CoCal Calendar

Team