users and the system. Moreover, they describe the system holistically, with requirements that apply both to the client- and to server-side system operations.

The purpose of the SOS is defined in Section 1.1. Following that, the scope of the system is defined in Section 1.2. Section 1.3 contains a list of relevant terms, acronyms, definitions and abbreviations used throughout the system. Finally, Section 1.4 contains a brief outline of this document. Following chapters including a Use Case model of the planned system (Section 3), an Analysis model (Section 4), and a detailed section on project management (Section 2).

1.1 Purpose of the System

The Student Organization System (SOS) is a web-based system meant to provide leaders and administrators of organizations a fast, interactive, and accessible way to manage members and events from a single, centralized place. Simultaneously, the SOS system also allow users to monitor and keep up-to-date information about the events and requirements of the organizations they belong to. Finally, the system also allows organizers to advertising their organizations and recruit new members from the general userbase. In essence, the Student Organization System is meant to aid the interaction between members and organizations.

Although the system is meant primarily for academic settings, with Universities being the main target, organization creation and management is open and could be used in other environments, both academic (High Schools, etc.) and non-academic (Company Campuses, Community Centers, etc.)

1.2 Scope of the System

The managerial side of the Student Organization System (SOS) allows organizers to administer their organizations by in four core ways:

i. It provides a single point of access for users, members and non-members alike;
ii. It organizes all the members of the club under a single network;
iii. It provides tools to manage organization leaders and their privileges; and
iv. It provides means to create and advertise organization-related events;

The system also allows users to interact with the following ways:

i. By collating all the organizations that they belong (or might interested in) to in a single place.
ii. By presenting them with events in their surrounding hosted by their organizations or any other public events.
iii. By connecting them with new organizations and other members.

The event system specially is at the core of both the managerial and the user side of the system. Events postings are created by organizers to promote their organizations and are attended by users. Events are geo-tagged and presented primarily by their location and time. Moreover, the system also provides attendance tracking and RSVP functionalities, which are integrated into a point-ranking system that organizers might choose to enable for their organizations in order to foster member participation.

The following functionalities, although related to organization management and user communities, are outside of the scope of the SOS and are not covered by this document:

i. Integration with social media sites (e.g., Facebook)
ii. Social media features such as comments and postings.
iii. User-defined (as contrasted to organization-defined) events.
iv. Detailed organization leader management features (e.g., payments, duties, etc.)
v. Detailed organization tasks and projects management systems.
vi. Organization-relation features (e.g., community chapters)

A future version of the SOS might include some of the aforementioned features.

1.3 Development Methodology

The development of the Student Organization System (SOS) follows the Unified Software Development Process (USDP; Jacobson, Booch, & Rumbaugh, 1999). The USDP can be seen as defined by a set of interconnected models: (a) use case model, (b) analysis model, (c) design model, (d) deployment model, (e) implementation model, and (f) test model.

This document contains the first two models, the use case model in Section 4, Requirements Elicitation; and the analysis model in Section 5, Requirements Analsysis. The first of these two, the use case model, captures the system requirements in the form of use cases, templated documents containing a detailed sequence of events describing all the possible interactions between users and the system for a particular piece of functionality. The second of the two, the analysis model, contains a more structured and formal description of these use cases using Unified Modeling Language (UML) class, sequence, and object diagrams. Between them, the two models contained in this document provide a consice description of the system and its functionalities.

1.4 Definitions, Acronyms, and Abbreviations

Table 1: Definitions, Acronyms, and Abbreviation, contains a series of terms and acronyms used through this document. A further glossary can also be found in Section 6 of this document.

Term	***Meaning***
API	Application Programming Interface
DB	Data Base (Data Storage)
DD	Design Document
FIU	Florida International University
FSD	Final Systems Document
N/A	Not Applicable

SOS	Student Organization System
SRD	Software Requirements Document
UML	Unified Modeling Language
USDP	Unified Software Design Process
V&V	Validation & Verification

Table 1: Definitions, Acronyms, and Abbreviation

1.5 Overview of the Document

This document is structured into chapters, each of which tackles a different aspect of the requirements for the SOS project. Chapter 2 discuss the possibility of current systematic solutions and compares them to the SOS system. Following that, Chapter 3 includes project planning and the schedule, and it introduces the team, explains the hardware and software requirements, project schedule including tasks and milestones. Chapter 4 introduces the requirements elicitation which consists of 30 total use cases (including the security use cases) as well as the use case diagram. Chapter 5 contains the system requirements analysis. This includes a description for ten different assumed scenarios, object diagrams, a class diagram, and sequence diagrams. Chapter 6 contains the glossary, which defines and describes domain specific terms. Finally, after that, chapters 7 and 8 contain the approval page with team mate signatures and the references respectively.

The appendices are contained in chapter 9. Three appendices are included. Appendix A contains a detailed project schedule in the form of a Gantt chart. Appendix B contains the user interface design as a collection of captures from the system prototype. And finally, Appendix C collects the diary of meetings and tasks.

2 Current Systems

Although the Student Organization System (SOS) shares some functionalities with social media sites such as Facebook (which can create group pages to which individual profiles can subscribe to), it is most similar to existing student organization systems that are hosted and managed by Universities and Colleges.

An example of this system is Panther Connect (Campus Lab, 2019) which is hosted by Florida International University (FIU). Panther Connect is a website where FIU students can search for clubs and organizations to join. It also allows them to keep track off and attend events happening on the FIU campus and sign up for volunteering activities. Event organizers can keep track of their member's involvement and stay in contact with them, as well as send invitations to new users. The platform also allows to create events that members can RSVP to.

Because of the relationship the system has with FIU, it provides another functionality which the SOS system does not provide: forms for campus and institutional requests. These are accessible through the "Forms" option in the homepage. However, privacy is a problem with this system. Whenever an organization creates a form, it is automatically added to the list where any user can view it. They do not need to have an account or a membership with the organization in order to access it.

Membership management has a couple of functionality issues. This is related to the fact that the platform is event-centered rather management-centered. For management, the platform provides the option to end a member's membership. However, it doesn't provide the functionality of selecting multiple members at once (except for an "End All Memberships" button, but it is not needed most of the time). If a user needs to end a decent number of memberships, they must select each member one by one. It also allows users to invite others through email, but it can only invite 500 people at a time. There is also a way for users to send mass emails to their rosters. Unfortunately, its mail server can be unreliable at times and may sometimes take up to 3 hours to send an email.

Most other communities (non-academic) have organization management systems but these are either offline, with a primary focus on management, or they are manual (e.g., community centers sharing events through newsletters or posters).

3 Project Plan

The following sections describes the how the project is structured, which includes managerial information, constraints on the system, and a schedule of activities. Section 3.1 describes the organization of the project, which includes information about the members, the established communication mechanisms, the schedule of meetings, and the assigned roles. Section 3.2 contains the hardware and software constrains of the system. Finally, Section 3.3 contains the schedule table for the project.

3.1 Roles

This section contains the roles used in the project and who they are assigned too. Each member has several roles assigned to them at the same time. Two contexts for roles are differentiated, *management roles*, some of which refer to roles which are only active during the weekly or supplementary meetings; and *project roles*, which refer to roles in action during the project as a whole.

3.1.1 Management Roles

Each member has a single, or none, management role assigned:

Member Name	Roles
Armando J. Ochoa	Primary Faciliatory
Anthony Sanchez-Ayra	Team Leader
M. Kian Maroofi	Time Keeper
Yovani Jones	
Teriq Douglas	Minute Take

Table 2: The management roles assigned to the team members.

3.1.2 Development Roles

Each member has one or more project roles assigned:

Member Name	Roles
Armando J. Ochoa	Front-End Developer, Document Editor
Anthony Sanchez-Ayra	Back-End Developer, Database implementor
M. Kian Maroofi	Front-End Developer
Yovani Jones	Back-End Developer, Tester
Teriq Douglas	Back-End Developer

Table 3: The development roles assigned to the team members.

3.2 Hardware and Software Requirements

The hardware and software materials needed to complete the project are captured in the following subsections.

3.2.1 Hardware Requirements

The testing environment is a network-enabled computer system with the following hardware requirements:

- Processor: Intel (R) Core (TM) i7-7700 CPU @ 3.60GHz

- Installed Memory (RAM): 16GB DDR4 SDRAM
- Storage: 512GB
- Network Adapter: Inter (R) Ethernet Connection (2) I219-LM

Each member has its own individual station. The details of these stations are not reported in this document.

3.2.2 Software Requirements

The testing environment has the following software applications:

- MySQL 8.0, which is used for a back-end data store server.
- Java JDK 1.8.0_221-b11, with the following external libraries:
  - *netty-socketio*, a java implementation of *socket.io* used for front-end/back-end communication.
- Node.JS version 10.16.3 LTS, with the following external libraries:
  - *React*, which is used to create the front-end.
  - *Redux*, which is used for state management of the front-end.
  - *Router*, which is used to handle front-end navigation.

3.3 Project Schedule Table

The project is divided into several tasks, which are collected in Section 3.3.1. These tasks build towards the following deliverables:

- Deliverable 1, Software Requirements Document (SRD; this document), which has the following milestones:
  - M1, Section 4, Requirements Elicitation, and Section 5, Requirements Analysis, are completed.
  - M2, Software Requirements Document is completed.
- Deliverable 2, Design Document (DD), which has the following milestones:
  - M3, Section 2, Proposed Software Architecture, and Section 3, Detailed Design, are completed
  - M4, Design Document is completed.
- Deliverable 3, Final Systems Document (FSD), which has the following milestones:
  - M5, System Implementation is Completed.
  - M6, Section 7, Testing Process, and Appendix F, Document code for Test Driver are completed.
  - M7, Final System Document is completed.

3.3.1 Task Schedule

Task	Group	Description	Duration (Days)	Dependencies
T1		Set up Communication Methods: Weekly Meeting, WhatsApp Group, and GitHub.	1	
T2	Requirements Elicitation	Identify Use Cases for the Complete System	15	
T3	Requirements Elicitation	Decide on 10 Implementation Use Cases	1	T2
T4	Requirements Elicitation	Create Use Case Diagrams	6	T2
T5	Requirements Analysis	Create Sequence Diagrams and Class Diagram for 10 Implementation Use Cases	5	T3
T6	Requirements Analysis	Create Scenarios and Object Diagram for Use Cases	5	T3
M1		Write up SRD Sections 4 and 5	1	T1, T4, T5, T6
T7		Write up SRD Sections 1-3, 6-9	22	T1
M2		Compile SRD	2	M1, T7
T8	Development	Set up Project Environments	1	T1
T9	Development	Create React Mock-up	23	T8
M3		SRD Presentation	3	M2, T9
T10	Detailed Design	Decide on the Subsystem, Data Management.	5	M2
T11	Detailed Design	Create Detailed Class Design for the Subsystems.	8	T10
M4		Write up DD Section 2 and 3	12	T11
T12		Write up DD Sections 1, 4-7	25	M2
M5		Compile DD	6	M4, T12
T13	Development	Implement Front-End Subsystems	15	M3, T10
T14	Development	Implement Back-End Logic Subsystems	15	T10
T15	Development	Implement Back-End Datastore Subsystems	15	T10
M6	Development	Integrate the Subsystem. The Implementation is Completed.	3	T13, T14, T5
T16		Update Sections from DD and SRD to their FSD version.	3	M5, M6
T17	Testing	Set up Testing Environment and Formulate Test Cases	5	M6
T18	Testing	Perform Testing Process	5	T17
M7		Write up FSD Section 7 and Appendix F	5	T18
T19	Development	System Verification	3	T18
T20		Write up FSD Sect. 4-6, Appendix A-E, G	5	T19
M8		Complete FSD	4	T16, M7, T20
M9		FSD Presentation	3	M8

Table 4: Task Schedule for the Project

4 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.

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.

4.1.1 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

4.1.2 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.

4.1.2.1 Create Event

Use Case ID: SOS01 - Create Event

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 form, which shall present them with a template for data entry.

3. The Organizer shall enter the following data:
    - **Event Name**
    - **Event Date and Time**
    - **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. Use case ends when the system receives the Event specifications, generates a **unique event id** and publishes the Event according to the given specifications.

- **Relevant requirements:**

  None

- **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. SOS21 – Avoid Time Conflicting Events

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

4.1.2.2 Grant Organizer Role

Use Case ID: SOS2 – Grant Organizer Role

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
  a) No previous training or knowledge.
  b) Tutorial or Help frame should be provided.
  c) Organizer should take less than 10 minutes to complete the invitation.
- Reliability
  a) Mean Time to Failure – 1% failure yearly is acceptable.
  b) Availability – 30 minutes in a 24-hour period for backup and maintenance.
- Performance
  a) Privilege Checks should be done within 2 seconds.
  b) The system should handle 20 privilege checks in 1 minute.

- Supportability
    a) Should be supported by all browsers.
- Implementation
    a) Using Java-based software for back-end.

Modification History

Owner: Armando J. Ochoa

Initiation date: 09/01/2019

Date last modified: 09/15/2019

4.1.2.3 Earn Points by Attending an Event

Use Case ID: SOS3 – Earn Points by Attending an Event

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, 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
    a) No previous training or knowledge. The system should respond without user interaction after the attendance is completed.

- Reliability
    a) Meant Time to Failure: 5% failure monthly is acceptable.
- Performance
    a) The system should be able to handle 20 requests in 1 minute.
    b) The system should update the Event, Member, and Organization logs within 2 seconds.
- Supportability
    a) Point earning should be supported by Chrome, Mozilla, and IE.
- Implementation
    a) 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

4.1.2.4 Attending an Event

Use Case ID: SOS04 - 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 an organization and is attending an event hosted by said organization.
- **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 occurring.
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 organization 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 organization leader.

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

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

Constraints:

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

4.1.2.5 Ensure User Access

Use Case ID: SOS05 – Ensure User Access

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

a) User must be aware of their privileges and what actions those privileges permit.

- Reliability
    a) Mean Time to Failure – 1% failure yearly is acceptable.
    b) Availability – 30 minutes in a 24-hour period for backup and maintenance.
- Performance
    a) Privilege Checks should be done within 2 seconds.
    b) The system should handle 20 privilege checks in 1 minute.
- Supportability
    a) Should be supported by all browsers.
- Implementation
    a) Using Java-based software for back-end.

Modification History

Owner: Armando J. Ochoa

Initiation date: 09/01/2019

Date last modified: 09/15/2019

4.1.2.6 Ensure User Profile Privacy

Use Case ID: SOS6 – Ensure User Profile Privacy

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 – Edit Profile

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
  a) User must be aware of their privileges and what actions those privileges permit.

- Reliability
    a) Mean Time to Failure – 1% failure yearly is acceptable.
    b) Availability – 30 minutes in a 24-hour period for backup and maintenance.
- Performance
    a) Privilege Checks should be done within 2 seconds.
    b) The system should handle 20 privilege checks in 1 minute.
- Supportability
    a) Should be supported by all browsers.
- Implementation
    a) Using Java-based software for back-end.

Modification History

Owner: Armando J. Ochoa

Initiation date: 09/01/2019

Date last modified: 09/15/2019

4.1.2.7 Edit Profile

Use Case ID: SOS7 – Edit Profile

Use Case Level: Security

Details:

- **Actor:** User
- **Pre-conditions:**
    1. User have already signed up.
    2. User is currently at their profile page.
- **Description:**
    1. Use case begins when user clicks on the edit profile button.
    2. The system then will retrieve current user data by contacting the data storage and send the data back to the front-end.
    3. The page shall display the retrieved data in an input form which will allow the user to modify the data in the edit profile form:
        - Email
        - Phone number
        - Privacy

- Date Of Birth

4. The user inputs the modified data and clicks on the submit button.
5. The system shall ask the user for their password.
6. The user inputs their password and clicks confirm.
7. The system shall transmit the modified data to the data storage.
8. The case ends when there is a confirmation message.

- **Relevant requirements:**

  None.

- **Post-conditions:**

1. User information in the datastore has updated values.
2. Profile page has been updated with the updated values.

Alternative Courses of Action:

1. In step D.4, it is possible that the user closes the input form without clicking the submit button. In that case system shall not change the current user information.

Extensions:

None.

Exceptions:

None.

Concurrent Uses:

None

Related Use Cases:

SOS6 – Ensure User Profile Privacy

Decision Support

Frequency: On average, 20 Users will change their privacy settings on a given week.

Criticality: Low. User-set privacy is a secondary feature of the system.

Risk: Medium. This does not require any complex background knowledge except for some basic knowledge about access control.

Constraints:

- Usability
  a) User will take about 20 seconds to find and use this piece of functionality.
- Reliability
  a) Mean Time to Failure – 5% failure monthly is acceptable.

b) Availability
   - Downtime for Login Back-up – 30 minutes in a 24-hour period.
   - Downtime for Maintenance – 1 hour in a 2 weeks period.

- Performance
  a) Privilege Checks should be done within 2 seconds.
  b) The system should handle 20 privilege checks in 1 minute.
- Supportability
  a) Should be supported by all browsers.
- Implementation
  a) Using Java-based software for back-end.

Modification History

Owner: Kian Maroofi

Initiation date: 09/10/2019

Date last modified: 09/27/2019

4.1.2.8 Sharing

Use Case ID: SOS08 - Sharing

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
  a) No previous training or knowledge.
- Reliability
  a) Meant Time to Failure: 5% failure monthly is acceptable.

- Performance
    a) The system should be able to handle 20 requests in 1 minute.
    b) Sharing should happen instantly.
- Supportability
    a) Point earning should be supported by Chrome, Mozilla, and IE.
- Implementation
    a) 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

4.1.2.9 Member Ranking

Use Case ID: SOS9 – Member Ranking

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
  a) User must be aware of their privileges and what actions those privileges permit.
- Reliability
  a) Mean Time to Failure – 1% failure yearly is acceptable.
  b) Availability – 30 minutes in a 24-hour period for backup and maintenance.
- Performance
  a) Privilege Checks should be done within 2 seconds.
  b) The system should handle 20 privilege checks in 1 minute.
- Supportability

a) Should be supported by all browsers.

- Implementation
    a) Using Java-based software for back-end.

Modification History

Owner: Kian Maroofi

Initiation date: 09/10/2019

Date last modified: 09/15/2019

4.1.2.10 Access Events by Location

Use Case ID: SOS10 – Access Events by Location

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 webpage 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
    a) User must be aware of their privileges and what actions those privileges permit.
- Reliability
    a) Mean Time to Failure – 1% failure yearly is acceptable.
    b) Availability – 30 minutes in a 24-hour period for backup and maintenance.
- Performance

a) Privilege Checks should be done within 2 seconds.

b) The system should handle 20 privilege checks in 1 minute.

- Supportability
    a) Should be supported by all browsers.
- Implementation
    a) Using Java-based software for back-end.

Modification History

Owner: Kian Maroofi

Initiation date: 09/10/2019

Date last modified: 09/15/2019

4.1.2.11 Score System

Use Case ID: SOS22 – Score System

Use Case Level: User Goal

Details:

- **Actor:** Organizer
- **Pre-conditions:**
    1. Organizer have already posted an event.
    2. Users (Members & Guests) enabled access to their current location via GPS.
- **Description:**
    1. Use case begins when the Organizer has posted a new event on the platform.
    2. The system shall provide an input feature such that provides the organizer a score definition system.
    3. The score that each attendee earns shall able to be defined by the organizer at the time of creating the event.
    4. The score must be selected from the following set which depends on the importance of the event which organizer defines. Score set is 5, 10, 15, 20, 25.
    5. The case ends when the event is created by the organizer.
- **Relevant requirements:**

    GPS and Geolocation Services available in most web browsers (Desktop and Mobile).
- **Post-conditions:** None.
- **Alternative Courses of Action:**

Extensions:

None.

Exceptions:

None.

Concurrent Uses:

None

Related Use Cases:

SOS4 – Attending an Event

SOS9 – Member Ranking

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. Does not require specialized knowledge.

Constraints:

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

Modification History

Owner: Kian Maroofi

Initiation date: 09/10/2019

Date last modified: 09/22/2019

4.1.2.12 Set Up Two Factor Authentication (2FA)

Use Case ID: SOS12 – Set Up Two Factor Authentication (2FA)

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

Modification History

Owner: Kian Maroofi

Initiation date: 09/10/2019

Date last modified: 09/15/2019

4.1.2.13 Kick Privileges

Use Case ID: SOS13 – Kick Priveleges

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 Organizer will provide a short description to the member why they are being kicked from their organization.
    6. The Organizer will send the request by selecting the send button.
    7. The system shall notify Organizer if the request was submitted correctly.
    8. Use case ends when the system will remove the member from the organization.
- **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 organization.

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 organization to kick.

Related Use Cases: None.

Decision Support

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

Criticality: High. Allows the Organizer 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:
    a) No previous training required.
    b) On average the user should take 2 minutes to complete the kick request to the system.
- Reliability
    a) Mean time to failure – 5% failures for every month of operation is acceptable.
    b) Availability – Down time for Login Back-up 30 minutes in a 24 hour period.
- Performance
    a) Request should be sent and saved within 6 seconds.
    b) System should be able to handle 100 requests in 1 minute.
- Supportability
    a) The Event Creation should be supported by Chrome, Mozilla, and IE.
- Implementation
    a) 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

4.1.2.14 Create Roles

Use Case ID: SOS14 – Create Roles

Use Case Level: User Privileges

Details:

- **Actor:** Organizer.
- **Pre-conditions:**
    1. Organizer has successfully logged onto the system.
    2. Organizer has Manage Roles Privileges.
    3. The application is open.
- **Description:**
    1. Use case begins when Organizer clicks on the **Organization Roles** tab within the **Organization Management** view.
    2. The system shall display a view with a description of the current organization roles along with options to **Edit** the roles and **Create New Role**.
    3. The Organizer clicks on the **Create New Role** button.
    4. The system shall prompt the Organizer with a Role Creation form, which shall present them with a template for data entry.
    5. The Organizer shall enter the following data:
        - **Role Name**
        - **Privileges of the Role**, which come from a set list of privileges including:
            i. Kick
            ii. Invite
            iii. Promote
            iv. Manage Event
            v. Manage Roles
        - **Security Requirement**, which come from a set list including:
            i. 2-Factor Authentication
            ii. Organization-Defined Password
    6. The Organizer shall complete the Role Creation by selecting the **Submit** button.
    7. The System shall notify the Organizer that the Role was added correctly.
    8. Use Case ends when the system adds the new role to the Organization.

- **Relevant requirements:**

  None

- **Post-conditions:**

  1. The request to create a new role is saved by the system
  2. The new role appears as an option when assigning roles to Organizers.

Alternative Courses of Action

1. In step D.2, the list of roles can be sorted alphabetically or by privileges.
2. In step D.5, if any of the fields are left empty the system will require the user to fill in those requirements.
3. In step D.6, the Organizer has the option to **Cancel** the new role creation.

Exceptions:

1. The organization administrator attempts to make a role that already exists.

Concurrent Use Cases:

None.

Related Use Cases:

None.

Decision Support

Frequency: On average 5 role creation requests are made every 3 months by organizer.

Criticality: High. Allows the Organizer to give different privileges to users to ensure that organization management runs smoothly.

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

Constraints:

- Usability:
  a) No previous training required.
  b) On average the user should take 2 minutes to complete the promotion request to the system.
- Reliability
  a) Mean time to failure – 5% failures for every month of operation is acceptable.
  b) Availability – Down time for Login Back-up 30 minutes in a 24 hour period.
- Performance

a) Request should be sent and saved within 6 seconds.

b) System should be able to handle 100 requests in 1 minute.

- Supportability
    a) Shall be supported by Chrome, Mozilla, and IE.
- Implementation
    a) 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: 09/22/2019

4.1.2.15 Notifications

Use Case ID: SOS15 - Notifications

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 organizations tab.
    2. The system shall provide the member with a set of cards that represent the organizations that they are a part of.
    3. The member will click on the organization that they want to obtain notifications for.
    4. The member will click on get event news button on the organization 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 organization.

- **Relevant requirements:**

  None

- **Post-conditions:**

  1. The request to receive notifications from the organization 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:
  a) No previous training required.
  b) On average the user should take 2 minutes to complete the notification request to the system.
- Reliability
  a) Mean time to failure – 5% failures for every month of operation is acceptable.
  b) Availability – Down time for Login Back-up 30 minutes in a 24 hour period.
- Performance
  a) Request should be sent and saved within 6 seconds.
  b) System should be able to handle 100 requests in 1 minute.
- Supportability

a) The Event Creation should be supported by Chrome, Mozilla, and IE.

- Implementation
    a) 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

4.1.2.16 Create Organization

Use Case ID: SOS16 – Create Organization

Use Case Level: High-Level

Details:

- **Actor:** User
- **Pre-conditions:**
    1. User has an account in our application.
    2. User is successfully logged into the application.
- **Description:**
    1. Use case begins when User clicks on the Organization tab in their current page (home page for example) and the homepage refreshes and provides the Organizer with the Organization page.
    2. The organization page shall provide the User with a set of cards that represent the organizations that they are a part of and a Create Organization option.
    3. The User will click on the Create Organization option.
    4. The organization page shall provide the User with a form to fill out, asking for the following details:
        - **Organization Name**
        - **Organization Description**
        - **Requirements for Joining**
        - **Privacy of the Organization** (whether it's open to others or not).
    5. The system shall notify the User that the request was submitted correctly by showing a notification in the Organization page.

6. Use case ends when the organization page the displays the new organization that the User has created a new organization.

- **Relevant requirements:**

  None

- **Post-conditions:**

  1. The request to create an organization is stored in the system.
  2. The organization is shown to members depending on its privacy settings.
  3. The User has gained owner status with respect to the created organization.

Alternative Courses of Action:

1. In step D.4 the user has the option to cancel the creation of their organization.
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 an organization.

Exceptions:

1. If the User tries to make an organization that already exists, then they will get an error message.

Concurrent Use Cases:

None.

Related Use Cases:

None.

Decision Support

Frequency: On average 20 organization creation requests are made monthly by the User.

Criticality: High. Allows the User to create an organization which allows new communities to grow around campus.

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

Constraints:

- Usability:
  a) No previous training required.

b) On average the user should take 2 minutes to complete the notification request to the system.

- Reliability
    a) Mean time to failure – 5% failures for every month of operation is acceptable.
    b) Availability – Down time for Login Back-up 30 minutes in a 24 hour period.
- Performance
    a) Request should be sent and saved within 6 seconds.
    b) System should be able to handle 200 requests in 1 minute.
- Supportability
    a) The Event Creation should be supported by Chrome, Mozilla, and IE.
- Implementation
    a) 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

4.1.2.17 Cancel an Event
Use Case ID: SOS17 - Cancel an Event

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 organization.
- **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 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:
  a) No previous training required.
  b) On average the user should take 2 minutes to complete the notification request to the system.
- Reliability
  a) Mean time to failure – 5% failures for every month of operation is acceptable.

b) Availability – Down time for Login Back-up 30 minutes in a 24 hour period.

- Performance
    a) Request should be sent and saved within 6 seconds.
    b) System should be able to handle 100 requests in 1 minute.
- Supportability
    a) Shall should be supported by Chrome, Mozilla, and IE.
- Implementation
    a) 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

4.1.2.18 Create Task
Use Case ID: SOS18 – Create Task

Use Case Level: User Goal

Details:

- **Actor:** Organizer
- **Pre-conditions:**
    1. An Event has been created by an Organization
    2. Organizer has privileges on the given Organization, and is logged in.
- **Description:**
    1. Use case begins when the Organizer clicks on the Add **Task** in the edit view of an Event page.
    2. The system shall prompt the Organizer with an **Add Task** form, which shall present them with a template for data entry.
    3. The Organizer shall input the following data in the template:
        - **Task Name**
        - **Task Description**
        - **Expected Number of Participants**

4. The Organizer shall finish adding the task by selecting the **Complete** button.
5. The page shall notify the Organizer that the task was added to the Event.
7. Use case ends when the system updates the Event with the task according to the specification.

- **Relevant requirements:**

  None

- **Post-conditions:**

1. The Event has been updated so that it shows the details pertaining to the task in the backing database, and this change is reflected in the Event's page.

Alternative Courses of Action:

1. In step D.4, the Organizer has the option to **Cancel** the task creation.

Exceptions:

None

Concurrent Use Cases:

None.

Related Use Cases:

None.

Decision Support

Frequency: On average, 20 tasks are added to events a week.

Criticality: Medium. Not all events require tasks to be complete, so not all users will use this functionality.

Risk: Medium. Implementation does not require any complex specialized knowledge besides a database system.

Constraints:

- Usability:
  a) Requires minimal training.
  b) One or two help frames on the Help page shall be provided explaining how to add tasks.
  c) On average the user should less than 5 minutes to complete the notification request to the system.
- Reliability
  a) Mean time to failure – 5% failures for every 24 hours of operation is acceptable.

b) Availability
   - Downtime for Login Back-up – 30 minutes in a 24-hour period.
   - Downtime for Maintenance – 1 hour in a 2 weeks period.

- Performance
  a) Request should be sent and saved within 6 seconds.
  b) System should be able to handle 100 requests in 1 minute.
- Supportability
  a) Shall be supported by Chrome, Mozilla, and IE.
- Implementation
  a) The implementation shall use JS React for front-end, and Java-based software for back-end.

Modification History

Owner: Yovanni Jones

Initiation date: 09/02/2019

Date last modified: 09/22/2019

4.1.2.19 Request Organization Information
Use Case ID: SOS19 – Request Organization Information

Use Case Level: Access Organization Page

Details:

- **Actor:** Organizer
- **Pre-conditions:**
  1. User is logged into the system.
- **Description:**
  1. Use case begins when the User opens the Sidebar and clicks on the **Organization** tab.
  2. The system shall change the view to the Organizations view, listing all the available organizations.
  3. The User selects an Organization by clicking on it.
  4. Use Case ends when the system changes the view to the Organization's page, which shall contain a description of the Organization and Event information.

- **Relevant requirements:**

  None

- **Post-conditions:**

  1. The view of the User has changed to the Organization's page.

Alternative Courses of Action:

1. In step D.4, if the User has privileges over the chosen Organization, a privileged view providing access to the Event Creation, Task Creation, and other Organization management tabs will be displayed instead.

Exceptions:

1. The page for the Organization cannot be found or has been deleted.

Concurrent Use Cases:

None.

Related Use Cases:

None.

Decision Support

Frequency: On average, asking for a description of what the organization is could happen 1000 times a day.

Criticality: High. This a core functionality of the system.

Risk: Low. Requires no specialized knowledge.

Constraints:

- Usability:
  a) No previous training required.
  b) Should take under 5 minutes to acquire info on organization
- Reliability
  a) Mean time to failure – 5% failures for every 24 hours of operation is acceptable.
  b) Availability
     - Downtime for Login Back-up – 30 minutes in a 24-hour period.
     - Downtime for Maintenance – 1 hour in a 2 weeks period.

- Performance
    a) System should be able to handle 100 requests in 1 minute.
- Supportability
    a) Shall be supported by Chrome, Mozilla, and IE.
- Implementation
    a) The implementation shall use JS React for front-end, and Java-based software for back-end.

Modification History

Owner: Yovanni Jones

Initiation date: 09/02/2019

Date last modified: 09/22/2019

4.1.2.20 Remove Organization
Use Case ID: SOS20 – Remove Organization

Use Case Level: User Goal

Details:

- **Actor:** Organizer
- **Pre-conditions:**
    1. Organizer is the owner of the target Organization.
- **Description:**
    1. Use case begins when the Organizer clicks on the **Remove Organization** button on the Organization's Settings page.
    2. The system shall prompt the Organizer with a form, requesting for the Organization's unique ID number.
    3. The Organizer shall enter the unique ID number.
    4. The Organizer shall complete the deletion by selecting the **Confirm** button.
    5. The system shall remove all the future Events by the Organization from the Event views and delete their records. Past Events shall be kept and displayed on the User's page.
    6. The system shall revoke the Member status from Users who were members of the Organization. Same thing for Organizers.

7. Use case ends when the system has notified the relevant users and saved a record of the deletion.

- **Relevant requirements:**

  None

- **Post-conditions:**

  1. The Organization has been deleted from the system and it will no longer appear on the Organization tab.
  2. The future Events by the Organization have been deleted.
  3. Users with Member or Organizer status on the Organization have been stripped of these status.
  4. A record has been saved of the deletion request.

Alternative Courses of Action:

1. In step D.2, the Organizer has the option to **Cancel**.

Exceptions:

The Organizer is missing the required permissions for deletion (is not the owner).

The Organization has special privileges preventing it from being deleted.

Concurrent Use Cases:

None.

Related Use Cases:

None.

Decision Support

Frequency: On average, 3 organizations removed per week.

Criticality: High. Deletions and disbandment should be handled correctly and the information on the website should be kept up-to-date.

Risk: Medium. Implementation does not require any complex specialized knowledge, but a secure implementation is required to make sure no unauthorized person is able to delete an Organization.

Constraints:

- Usability:
  a) Requires minimal training.

b) One or two help frames on the Help page shall be provided.

c) On average the user should less than 5 minutes to complete the notification request to the system.

- Reliability
  a) Mean time to failure – 5% failures for every 24 hours of operation is acceptable.
  b) Availability
    - Downtime for Login Back-up – 30 minutes in a 24-hour period.
    - Downtime for Maintenance – 1 hour in a 2 weeks period.
- Performance
  a) Request should be sent and saved within 6 seconds.
  b) System should be able to handle 100 requests in 1 minute.
- Supportability
  a) Shall be supported by Chrome, Mozilla, and IE.
- Implementation
  a) The implementation shall use JS React for front-end, and Java-based software for back-end.

Modification History

Owner: Yovanni Jones

Initiation date: 09/02/2019

Date last modified: 09/22/2019

4.1.2.21 Avoid Time Conflicting Events

Use Case ID: SOS21 – Avoid Time Conflicting Events

Use Case Level: User Goal

Details:

- **Actor:** Organizer
- **Pre-conditions:**
  1. Organizer has privileges on the given Organization, and is logged in.
- **Description:**
  1. Use case begins when the Organizer clicks on the **Create Event** on the administration page of their Organization.

2. The system shall prompt the Organizer with an Event Creation form, which shall present them with a template for data entry.
3. The Organizer shall enter the required data (see SOS1 – Create Event).
4. The system shall check the **Event Date and Time** against the other Events of the Organization.
5. If a conflicting Event is found, the system shall notify the Organizer of this conflict and present the Organizer with a new form.
6. The Organizer shall enter the following data:
    - **New Event Date and Time** which will be preset with the conflicting date.
7. The Organizer complete the Event Creation by selecting the **publish** button.
8. The system shall notify the Organizer that the event was published correctly.
9. Use case ends when the system receives the Event specification, generates a unique event id and publishes the Event according to the specifications.

- **Relevant requirements:**

  None

- **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.6., the Organizer has the option of publishing the event at the original conflicting date by clicking **publish** without changing the default conflicting date.

Exceptions:

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

Concurrent Use Cases:

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

Modification History

Owner: Yovanni Jones

Initiation date: 09/02/2019

Date last modified: 09/22/2019

4.1.2.22 Registration

Use Case ID: SOS22 – Registration

Use Case Level: User Goal

Details:

- **Actor:** User

- **Pre-conditions:**
    1. The User does not have an account on the site.
- **Description:**
    1. Use case begins when the User presses the **Register** button on the log-in/register page.
    2. The system shall prompt the User with a **Registration** form, which shall present them with a template for data entry.
    3. The Organizer shall input the following data in the template:
        - **User Name**
        - **Email**
        - **Password**
        - **Confirm Password**
    4. The User shall complete the registration by selecting the **Ok** button.
    5. The system shall confirm that the registration was successful.
    6. Use case ends when the User is automatically logged into the system and the view is moved to home.
- **Relevant requirements:**

    None
- **Post-conditions:**

    None

Alternative Courses of Action:

1. In step D.3, If any of the fields have incorrect information or are left blank system will respond with a message saying that proper credentials should be entered.

Exceptions:

None

Concurrent Use Cases:

None.

Related Use Cases:

None.

Decision Support

Frequency: On average, 20 tasks are added to events a week.

Criticality: Medium. Not all events require tasks to be complete, so not all users will use this functionality.

Risk: Medium. Implementation does not require any complex specialized knowledge besides a database system.

Constraints:

- Usability:
    a) Requires minimal training.
    b) One or two help frames on the Help page shall be provided explaining how to add tasks.
    c) On average the user should less than 5 minutes to complete the notification request to the system.
- Reliability
    a) Mean time to failure – 5% failures for every 24 hours of operation is acceptable.
    b) Availability
        - Downtime for Login Back-up – 30 minutes in a 24-hour period.
        - Downtime for Maintenance – 1 hour in a 2 weeks period.
- Performance
    a) Request should be sent and saved within 6 seconds.
    b) System should be able to handle 100 requests in 1 minute.
- Supportability
    a) Shall be supported by Chrome, Mozilla, and IE.
- Implementation
    a) The implementation shall use JS React for front-end, and Java-based software for back-end.

Modification History

Owner: Yovanni Jones

Initiation date: 09/02/2019

Date last modified: 09/22/2019

4.1.2.23 Admin: Manual Deletion of Events

Use Case ID: SOS23 – Admin: Manual Deletions of Events

Use Case Level: Administrator Role

Details:

- **Actor:** Administrator
- **Pre-conditions:**
    1. Administrator is logged into the system.
    2. A User has created an Event which violates privacy agreements, terms of use, promotes violence, or is otherwise deemed inadmissible.
- **Description:**
    1. Use case begins when the Event is presented to the Administrator to be reviewed, either because it has been reported by Users, or because it has been found inadmissible by the Administrator.
    2. The Administrator reviews the event with a system that checks the spelling for any misconduct. Nouns, Verbs, Adjectives, etc. that may imply some sort of malicious intention.
    3. The Administrator clicks on **Quarantine Event** to initiate a removal process, giving a reason as to why this measure was taken.
    4. The system shall delete the Event from the Events and Organization page.
    5. The system shall notify that the Event will be deleted, citing the reason given by the Administrator. A standard warning about misconduct shall be issued to the User.
    6. The Use Case ends when the system records the request for deletion, as well as record the infringement under the User's information for the Administrator to see in the future.
- **Relevant requirements:**

    None
- **Post-conditions:**
    1. The User who created the account will had been warned about the action. If continued infringements occur, he or she will be barred from creating more events or event banned from the system.
    2. The Event in question will had been deleted from public view.

Alternative Courses of Action:

1. In step D.3, the Administrator has an option to request more information by clicking **Inquire**, which will open an investigation to the Event and contact the Organization and the User

Exceptions:

None

Concurrent Use Cases:

None.

Related Use Cases:

None.

Decision Support

Frequency: On average, the system will have to do 5-10 checks daily.

Criticality: High. Users will be making a lot of posts so making sure they are not dangerous is crucial.

Risk: Medium. Implementation does not require any complex specialized knowledge besides a database system.

Constraints:

- Usability:
    a) Will require training for Administrator to deal with and recognize threats, but the system itself should be easy to use.
    b) One or two help frames explaining the Quarantine and Inquire process should be provided.
- Reliability
    a) Mean time to failure – 5% failures for every 24 hours of operation is acceptable.
    b) Availability
        - Downtime for Login Back-up – 30 minutes in a 24-hour period.
        - Downtime for Maintenance – 1 hour in a 2 weeks period.
- Performance
    a) Request should be sent and saved within 6 seconds.

b) System should be able to handle 100 requests in 1 minute.

- Supportability
  a) Shall be supported by Chrome, Mozilla, and IE.
- Implementation
  a) The implementation shall use JS React for front-end, and Java-based software for back-end.

Modification History

Owner: Yovanni Jones

Initiation date: 09/02/2019

Date last modified: 09/22/20

4.1.2.24 Admin: Extended Privileges

Use Case ID: SOS24 – Admin: Extend Priveleges

Use Case Level: Administrator Role

Details:

- **Actor:** Administrator
- **Pre-conditions:**
  1. Administrator is logged into the system.
- **Description:**
  1. Use case begins when an Administrator accesses a User Profile, Organization Page, or Event Page.
  2. The system shall present the Administrator with privilege views over those pages, giving a more flexible control on each Event, Organization, and enabling monitoring and observing normal Users (Members, Organizers) for them.
  3. The Use Case ends when these pages are presented to the Administrator.
- **Relevant requirements:**

  None
- **Post-conditions:**

  None

Alternative Courses of Action:

None

Exceptions:

None

Concurrent Use Cases:

None.

Related Use Cases:

None.

Decision Support

Frequency: On average, 25 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.

Constraints:

- Usability:
    a) User must be aware of their privileges and what actions those privileges permit.
    b) Some training about privileges is required.
    c) One or two help frames explaining the extent of Administrator Privileges, Roles, and Expectations shall be provided.
- Reliability
    a) Mean time to failure – 5% failures for every 24 hours of operation is acceptable.
    b) Availability
        - Downtime for Login Back-up – 30 minutes in a 24-hour period.
        - Downtime for Maintenance – 1 hour in a 2 weeks period.
- Performance
    a) Privilege Checks should be done within 2 seconds.
    b) The system should handle 20 privilege checks in 1 minute.
- Supportability
    a) Shall be supported by Chrome, Mozilla, and IE.
- Implementation
    a) The implementation shall use JS React for front-end, and Java-based software for back-end.

Modification History

Owner: Kian Maroofi

Initiation date: 09/02/2019

Date last modified: 09/22/20

4.1.2.25 Filter Events

Use Case ID: SOS25 – Filter Events

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

4.1.2.26 Invite User from Roster
Use Case ID: SOS26 – Invite User from Roster

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 show 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 confirmation.
    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 clicks "*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:
    a) Might require light training.
    b) One help frame on the Help page provided.
    c) On average the user should take 1 minute to update their roster.
- Reliability
    a) Mean time to failure – 1% failures for every month of operation is acceptable.
    b) Availability – Down time for Login Back-up 30 minutes in a 24-hour period.
- Performance
    a) System should be able to handle 100 requests in 1 minute.
- Supportability
    a) The Event Creation should be supported by Chrome, Mozilla, and IE.
- Implementation
    a) 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

4.1.2.27 Remove User from Roster
Use Case ID: SOS27 – Remove User From Roster

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 show 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 confirmation.
    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 clicks "*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:
    a) Might require light training.
    b) One help frame on the Help page provided.
    c) On average the user should take 1 minute to update their roster.
- Reliability
    a) Mean time to failure – 1% failures for every month of operation is acceptable.
    b) Availability – Down time for Login Back-up 30 minutes in a 24-hour period.
- Performance
    a) System should be able to handle 100 requests in 1 minute.
- Supportability
    a) The Event Creation should be supported by Chrome, Mozilla, and IE.
- Implementation
    a) 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

4.1.2.28 User RSVP
Use Case ID: SOS28 - RSVP

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 if they agree with the terms set by the hosts.

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

Constraints:

- Usability:
    a) Requires no training.
    b) On average the user should take 20 seconds to perform an RSVP.

- Reliability
    a) Mean time to failure – 1% failures for every month of operation is acceptable.
    b) Availability – Down time for Login Back-up 30 minutes in a 24-hour period.
- Performance
    a) RSVP requests should be processed within 5 seconds.
    b) The system shall be consistent when handling RSVP requests.
- Supportability
    a) The Event Creation should be supported by Chrome, Mozilla, and IE.
- Implementation
    a) 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

4.1.2.29 Unauthorized Organization Management
Use Case ID: SOS29 – Unauthorized Organization Management

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:
  a) Requires no training.
  b) 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
  a) Mean time to failure – 1% failures for every month of operation is acceptable.
  b) Availability – Down time for Login Back-up 30 minutes in a 24-hour period.
- Performance
  a) Should be able to produce results within 3 seconds.
- Supportability
  a) The Event Creation should be supported by Chrome, Mozilla, and IE.

- Implementation
    a) 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

4.1.2.30 Unauthorized Event Creation
Use Case ID: SOS30 – Unauthorized Event Creation

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 managerial 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:
    a) Requires no training.
    b) 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
    a) Mean time to failure – 1% failures for every month of operation is acceptable.
    b) Availability – Down time for Login Back-up 30 minutes in a 24-hour period.
- Performance
    a) Should be able to produce results within 3 seconds.
- Supportability
    a) The Event Creation should be supported by Chrome, Mozilla, and IE.
- Implementation
    a) 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

4.1.2.31 Log in
Use Case ID: SOS31 – Log in

Use Case Level: User Goal

Details:

- **Actor:** User
- **Pre-conditions:**
    1. The User has an account on the SOS site.
- **Description:**
    1. Use case begins when the user is in the **Log-In** page of the site.
    2. The login page shall provide an input form with to following parameters:
        - **Email address**
        - **Password**
    3. The user inputs their email and password and then clicks on login.
    4. The system shall verify if the email and password match.
    5. Use case ends when system allows the user to login.
- **Relevant requirements:**

  None
- **Post-conditions:**
    1. the user is redirected to the **Home** page.

Alternative Courses of Action:

1. In step D.4, if the user types an invalid password or email then the system will notify them that their "email and password do not match."

Exceptions:

None

Concurrent Use Cases:

None.

Related Use Cases:

None.

Decision Support

Frequency: On average, up to 10000 requests daily.

Criticality: High. Allows the user to log-in to view their organizations and nearby events.

Risk: Low. Implementing this use case doesn't requires specified knowledge.

Constraints:

- Usability:
    a) Requires no training.
    b) On average the user should take less than 10 seconds to type their information and attempt to log in.
- Reliability
    a) Mean time to failure – 1% failures for every month of operation is acceptable.
    b) Availability – Down time for Login Back-up 30 minutes in a 24-hour period.
- Performance
    a) Should be able to produce results within 3 seconds.
- Supportability
    a) The Event Creation should be supported by Chrome, Mozilla, and IE.
- Implementation
    a) The implementation shall use JS React for front-end, and Java-based software for back-end.

Modification History

Owner: Anthony Sanchez-Ayra

Initiation date: 09/06/2019

Date last modified: 09/16/2019

4.1.2.32 Log Out
Use Case ID: SOS32 – Log out

Use Case Level: User Goal

Details:

- **Actor:** User
- **Pre-conditions:**
    1. The User is currently logged into the SOS page.
- **Description:**
    1. Use case begins when the user clicks on the **Sign Out** button.
    2. The current page the user is in will call a system call to log the user out.
    3. The system will then attempt to log the user out of the webpage.
    4. Use case ends when website redirects the user to the **Login** page.
- **Relevant requirements:**

  None
- **Post-conditions:**

  None.

Alternative Courses of Action:

None.

Exceptions:

None.

Concurrent Use Cases:

None.

Related Use Cases:

None.

Decision Support

Frequency: On average, up to 10000 requests daily.

Criticality: High. Allows the user to log-out to make sure that no other user can tamper with their account if they were to access the site from the same computer.

Risk: Low. Implementing this use case doesn't requires specialized knowledge.

Constraints:

- Usability:
    a) Requires no training.
    b) On average the user should take less than 5 seconds to find the sign out button and click on it.
- Reliability
    a) Mean time to failure – 1% failures for every month of operation is acceptable.
    b) Availability – Down time for Login Back-up 30 minutes in a 24-hour period.
- Performance
    a) Should be able to produce results within 3 seconds.
- Supportability
    a) The Event Creation should be supported by Chrome, Mozilla, and IE.
- Implementation
    a) The implementation shall use JS React for front-end, and Java-based software for back-end.

Modification History

Owner: Anthony Sanchez-Ayra

Initiation date: 09/06/2019

Date last modified: 09/16/2019

4.2 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. Following that, Section 4.2.2., Implemented Use Case Diagram contains a UML Use Case diagram describing the Use Cases that are currently implemented.

4.2.1 Full Use Case Diagram

The Use Case diagram describing the whole system is shown in Figure 1 to

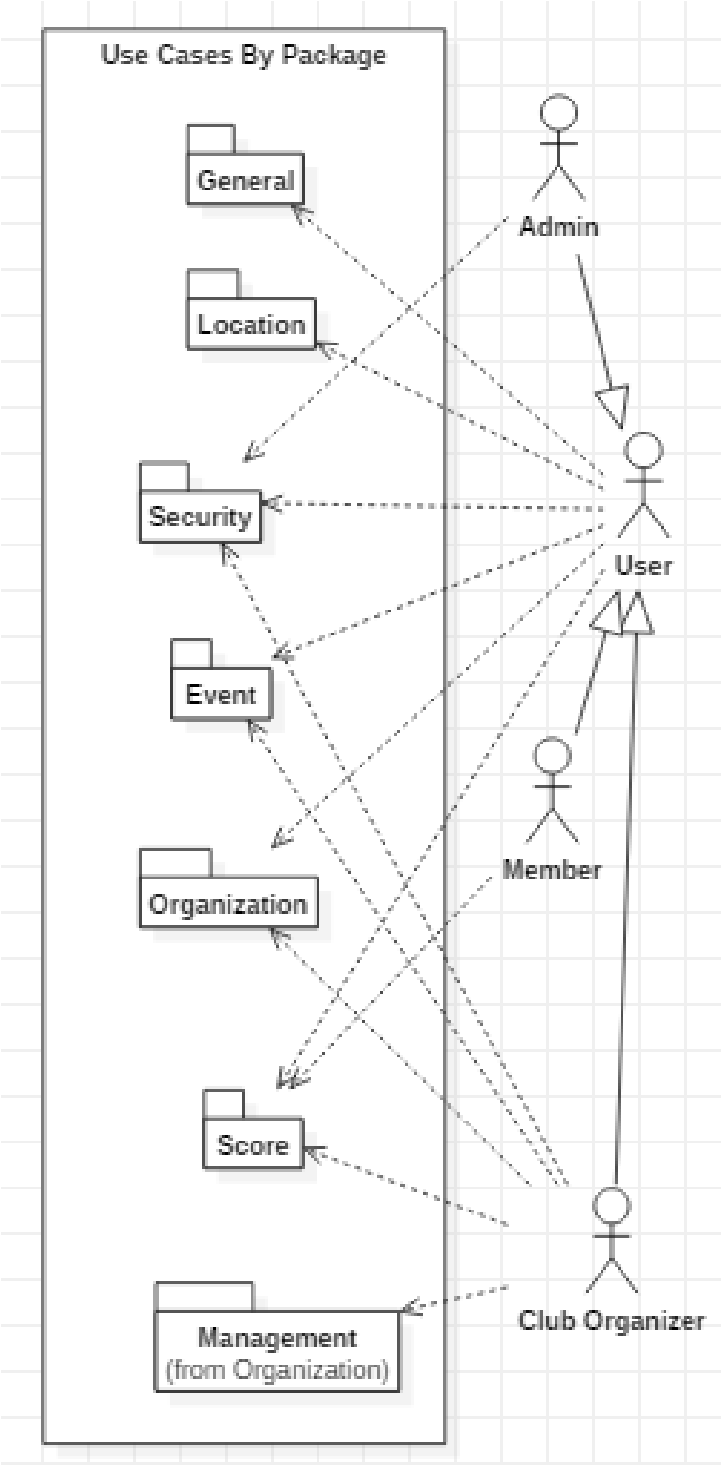

Figure 1: Use Case diagram for the whole system.

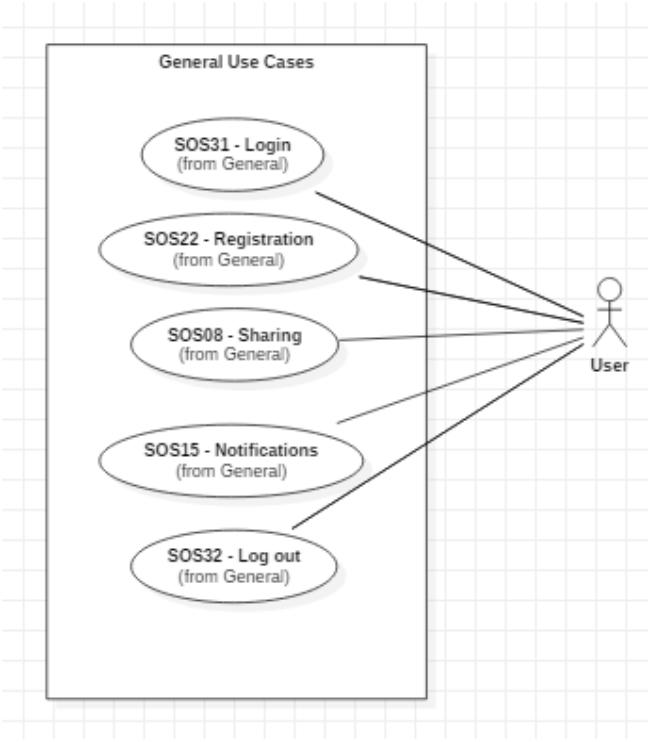

Figure 2: General Use Cases, all of which represent use cases that affects all users of the system.

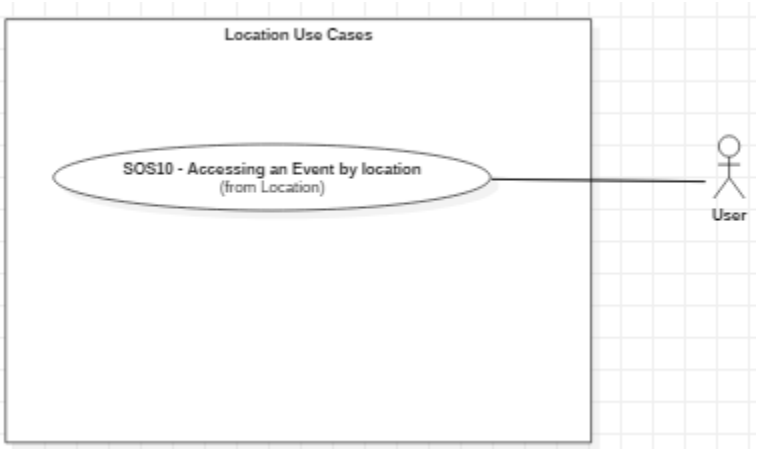

Figure 3: Location Use Case Diagram, which collects the use cases having to do with the external geolocation API.

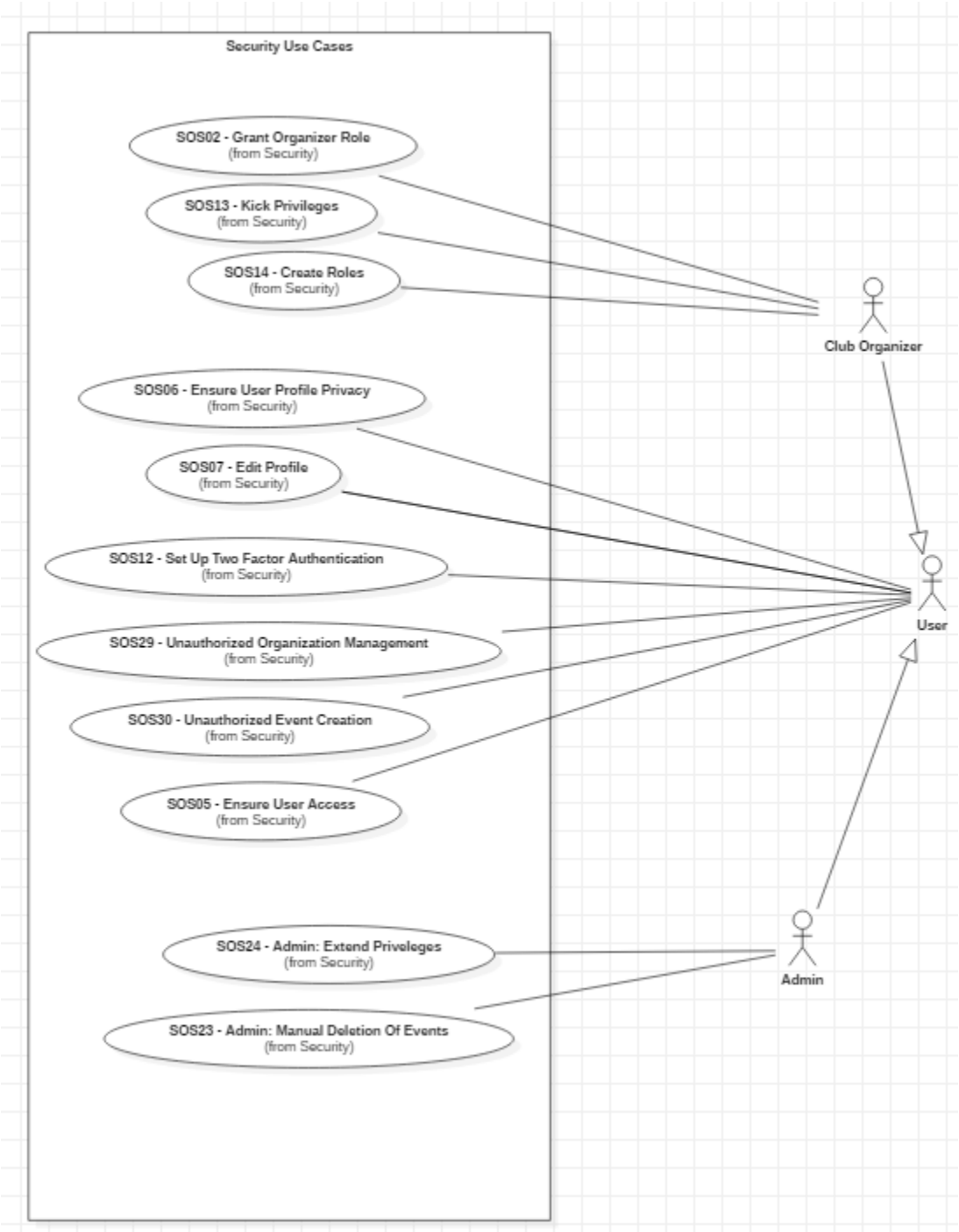

Figure 4: Security Use Cases, which collects all the use cases having to do with security.

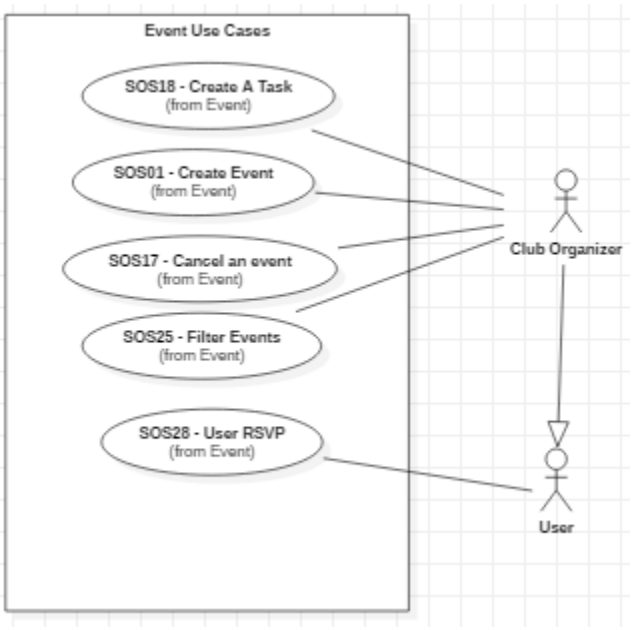

Figure 5: Event Use Case Diagram, which collects all the use cases related to event creation, destruction, etc.

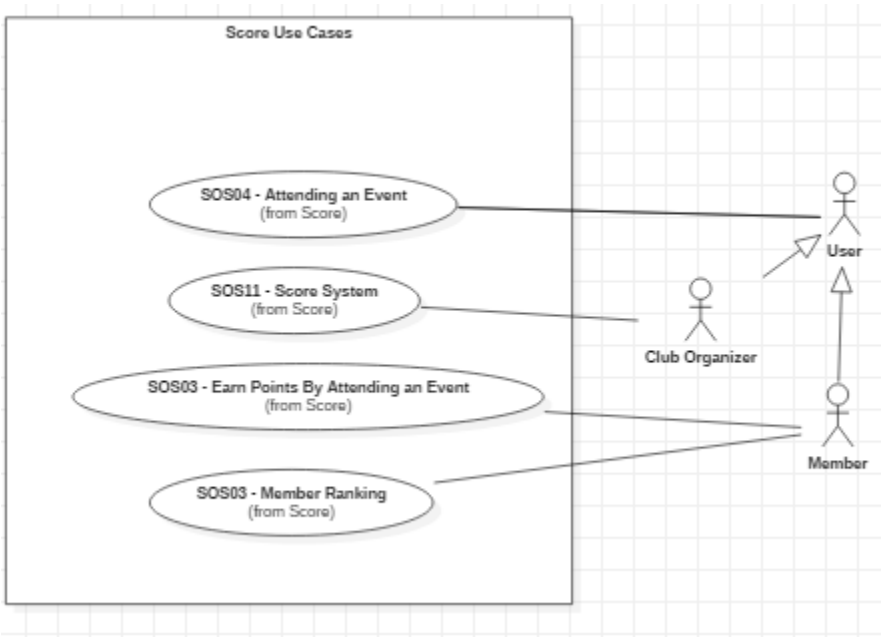

Figure 6: Score Use Case Diagram, which collects all the use cases having to do with the ranking and point system.

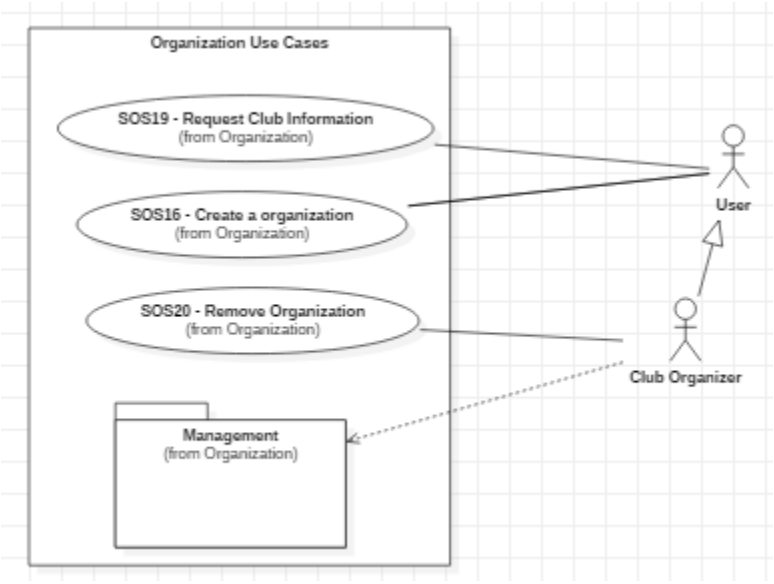

Figure 7: Organization Use Case Diagram, which collects all the use cases having to do with organizations.

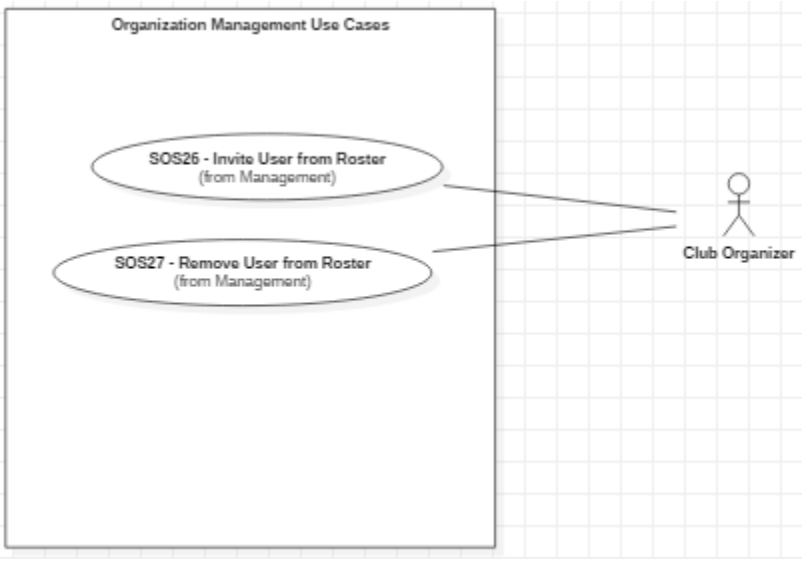

Figure 8: Organization Management Use Cases, which collects all the use cases having to do with organization management and is a subset of the Organization Use Cases.

4.2.2 Implemented Use Case Diagram

The following Use Cases are implemented:

- SOS01 – Create Event
- SOS02 – Grant Organizer Role
- SOS04 – Attending an Event
- SOS07 – Edit Profile
- SOS10 – Accessing an Event by Location
- SOS16 – Create an Organization
- SOS17 – Cancel an Event
- SOS22 – Registration
- SOS31 – Login
- SOS32 – Log Out

The Use Case diagram describing this subset is shown in Figure 9.

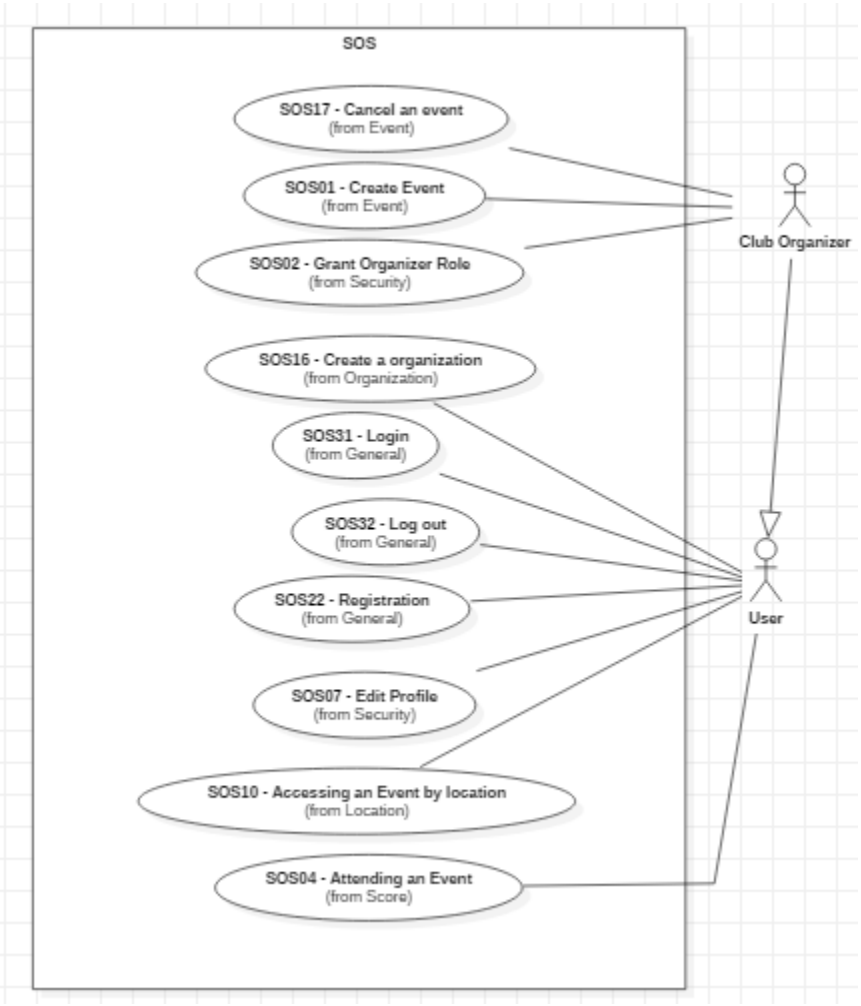

Figure 9: Use Case diagram for the implemented Use Cases.

5 Requirements Analysis

The following sections contain an Analysis model of the SOS system. Herein the expectations of the users for the system are defined. Section 5.1 contains scenarios instantiating each one of the implemented use cases. Each one of these scenarios has a corresponding object diagram and sequence diagram in Section 5.2. The object diagrams, together with a single class diagram, complete the static model of the system.

5.1 Scenarios

Each of the following scenarios instantiates a corresponding use case from Section 4.1. The scenarios are further used in Section 5.2 to generate object diagrams for the static model. The more specific view of the system helps capture the actions and attributes of classes and objects that the SOS system must handle.

5.1.1 Scenario: SOS16 - Creating an organization

A user named John Doe is logged into the system. John is on the organization page. John is part of no clubs but wants to create his own club. John attempts to create a club with the following parameters:

- Organization Name: “The Doe Crew”
- Organization Description: “A bunch of people excited for fishing.”
- Requirement for joining: “Anyone with an interest in fishing can join.”
- Privacy of the organization: PUBLIC

He submits his request and the page refreshes and shows him the newly created club with the parameters he provided.

5.1.2 Scenario: SOS04 - Attending an event

A user named Patricio Clarke is logged into the system. Patricio is on the events page. Patricio is part of the organization “The Doe Crew” and the system provides a list of events that he has signed up for. The system shows one event called “Deep sea fishing.” Patricio is currently at the event and he decides to let the system know that he is at the event by clicking “I’m here.” After clicking on the button Patricio receives a message claiming that his attendance has been noted.

5.1.3 Scenario: SOS14 - Registration

A user named John Day wants to join the SOS community. He enters the link to the log in page of the website and he clicks register and enters the following information:

- User Name = jday005
- Email = jday005@fiu.edu
- Password = abc123
- Confirm Password = abc123

John clicks on register and the site then takes a couple of seconds to verify his registration. John is then sent to the home page.

5.1.4 Scenario: SOS17 - Cancel an Event

An organizer named Juan Ciervo with event manager privileges is logged into the system and is in the event page. Juan sees all the events his organization is hosting which include “Hiking the Himalayas” and “Skydiving.” Juan Ciervo wants to cancel an the “Skydiving” event for his organization “The Ciervo Squad.” Juan clicks on view event description and then he clicks on cancel event. Juan is certain that the organization cannot host this event, so he confirms the cancellation request. Juan then sees that the event is no longer visible therefore it has been cancelled.

5.1.5 Scenario: SOS31 – Log in

A user named Nica Perez is not logged into the system but has an account with the SOS. Nica is currently on the log in screen of the website and she puts in the following information into the log in form:

- Email: npere203@fiu.edu
- Password: **********

The system allows her to log in as her information is correct. She ends the scenario at the homepage.

5.1.6 Scenario: SOS21 - Create Event

An organizer Mohammad Lee is logged into the system and he has event manager privileges. Mohammad is currently in the administration page of the organization he is a part of, “The Lees.” Mohammad wants to create an event, so he clicks on the create event button. A form appears and Mohammad inputs the following parameters:

- Event name: “Uphill biking”
- Event date and time: “10/05/2019 12:00:00 PM”
- Event location: “12345 SW 678 TER Kyoto, Japan 910112”
- Event Type: NORMAL
- Event Visibility: VISIBLE
- Event Duration: 1 hour

Mohammad publishes the new event he created, and the event is created successfully.

5.1.7 Scenario: SOS02 - Grant Organizer Role

An organizer Juana Cierva is logged into the system and she has the role manager privilege. Juana is part of the organization “The Ciervo Squad.” Juana is currently on the organization management view. Juana want to grant a role to another member of “The Ciervo Squad,” Bob Swanson. Using the invitation menu Juana enters the following data:

- Member Name: “Bob Swanson”
- Organizer Title: “Moderator”
- Powers and Privileges: KICK

Juana submits the request to grant the role to Bob. Bob is then notified, and Juana now sees that Bob owns the role of Moderator within “The Ciervo Squad.”

5.1.8 Scenario: SOS07 - Edit Profile

A user Janet Lake is logged into the system and wants to change her account privacy. Janet clicks on the edit profile button and she sees the current profile settings she has on the page and is able to edit the following fields:

- Email: jlake009@fiu.edu
- Phone Number: (123) – 456-7890
- Privacy: Public
- Date Of Birth: 01/01/1990

Janet changes her privacy to private and then submits the data. The page refreshes and shows a confirmation message displaying that her account is now private.

5.1.9 Scenario: SOS32 – Log out

A user Bob Hope wants to log out of the system. He presses log out from the current page that he is in and he successfully logs out of the system.

5.1.10 Scenario: SOS10 - Access Events by Location

A user Lolly Tates is logged into the system. Lolly is in the event page. Lolly allows the webpage to access her location (12345 SE 342 Ter 33029). After confirming, Lolly sees an event, "Volleyball tournament" occurring near her location.

5.2 Static Model

This section contains the static view of the system described through UML object and class diagrams. In the static model, each entity (both external and internal) of the system is represented in terms of classes with attributes and operations, and their dependencies and associations. Two views are provided: the instantiated object diagrams, which are collected in Section 5.2.1, and the more general class diagram, which appears in Section 5.2.2.

5.2.1 Object Diagrams

Each one of the UML object diagrams in this section is a subsection of the class diagram (in Section 5.2.2) which represents an instance of a Use Case (which can be seen in Section 4.1). The corresponding scenarios that the object diagrams are based on are in Section 5.1.

5.2.1.1 Object Diagram for Scenario: SOS16 – Create an Organization

The corresponding scenario for the object diagram in Figure 10 is in Section 5.1.1, which instantiates the Use Case in Section 4.1.2.16.

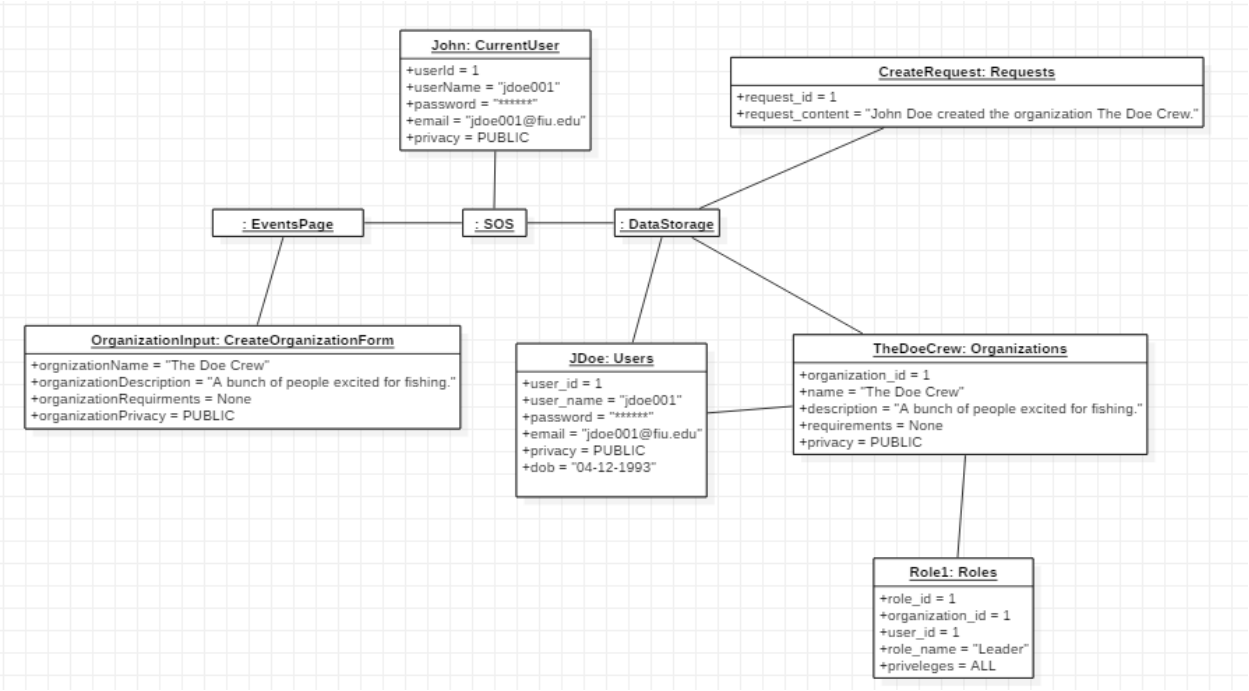

Figure 10: Object Diagram for Scenario: SOS16 - Create an Organization

5.2.1.2 Object Diagram for Scenario: SOS04 – Attending an Event

The corresponding scenario for the object diagram in Figure 11 is in Section 5.1.2, which instantiates the Use Case in Section 4.1.2.4.

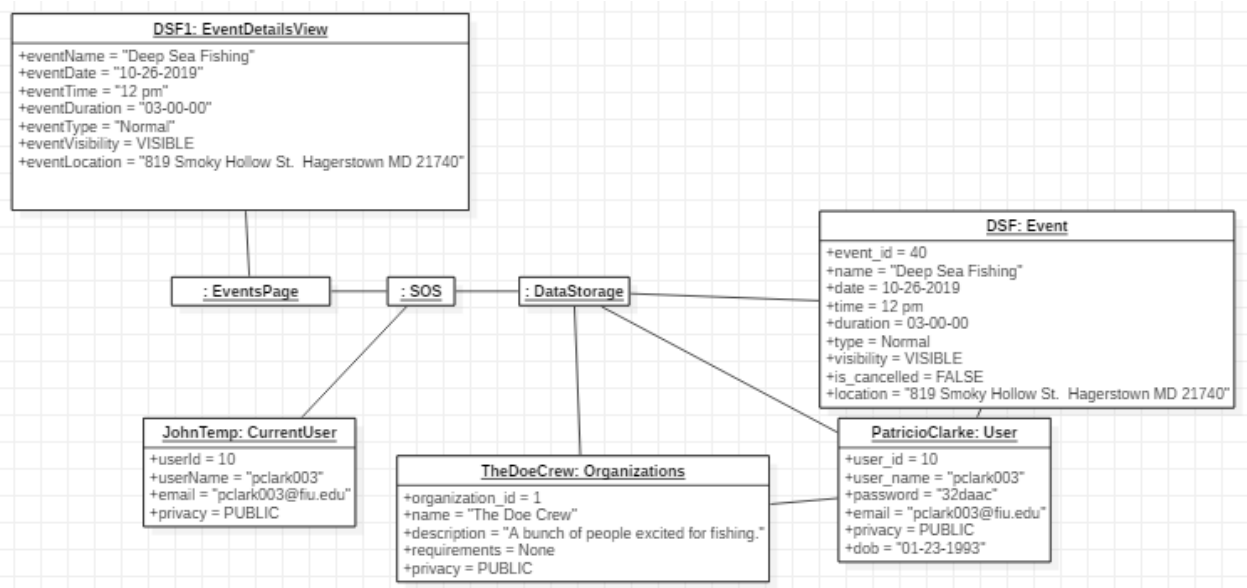

Figure 11: Object Diagram for Scenario: SOS04 - Attending an Event

5.2.1.3 Object Diagram for Scenario: SOS22 – Registration

The corresponding scenario for the object diagram in Figure 12 is in Section 5.1.3, which instantiates the Use Case in Section 4.1.2.22.

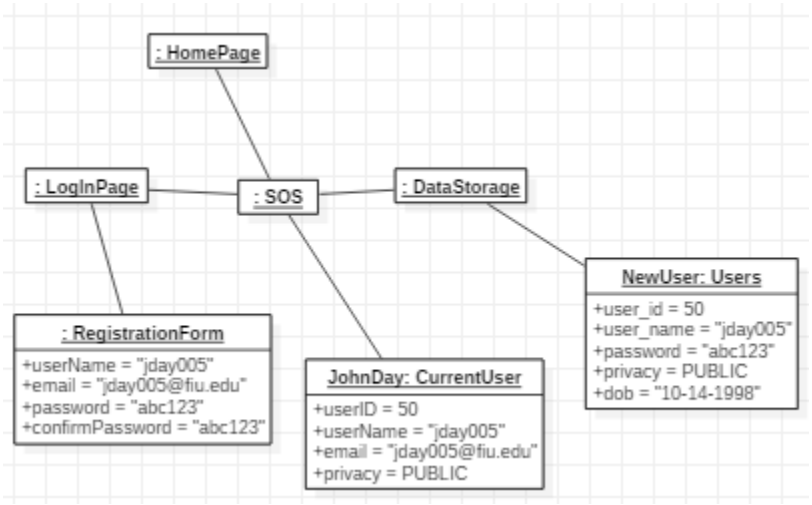

Figure 12: Object Diagram for Scenario: SOS22 – Registration

5.2.1.4 Object Diagram for Scenario: SOS17 – Cancel an Event

The corresponding scenario for the object diagram in Figure 13 is in Section 5.1.4, which instantiates the Use Case in Section 4.1.2.17.

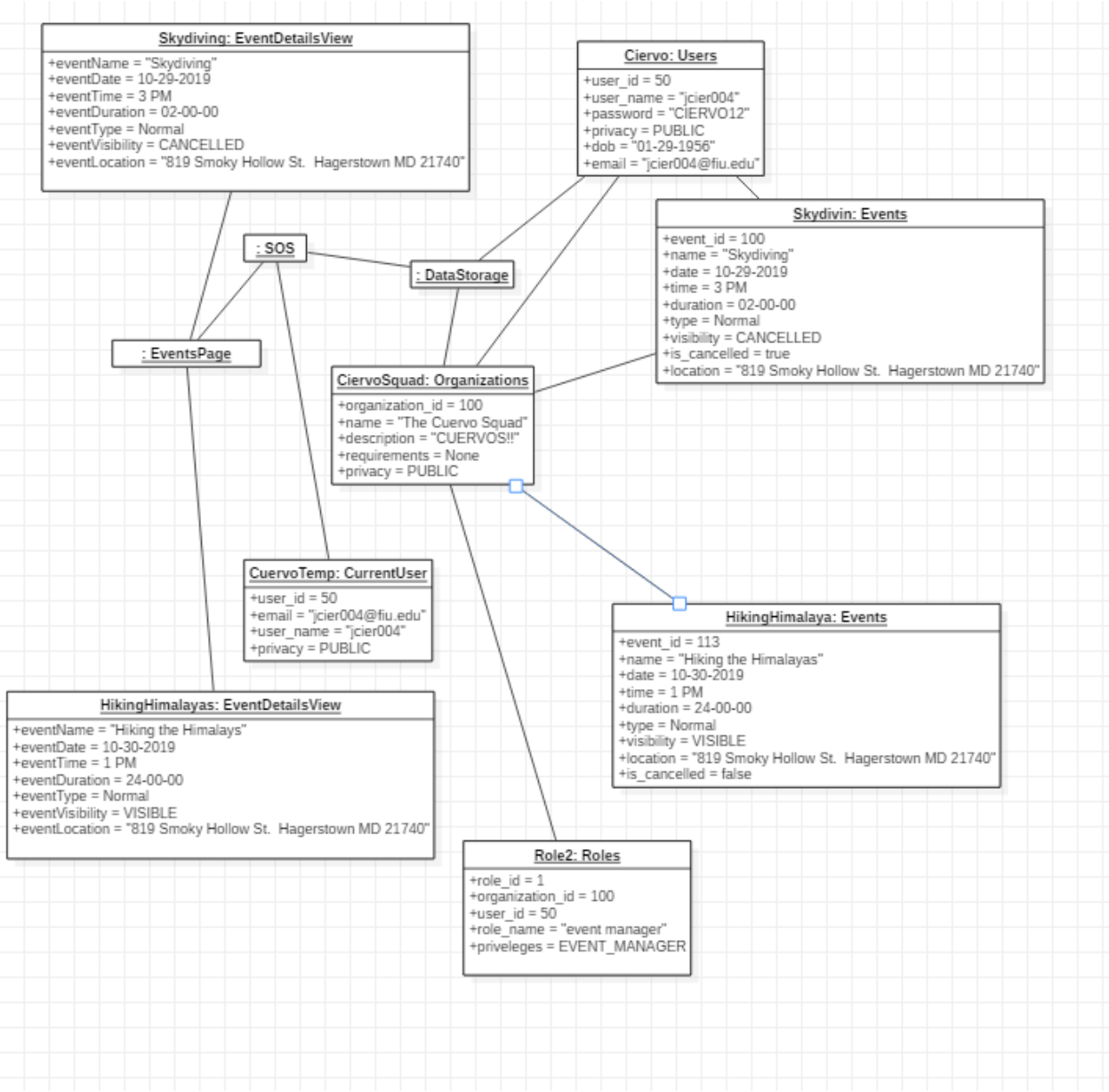

Figure 13: Object Diagram for Scenario: SOS17 – Cancel an Event

5.2.1.5 Object Diagram for Scenario: SOS31 – Log in

The corresponding scenario for the object diagram in Figure 14 is in Section 5.1.5, which instantiates the Use Case in Section 4.1.2.31.

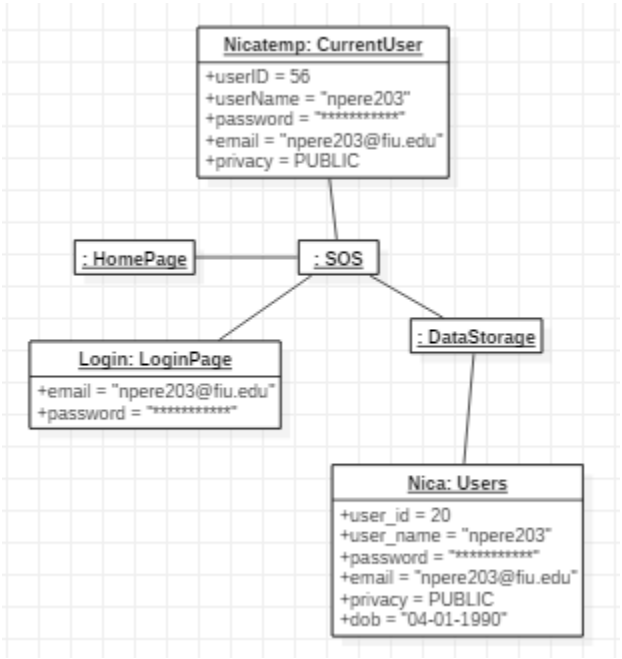

Figure 14: Object Diagram for Scenario: SOS31 – Log in

5.2.1.6 Object Diagram for Scenario: SOS01 – Create Event

The corresponding scenario for the object diagram in Figure 15 is in Section 5.1.6, which instantiates the Use Case in Section 4.1.2.1.

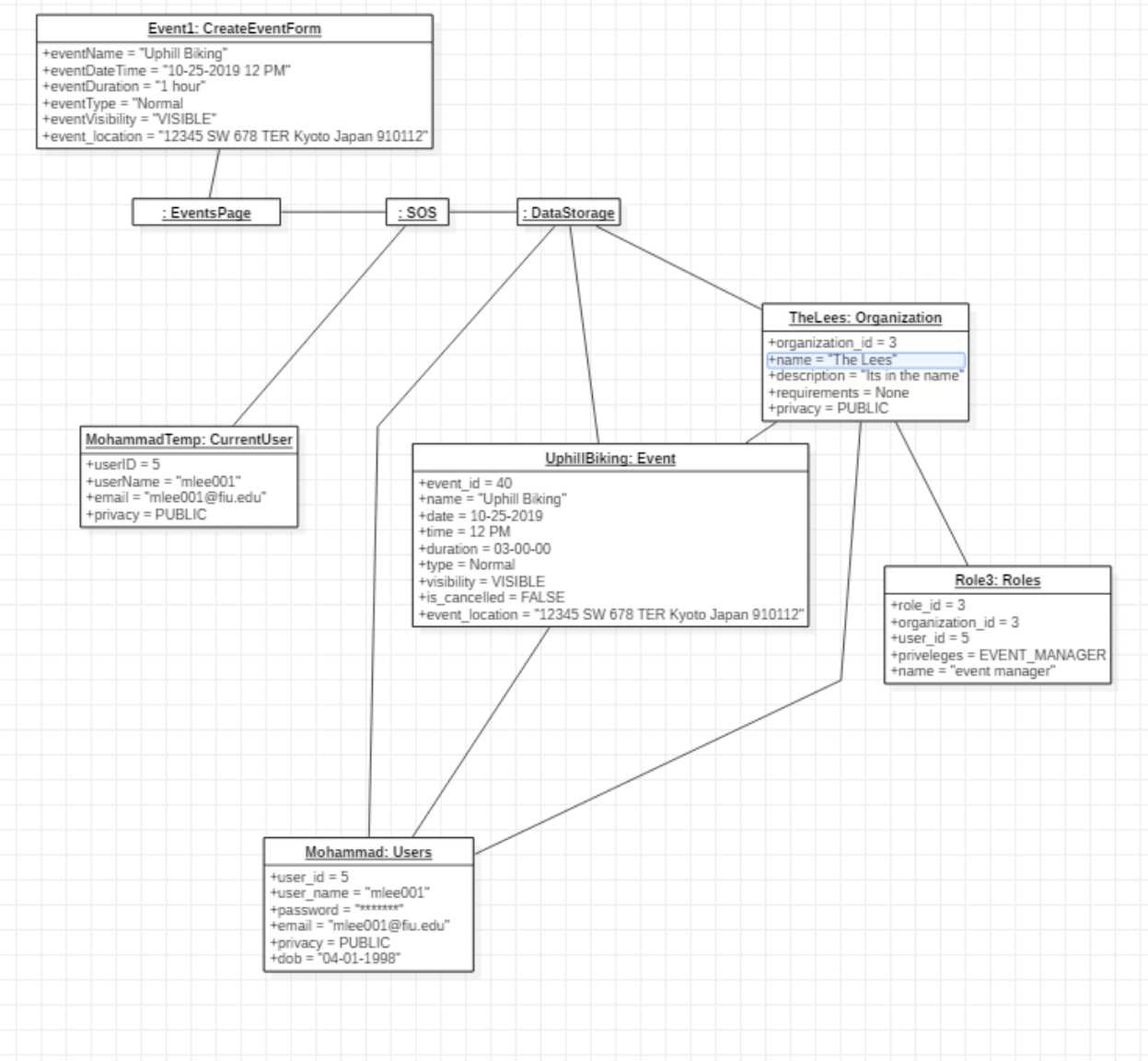

Figure 15: Object Diagram for Scenario: SOS01 – Create Event

5.2.1.7 Object Diagram for Scenario: SOS02 – Grant Organizer Role

The corresponding scenario for the object diagram in Figure 16 is in Section 5.1.7, which instantiates the Use Case in Section 4.1.2.2.

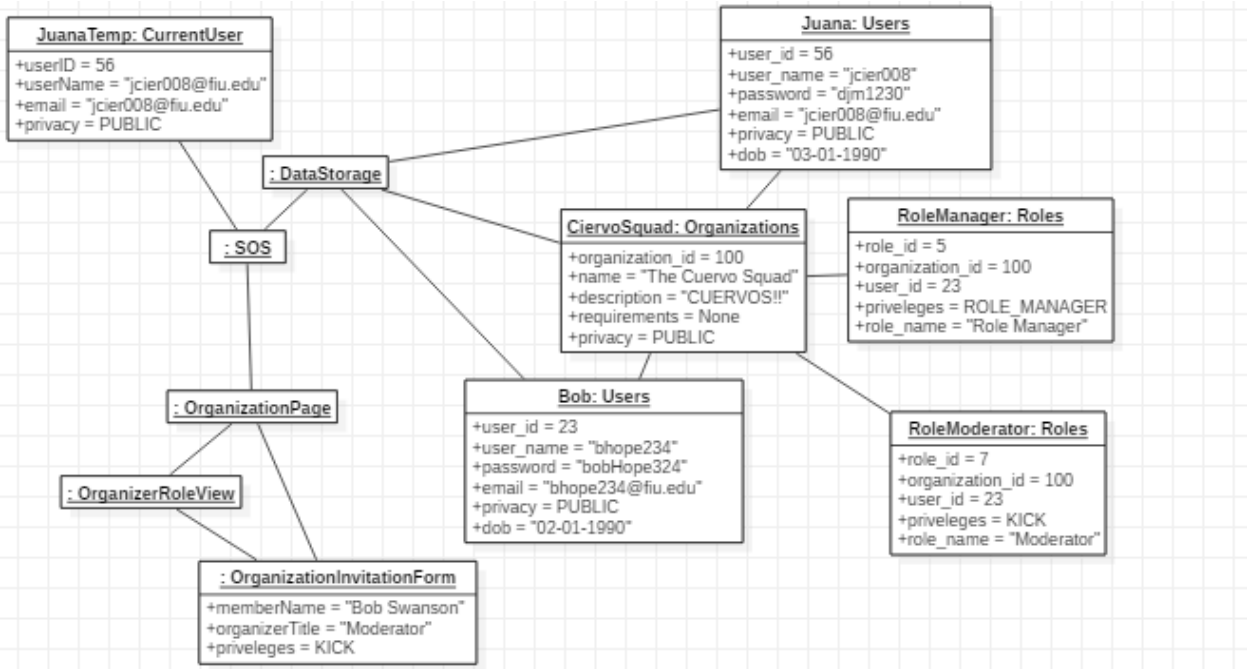

Figure 16: Object Diagram for Scenario: SOS02 – Grant Organizer Role

5.2.1.8 Object Diagram for Scenario: SOS07 – Edit Profile

The corresponding scenario for the object diagram in Figure 17 is in Section 5.1.8, which instantiates the Use Case in Section 4.1.2.7.

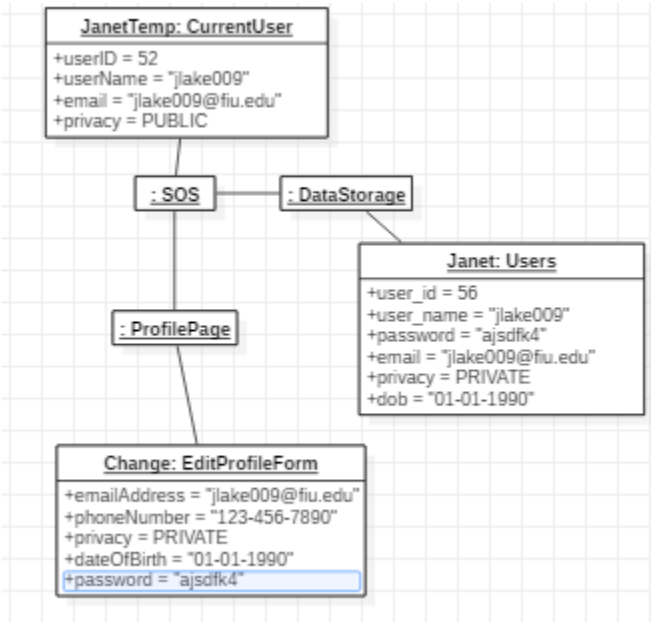

Figure 17: Object Diagram for Scenario: SOS07 – Edit Profile

5.2.1.9 Object Diagram for Scenario: SOS32 – Log out

The corresponding scenario for the object diagram in Figure 18Figure 12 is in Section 5.1.9, which instantiates the Use Case in Section 4.1.2.32.

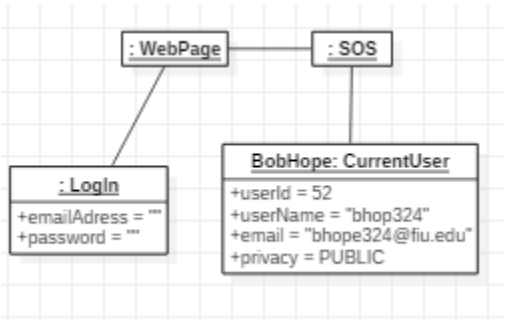

Figure 18: Object Diagram for Scenario: SOS32 – Log out

5.2.1.10 Object Diagram for Scenario: SOS10 – Access Events by Location

The corresponding scenario for the object diagram in Figure 19 is in Section 5.1.10, which instantiates the Use Case in Section 4.1.2.10.

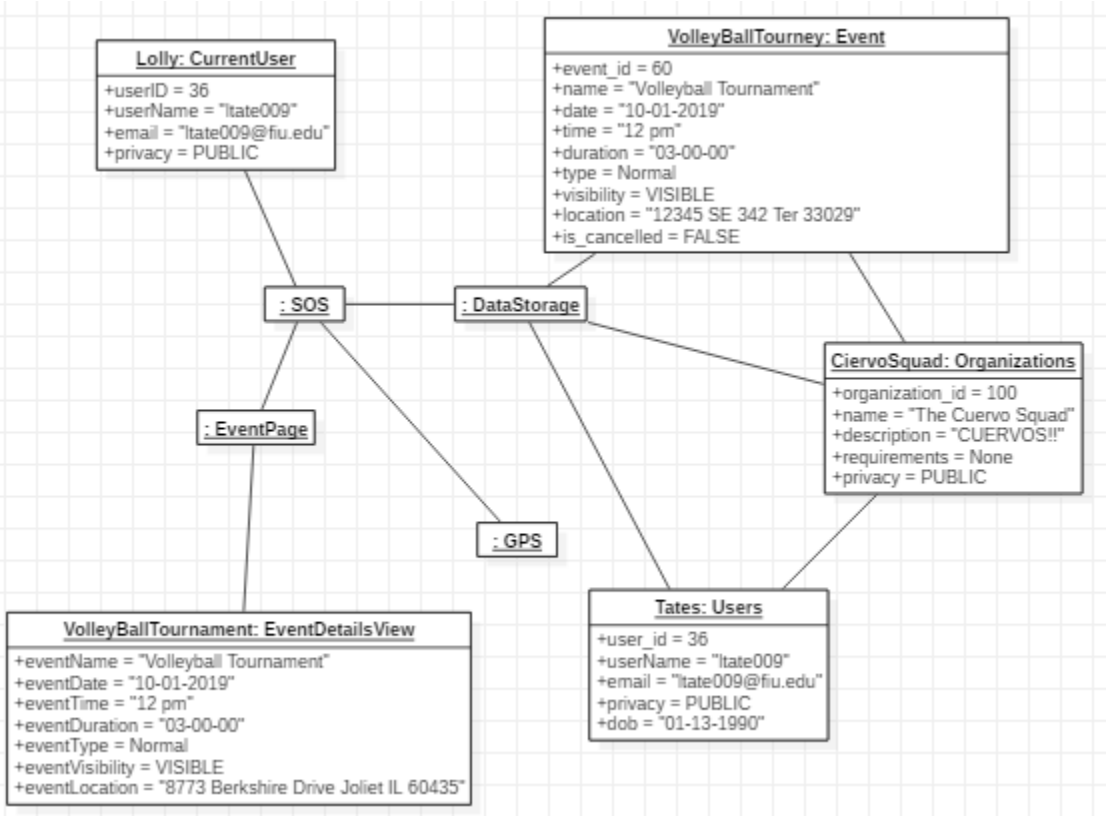

Figure 19: Object Diagram for Scenario: SOS14 – Registration

5.2.2 Class Diagram

The class diagram for the system is depicted in Figure 20.

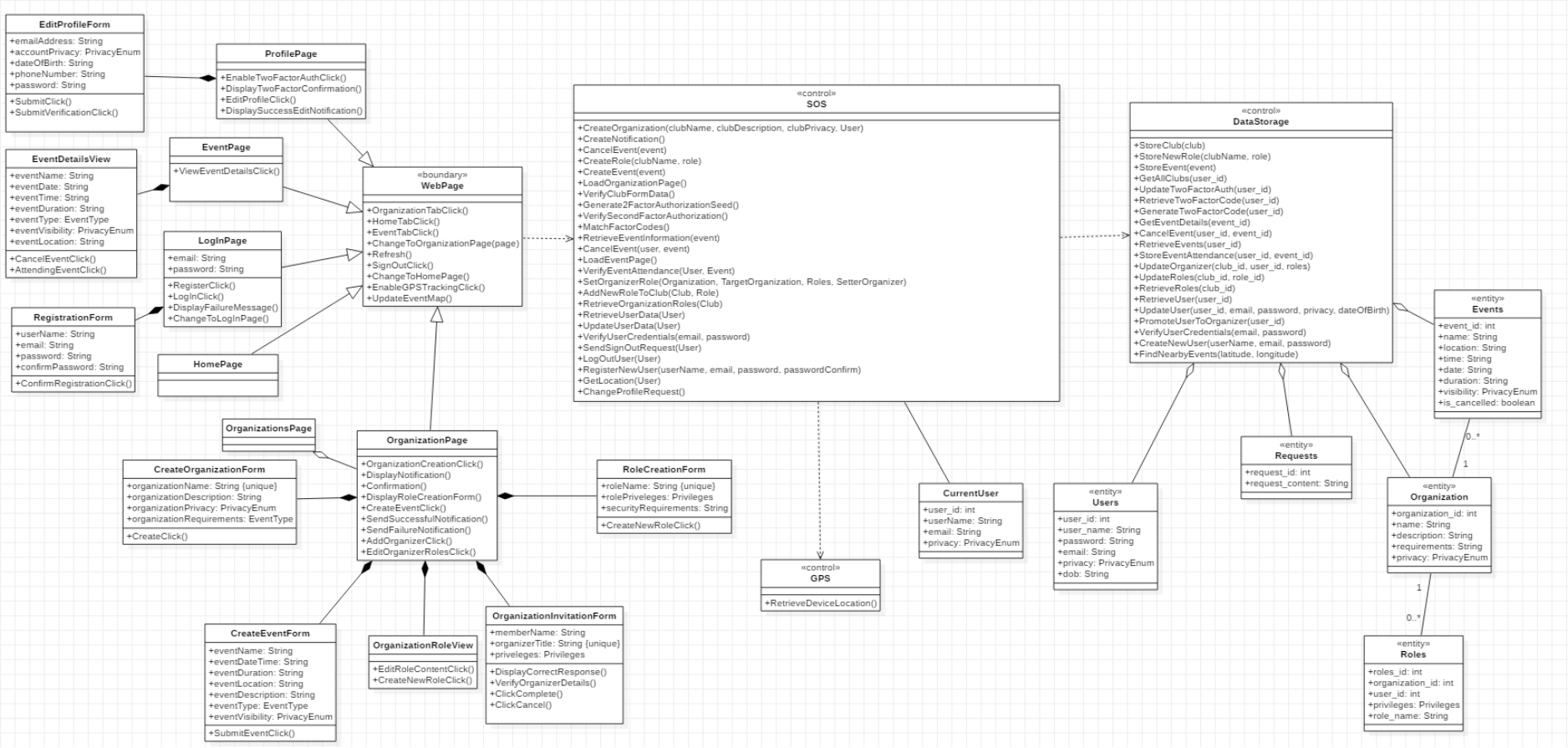

Figure 20: Class Diagram for the Implemented Use Cases

5.3 Dynamic Model

Each one of the following sections contains a sequence diagram representing their associated Use Case.

5.3.1 Sequence Diagram for SOS16 – Create an Organization

This sequence diagram in Figure 21 corresponds to the Use Case in Section 4.1.2.16. The sequence diagram displays the events fired by the system when the user clocks on the organization tab and the create button on the create organization form.

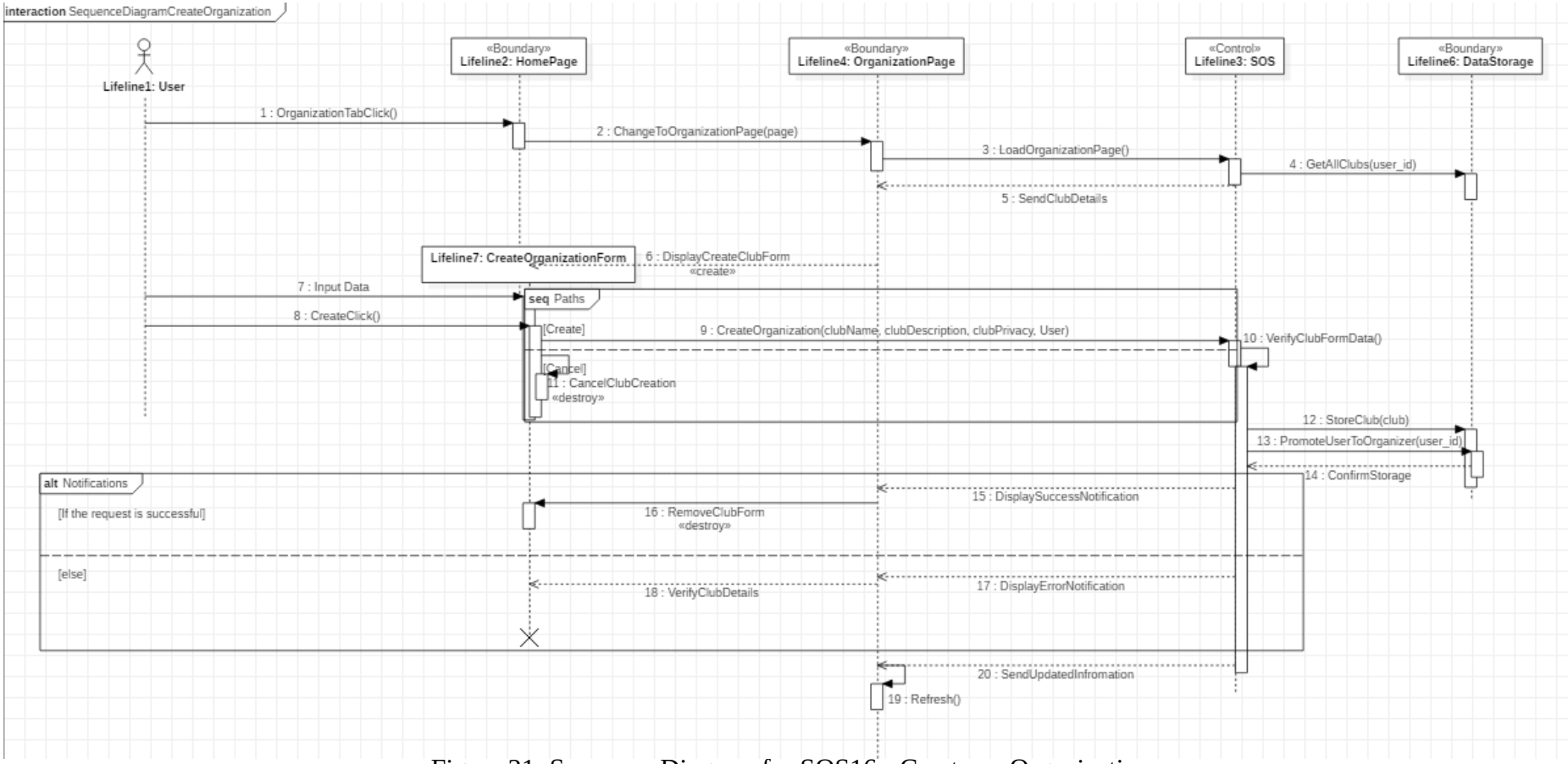

Figure 21: Sequence Diagram for SOS16 - Create an Organization

5.3.2 Sequence Diagram for SOS01 – Create an Event

This sequence diagram in Figure 22 corresponds to the Use Case in Section 4.1.2.1. The sequence diagram displays the events fired after the user clicks on the create event click and submits the request to create the event.

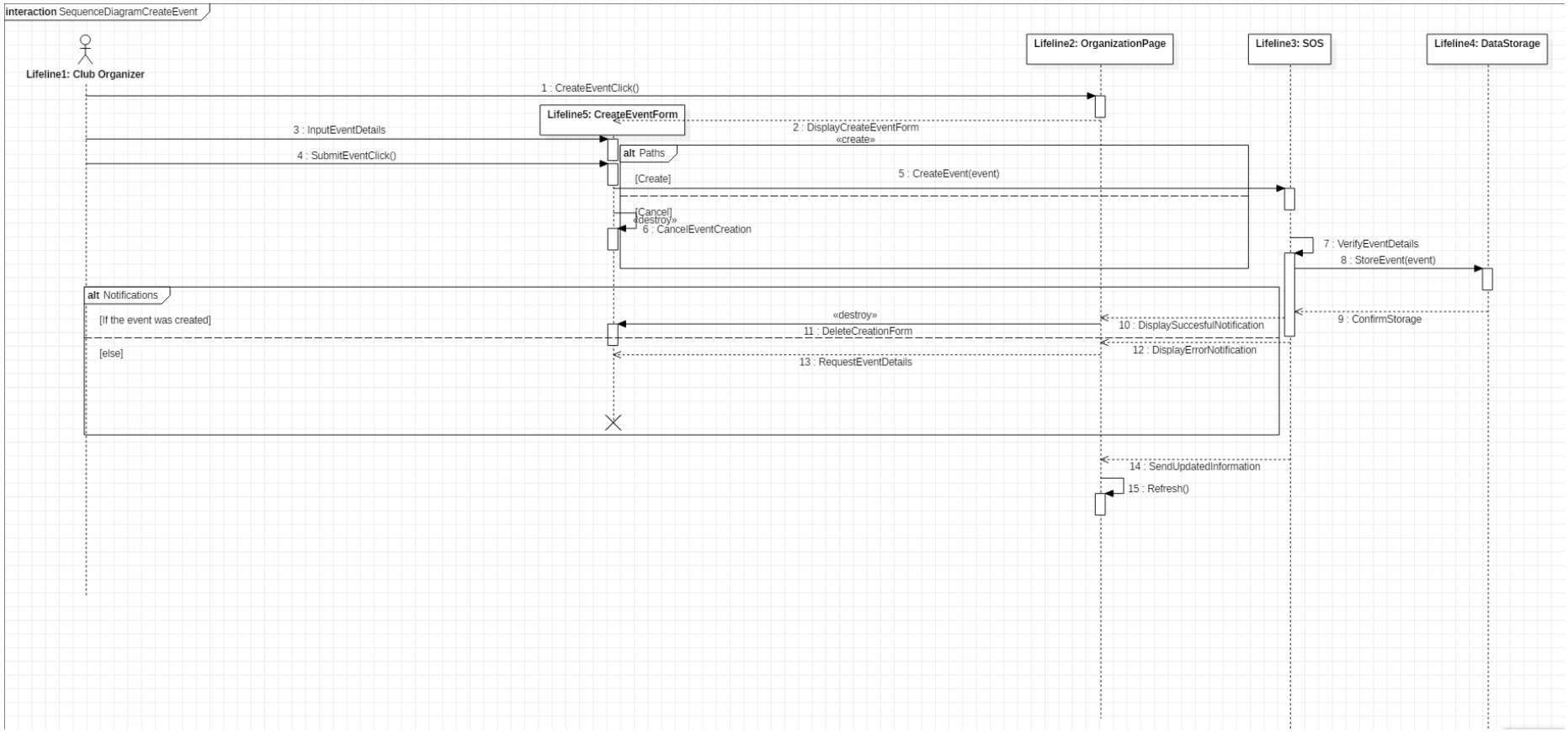

Figure 22: Sequence Diagram for SOS01 - Create Event

5.3.3 Sequence Diagram for SOS17 – Cancel an Event

This sequence diagram in Figure 23 corresponds to the Use Case in Section 4.1.2.17. This sequence diagram shows the events fired after the organizer clicks on view event details in the event page and the cancel event click on the event details view.

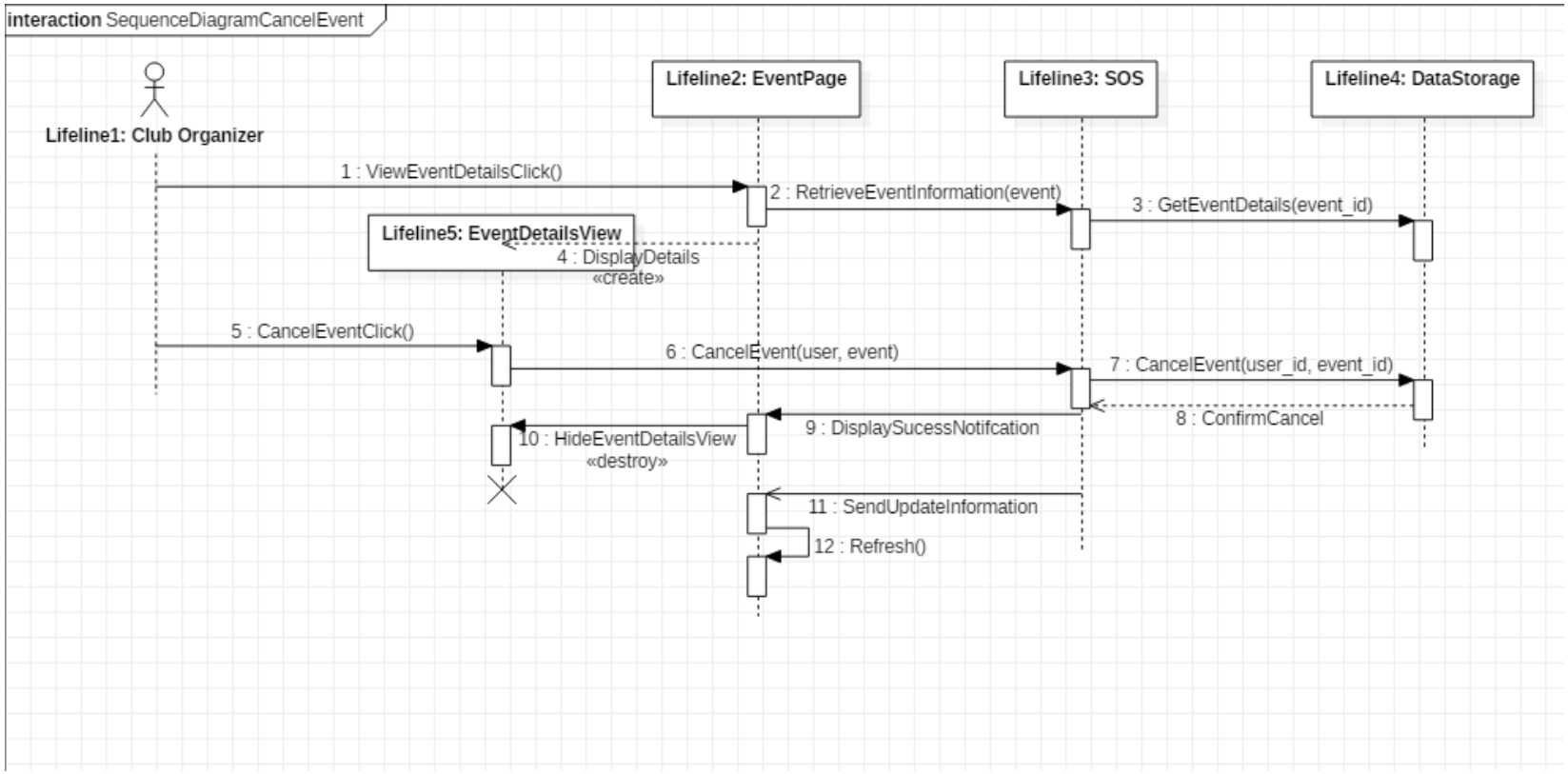

Figure 23: Sequence Diagram for SOS17 - Cancel an Event

5.3.4 Sequence Diagram for SOS04 – Attending an Event

This sequence diagram in Figure 24 corresponds to the Use Case in Section 4.1.2.4. This sequence diagram shows the operation performed after the member clicks on view event details and then submits a request to attend the event.

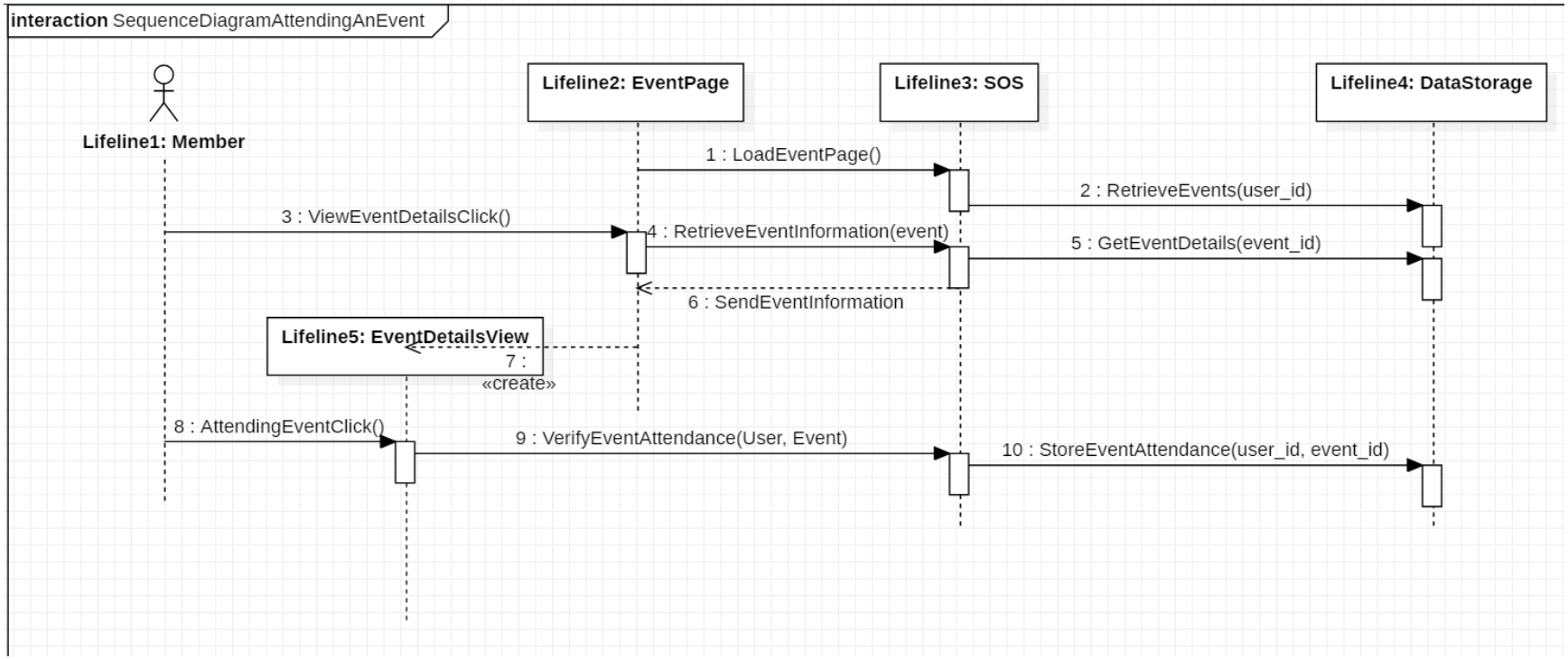

Figure 24: Sequence Diagram for SOS04 - Attending an Event

5.3.5 Sequence Diagram for SOS02 – Grant Organizer Role

This sequence diagram in Figure 25 corresponds to the Use Case in Section 4.1.2.2. The sequence diagram illustrates the actions the system will take after the user requests to add an organizer and either submits the request or cancels the granting of the role.

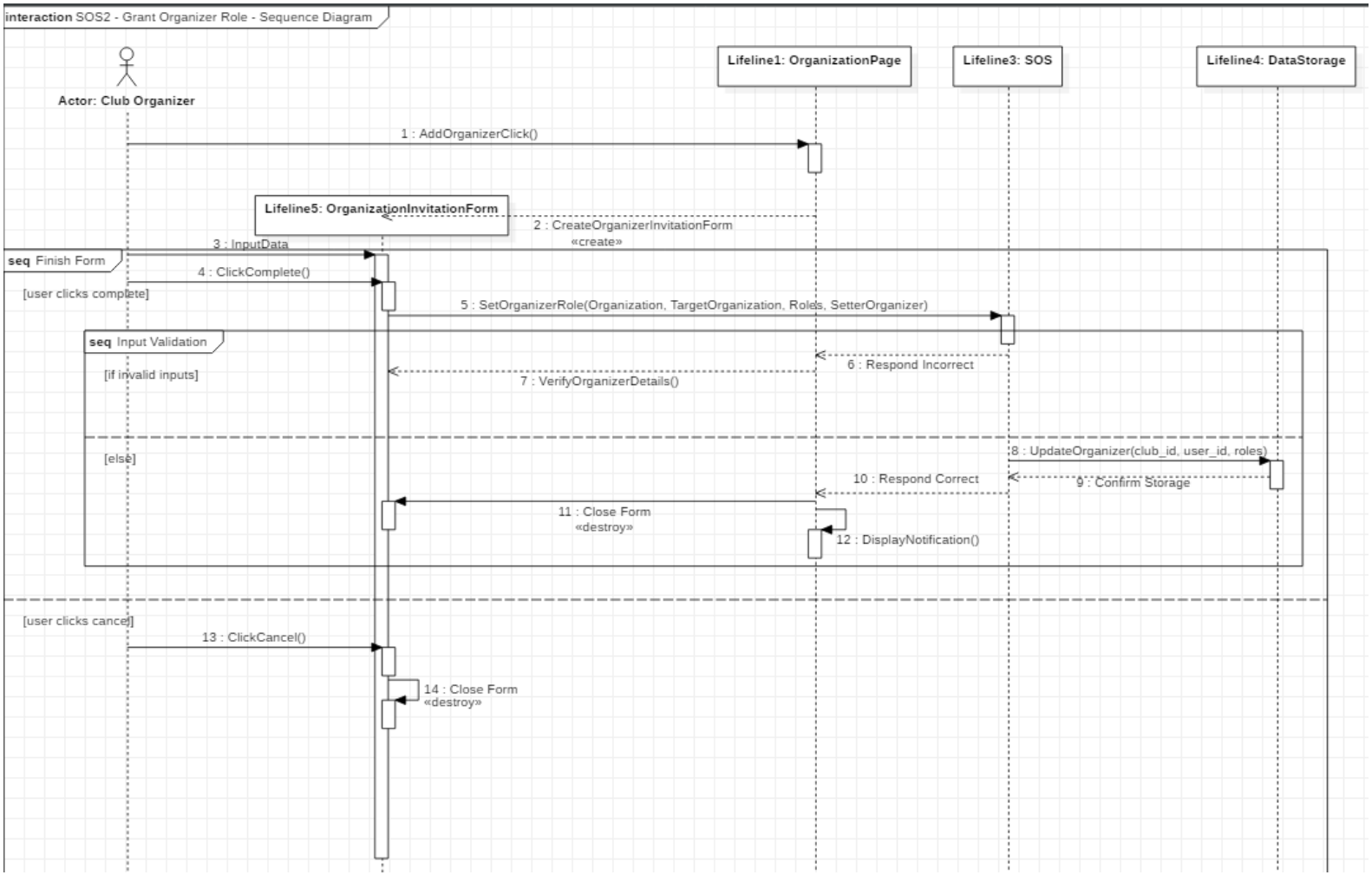

Figure 25: Sequence Diagram for SOS02 - Grant Organizer Role

5.3.6 Sequence Diagram for SOS07 – Edit Profile

This sequence diagram in Figure 26 corresponds to the Use Case in Section 4.1.2.7. This sequence diagram highlights the events occurring when a user clicks on edit profile, submits the edits and submit their password for authentication.

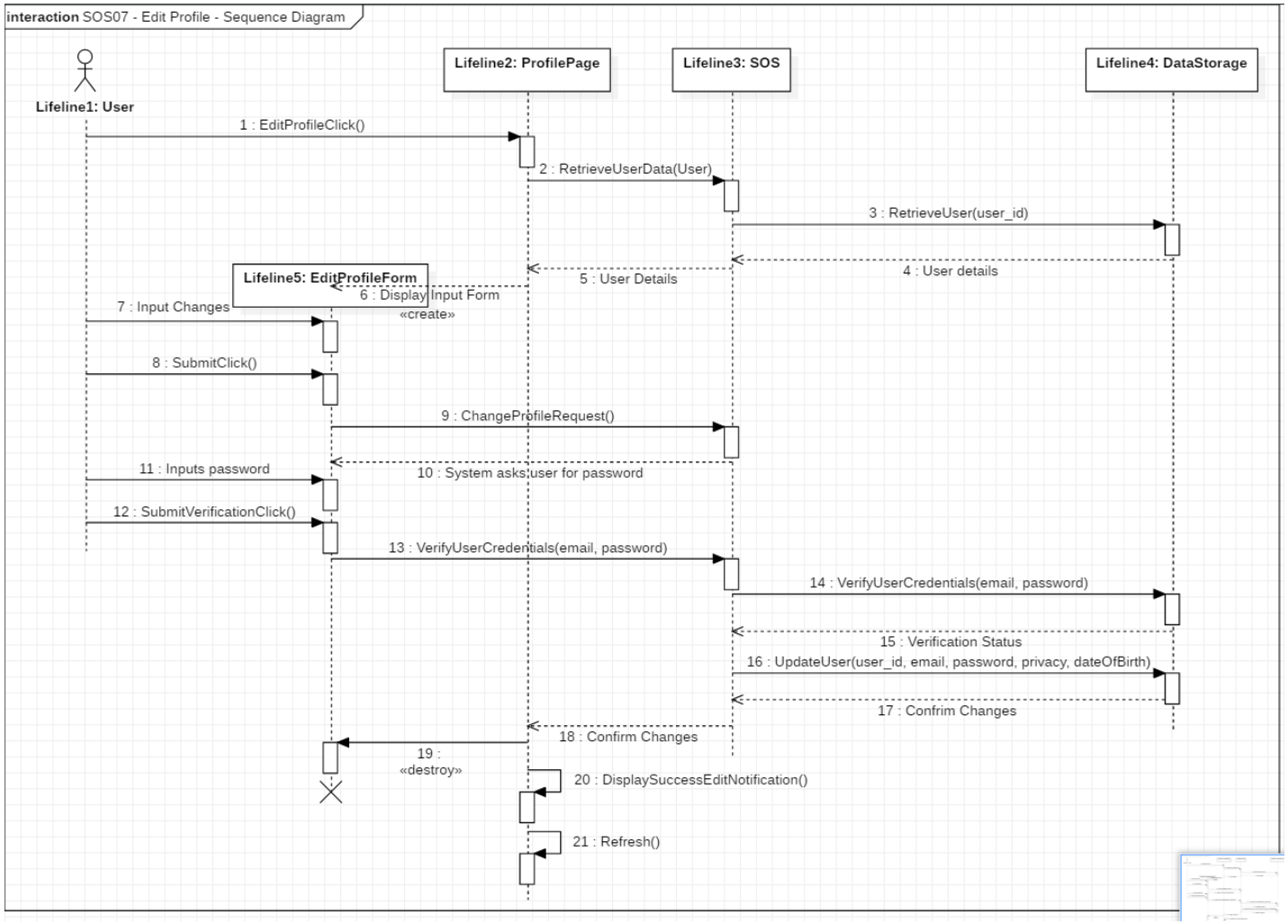

Figure 26: Sequence Diagram for SOS07 - Edit Profile

5.3.7 Sequence Diagram for SOS10 – Accessing an Event by Location

This sequence diagram in Figure 27 corresponds to the Use Case in Section 4.1.2.10. This sequence diagram shows the events fired after the user allows the system to track their location.

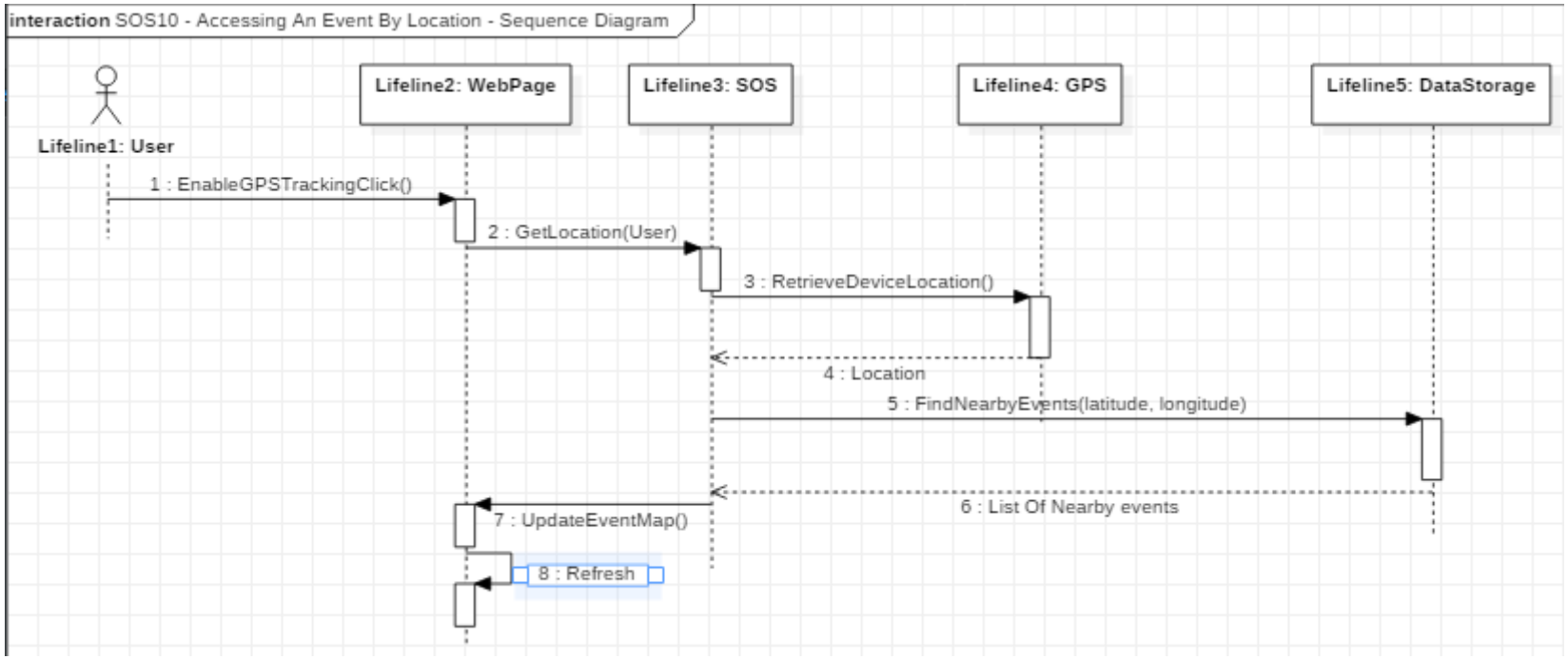

Figure 27: Sequence Diagram for SOS10 - Accessing an Event by Location

5.3.8 Sequence Diagram for SOS22 – Registration

This sequence diagram in Figure 28 corresponds to the Use Case in Section 4.1.2.22. The sequence diagram shows the actions taken by the system after the user clicks to register and confirms their registration.

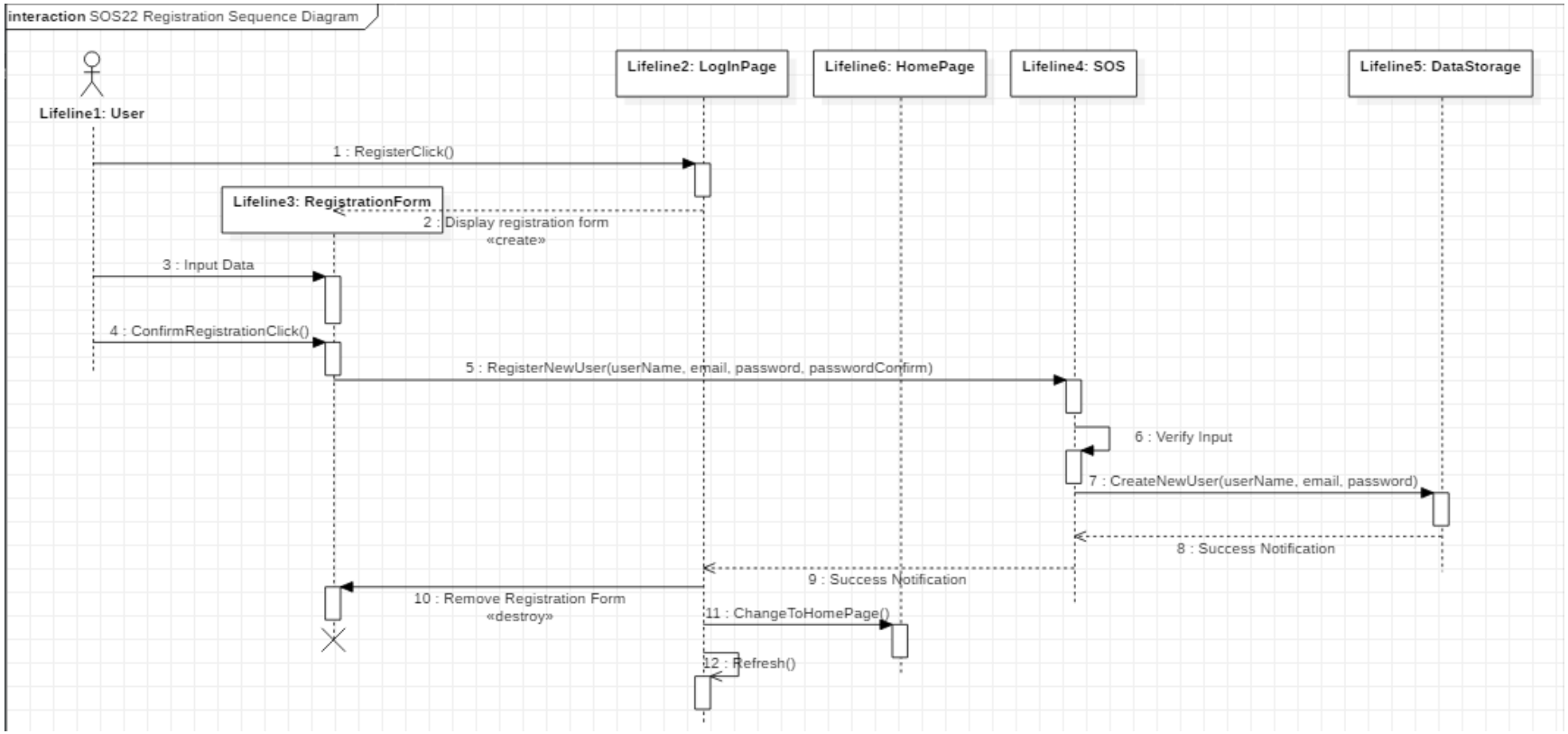

Figure 28: Sequence Diagram for SOS22 - Registration

5.3.9 Sequence Diagram for SOS32 – Log Out

This sequence diagram in Figure 29 corresponds to the Use Case in Section 4.1.2.32. The sequence diagram shows the processes occurring after the user clicks on sign out from any page in the site.

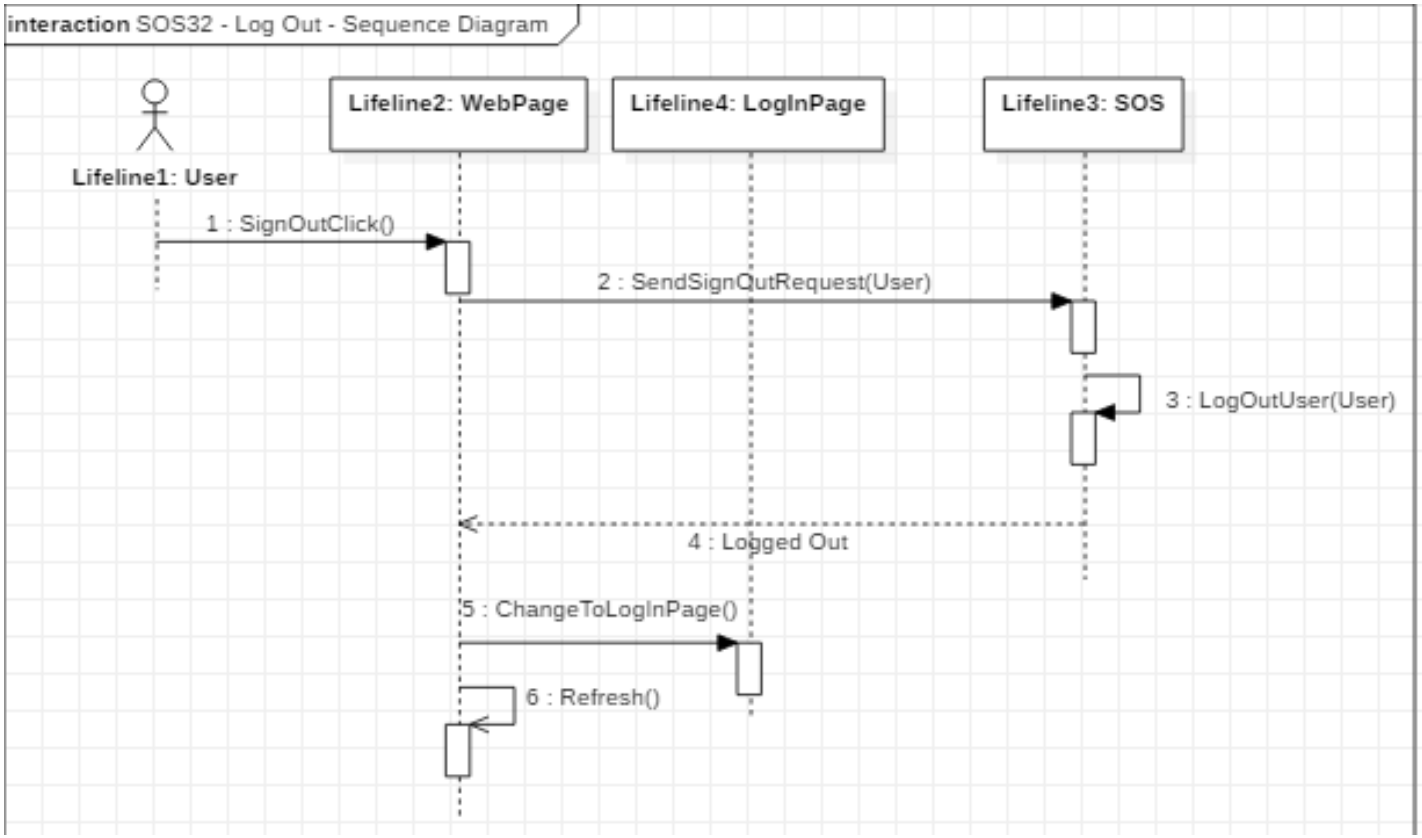

Figure 29: Sequence Diagram for SOS32 - Log Out

5.3.10 Sequence Diagram for SOS31 – Log In

This sequence diagram in Figure 30 corresponds to the Use Case in Section 4.1.16. The sequence diagram highlights the events occuring after the user enters their credentials and clicks to log into the system.

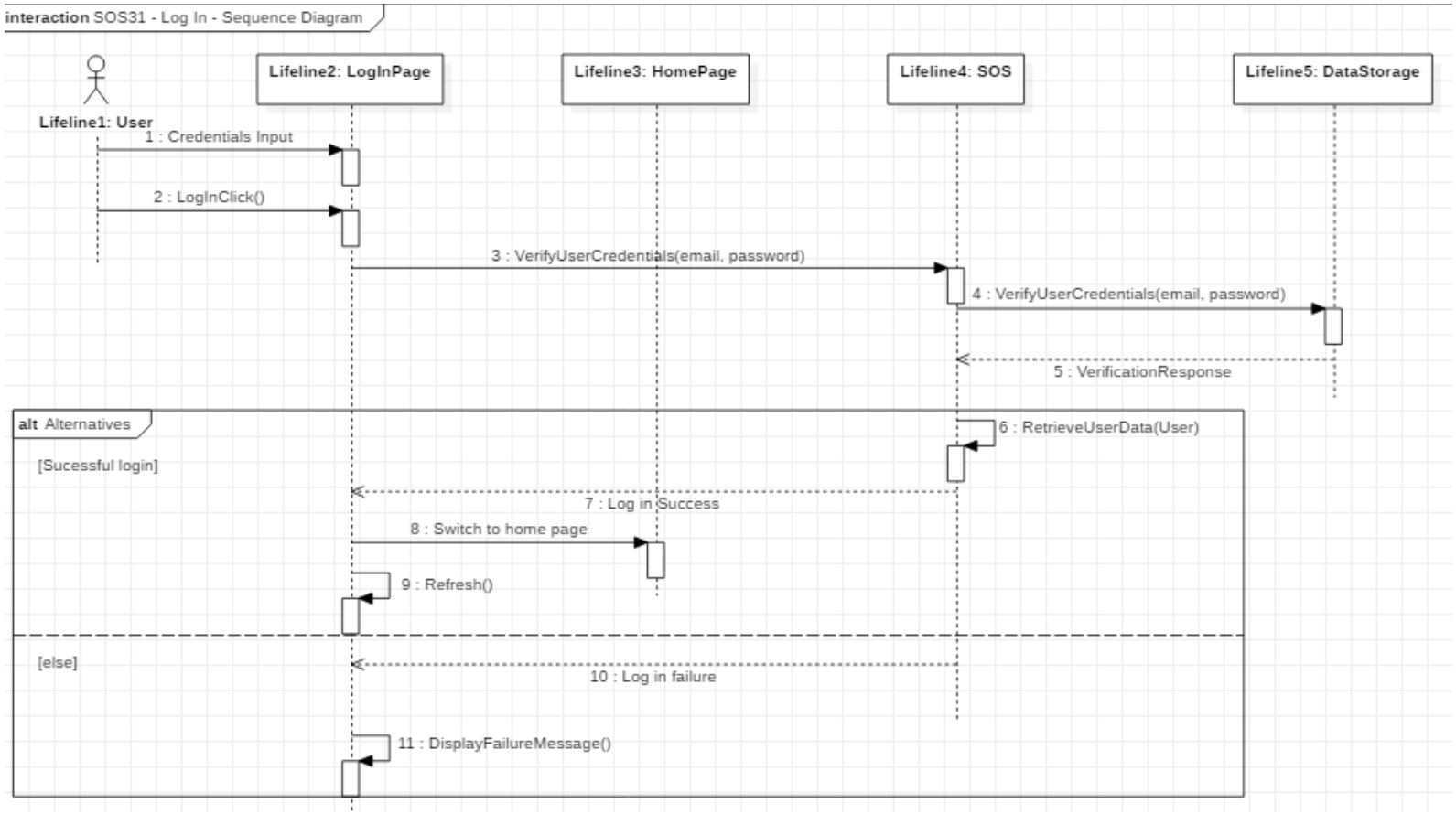

Figure 30: Sequence Diagram for SOS31 - Log In

6 Glossary

- **Scenario**, a scene that illustrates some interactions of the proposed system.
- **Static Model**, a model which does not depend on elements of time.
- **Dynamic Model**, a model which depends on or contains elements of time, especially allowing interactions between entities over time.
- **Gantt Chart,** a bar chart where the x-axis is time and the y-axis is the different tasks, and the duration of each task is represented by the length of a bar.
- **Unified Software Development Model**, …
- **Sequence Diagram,** an interaction diagram which focus on the time-ordering of messages and interactions.
- **Use Case Diagram,** a diagram that shows a set of use cases and actors; and their relations.
- **SOS,** Student Organization System.
- **Object Diagram,** a diagram that models the instances of things contained in a class diagram, i.e., a set of objects and their relationships at a point in time.
- **Class Diagram,** a UML diagram containing a representation
- **Attribute,** a variable on a UML class.
- **Operation,** a function on a UML class indicating an action.
- **Role,** a set of technical and managerial tasks that are expected from a participant or a team.
- **Activity,** a set of tasks performed towards a specific purpose.
- **Task,** an atomic unit of work that can be managed and that consumes resources.
- **Milestone,** end-point of a software process activity.
- **Deliverable,** a work product for the client.
- **Notation,** a graphical or textual set of rules representing a model.
- **Method,** a repeatable technique for solving a specific problem.
- **Methodology,** a collection of methods for solving a class of problems.
- **Use Case,** a sequence of events describing all possible actions between actors and the system for a given piece of functionality.
- **Actors,** the roles interacting with the system such as end-users and other computer systems.

7 Approval Page:

Approval Page of System Requirements Document of

Student Organization System

Member Signatures

Armando J. Ochoa	10/01/2019
Member Signature	Date
Yovanni Jones	10/01/2019
Member Signature	Date
M.Kian Maroofi	10/01/2019
Member Signature	Date
Teriq Douglas	10/01/2019
Member Signature	Date
Anthony Sanchez-Ayra	10/01/2019
Member Signature	Date

8 References

Campus Lab. (2019). *Panther Connect*. Retrieved from Panther Connect: https://fiu.campuslabs.com/engage/

Jacobson, I., Booch, G., & Rumbaugh, J. (1999). *The Unified Software Development Process.* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

9 Appendices

9.1 Appendix A – Project Schedule

The scheduled tasks are contained in Section 3.3. The Gantt chart of the schedule is below, in Figure 31

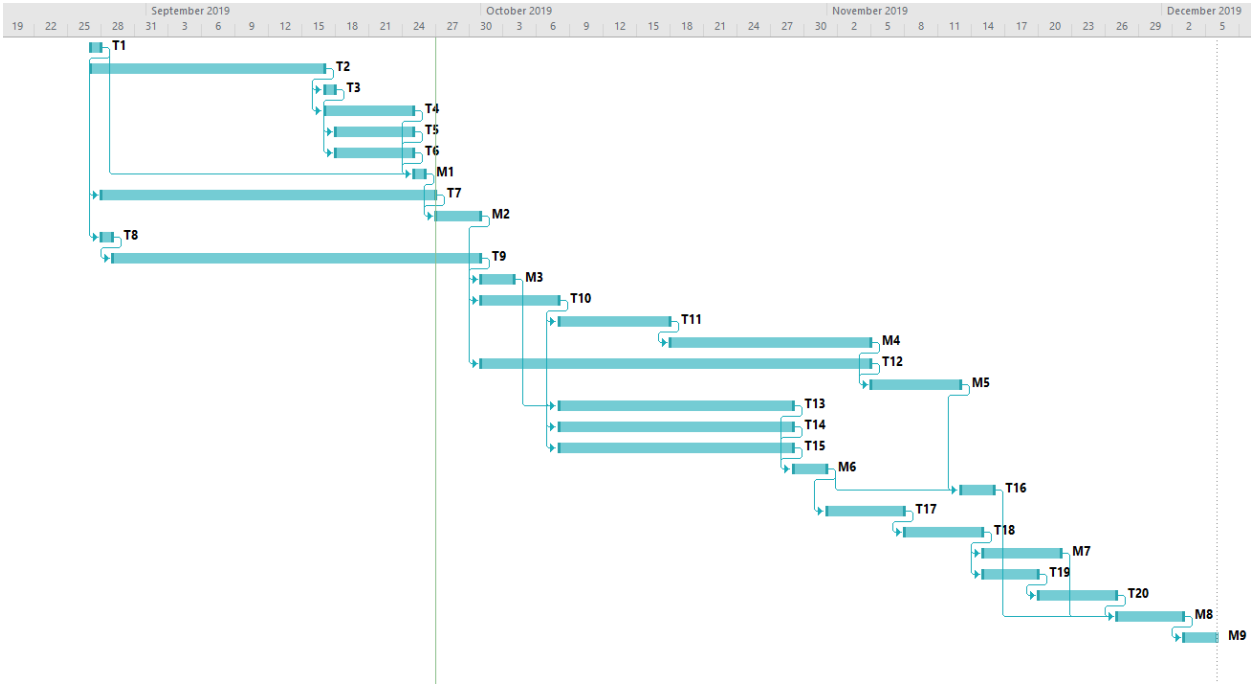

Figure 31: Gantt chart for the full project schedule.

9.2 Appendix B – User Interface Design

This section contains still from the SOS prototype describing the UI layout.

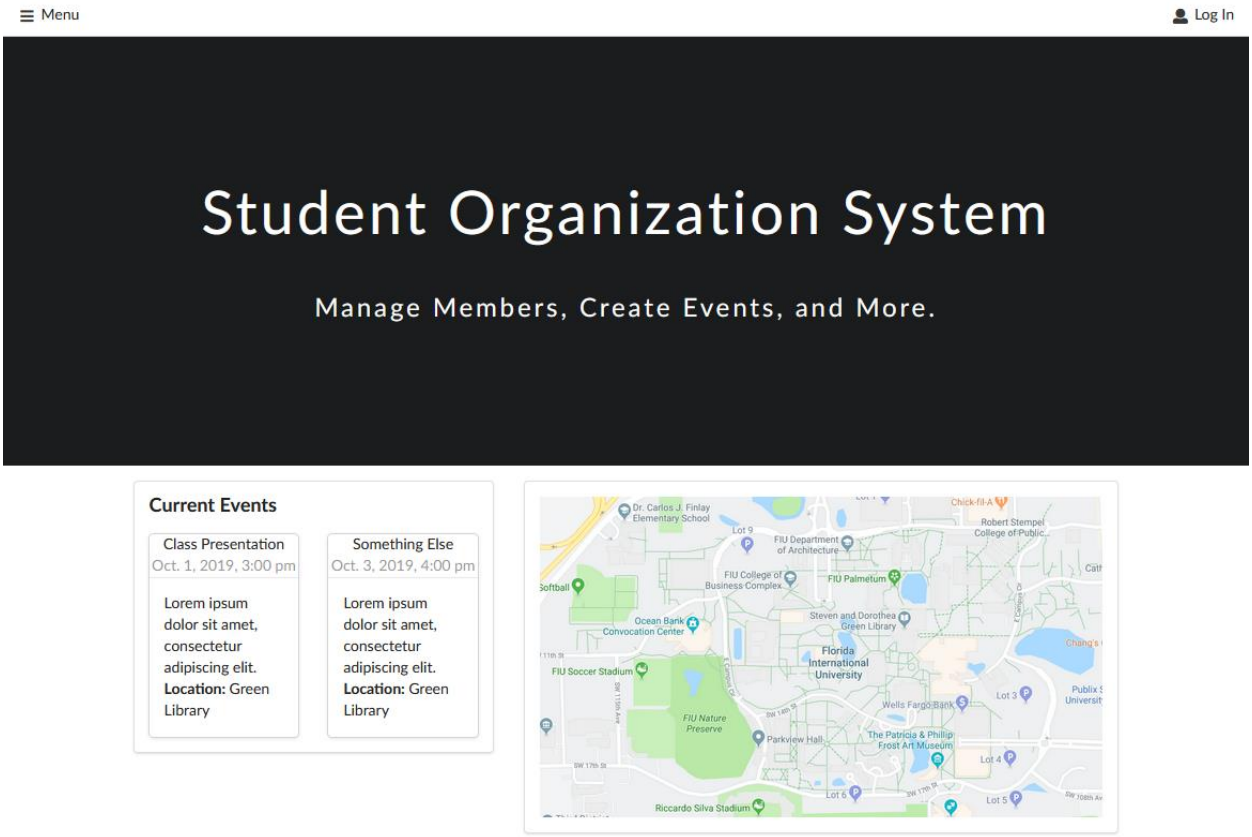

Figure 32: UI Layout for the Homepage. It shows the top-bar, which is a static element of the while system which contains a button to open the navigation menu, and another one to open a log-in pop-up.

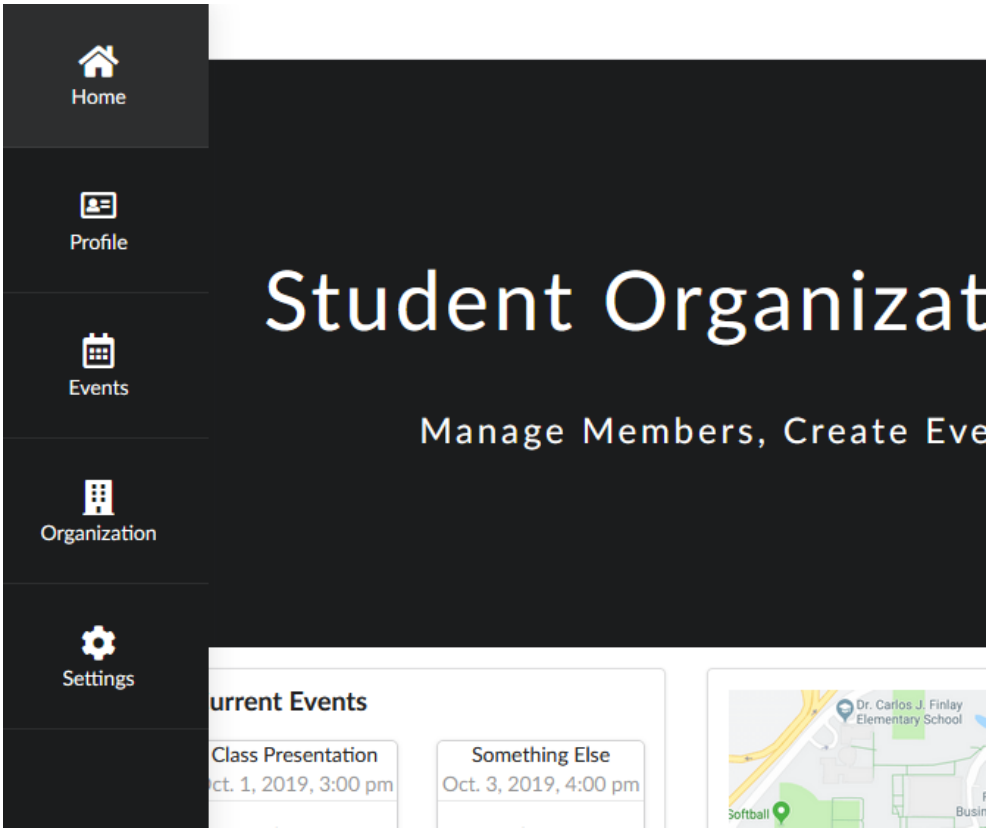

Figure 33: UI Layout for the menu sidebar (with the Homepage on the background). Each button redirects to a specific section on the website that the user can access.

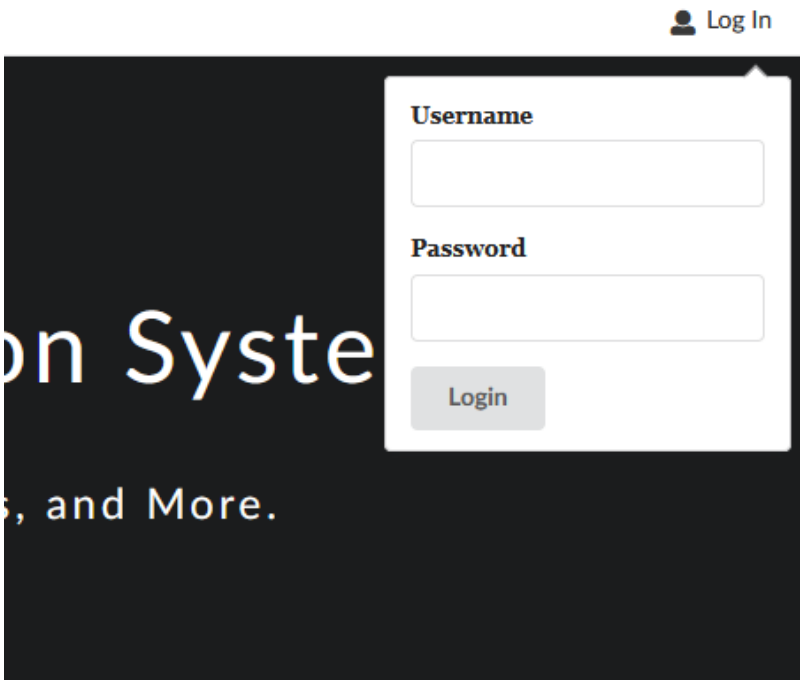

Figure 34: UI Layout for the Log In pop-up (with the Homepage on the background).

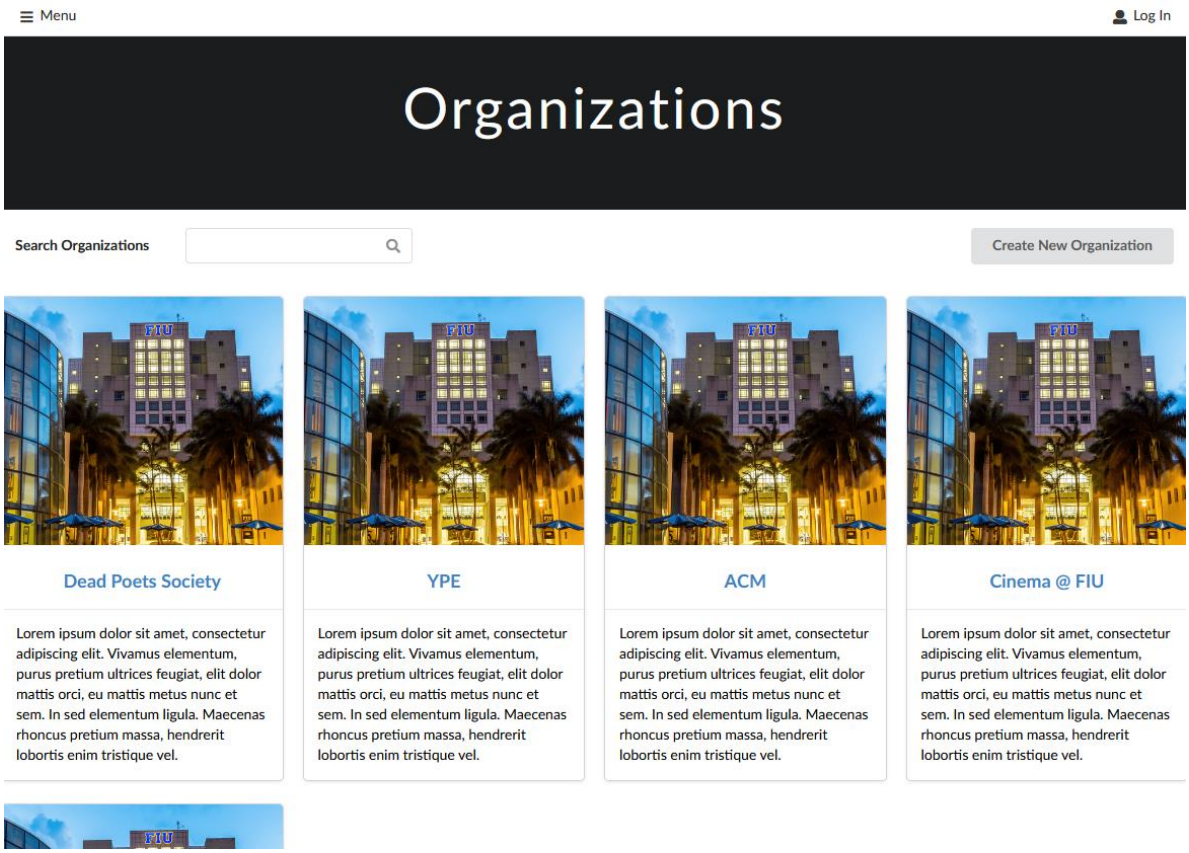

Figure 35: UI layout for the Organizations page. Each organization in the system that is displayed to the user is displayed as at-a-glance card. The create organization button triggers a modal/form (see Figure 36) to define a new Organization.

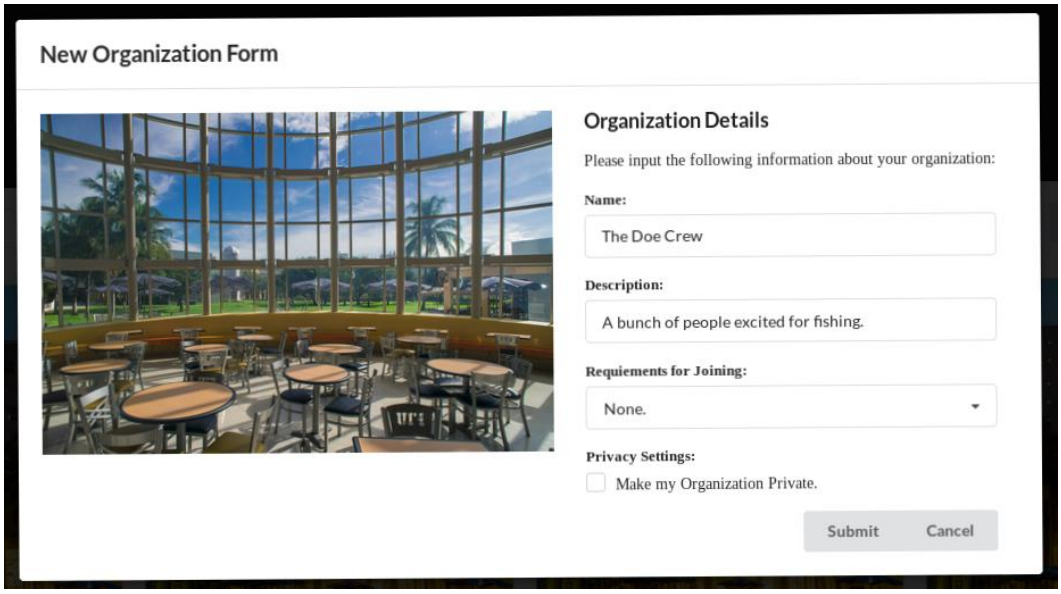

Figure 36: UI Layout for the New Organization Form modal.

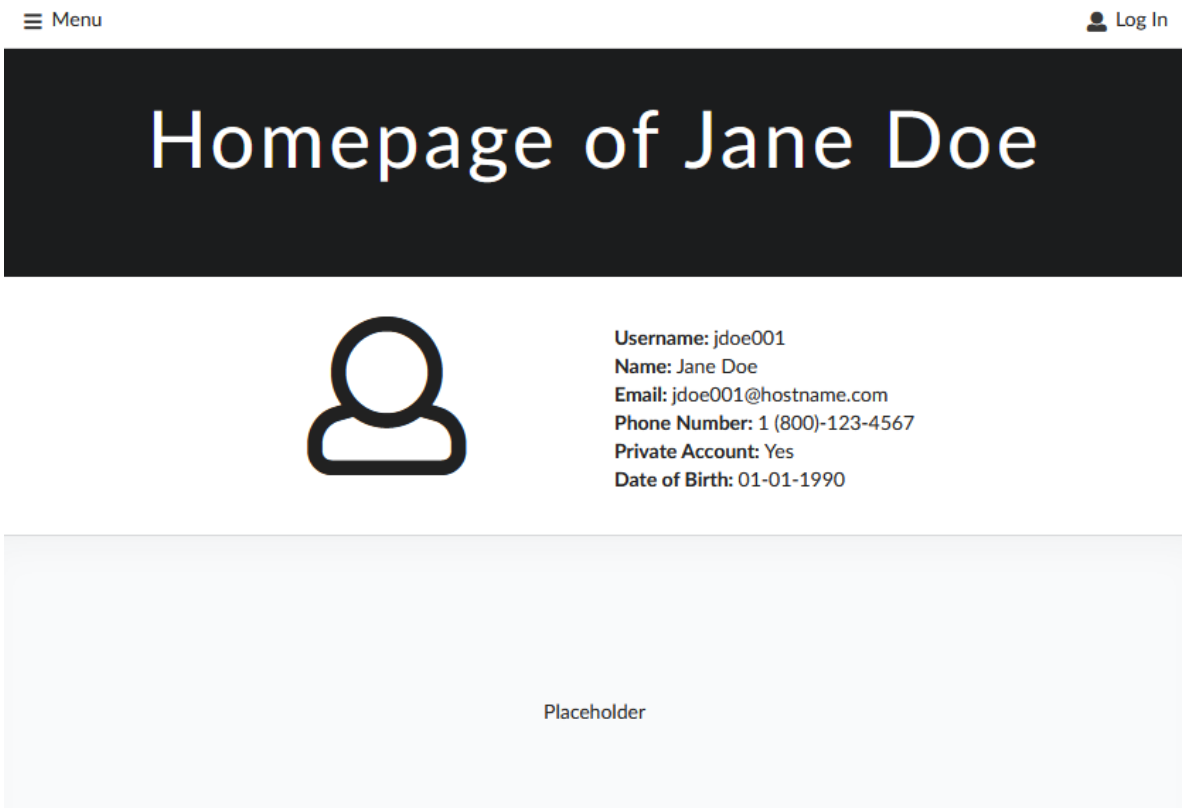

Figure 37: UI layout for the Profile page of a given user, in this case, Jane Doe. When the user is logged in, a edit profile option is given to the user, which triggers a Edit Profile Form modal (see Figure 38). The placeholder section shall include information about the user's clubs and / or events.

Figure 38: UI Layout for the Edit Profile Form modal.

9.3 Appendix C – Meeting Diaries

9.3.1 September 9, 2019

When and Where	Role
Date: 9/9	**Primary Facilitator:** Armando
Start: 1:30 pm	**Timekeeper:** Kian
End: 3:00 pm	**Minute Taker:** Anthony
Room: GL 625A	**Attending:** Armando, Kian, Teriq (late), Anthony, Yovani

1. Status

Initial meetings. Team members are introduced to each other and the initial tasks where distributed. An initial analysis of the system and its components was done.

2. Discussion

Use case requirements are a priority as everything else depends on them. Armando is assigned to formatting all the use cases into a single format (assigned the Document Editor role). As a task, Yovani and Kian have to finish their Use Cases as soon as possible.

Group split into front-end and back-end teams. Front-end team consists of Armando and Kian. Back-end team consists of Anthony, Teriq, and Yovani. Front-end team decided on using the React environment for the website. Research is to be done in several React components such as Redux, Saga, Router, etc. Front-end team needs to research the java software libraries that will be used for the backend. Some starting points are Apache, Java SQL and Restful API

The group discussed the potential entities of the system (in relation to the use cases). The following where agreed on: (a) Actors include Guests, Member, Admin, and Organizer; (b) Other system entities are Events, Clubs/Organizations.

We explored the possibility of using Geolocation and Calendars to present events. The website layout was discussed.

3. Wrap Up

- Yovani and Kian are behind on their use cases.
- Group divided into front-end and back-end teams.
- Layout and logic of the system started to be defined.

9.3.2 September 16, 2019

When and Where	Role
Date: 9/16	**Primary Facilitator:** Armando
Start: 1:30 pm	**Timekeeper:** Kian
End: 2:30 pm	**Minute Taker:** Teriq
Room: PG6 116	**Attending:** Armando, Kian, Teriq, Anthony, Yovani

1. Status

Update regarding use cases. Some members have to redo their use case in order to comply to the content and style requirements. Starting discussion about UML diagrams.

2. Discussion

To follow the requirements for the use cases, some members have to update their files. Anthony and Armando's use cases are complete and in the correct format. Teriq, Yovani, and Kian must update theirs. Information regarding what's missing was left on the project's GitHub's readme. Some use cases have to be fully redone as they overlap with outer members' use case.

In total, 18 use cases are completed. Of this group, the following set is planned for implementation: create roles, create organization, create event, two-factor authentication, cancel event, attending an event, ranking, earn points, and access events.

Some sequence diagrams are demoed by Anthony. Currently, 3 sequence diagrams (out of 10) have been completed (create role, create organization, create event). Currently, both Anthony and Armando will work on diagrams as the other team members focus on updating their use cases.

Discussion about the prototype/mock-up. The front-end team is using react and discussed about modules that would help implement geolocation, namely Google Map React.

3. Wrap Up

- Teriq, Yovani, and Kian must redo or retouch some of their use cases
- Implementation use cases have been partially decided upon.
- Some sequence diagrams where implemented.

9.3.3 September 23, 2019

When and Where	Role
Date: 9/23	**Primary Facilitator:** Armando
Start: 2:00 pm	**Timekeeper:** Kian
End: 4:00 pm	**Minute Taker:** Teriq
Room: PG6 116	**Attending:** Armando, Kian, Teriq, Anthony, Yovani

1. Status

Update regarding section 4.1 being completed. Initial prototype is also built. Also, some sequence diagrams where completed, and each member is assigned a sequence diagram to start during the meeting.

2. Discussion

Section 4.1, Use Cases, of the Requirements Elicitation is completed, which means that all use cases are set on a uniform format. With the complete set, a final implementation set can be decided on (in parenthesis who is tasked with creating the sequence diagram):

- SOS01 – create event
- SOS02 – grant organization roles*(Armando)
- SOS04 – attending event
- SOS10 – access events by location*(Teriq)
- SOS14 – create roles
- SOS16 – create organization
- SOS17 – cancel event
- SOS18 – create a task*(Yovanni)
- SOS12 – set up 2 Factor authorization
- SOS07 – set private accounts* (Kian)

The back-end group discussed how to implement the web service. Socket.io, which has implementations for java and react, so intercommunication should be simplified. For database, the group decided on using a java implementation of SQL.

Tasks were assigned for the following week for each member: Armando will work on chapter 3 of the SRD, project plan. He is also tasked with editing the document and making sure style, format, and language is uniform. Teriq will work on chapter 2 of the SRD, current systems. Kian will work on chapter 1 of the SRD, introduction. Yovani and Anthony will work on chapter 5, including object diagrams and sequence diagrams.

3. Wrap Up

- The implementation use cases where decided upon.
- The back-end group decided on Socket.io for the core web-service functionality.
- Tasks were assigned to each member for the following week.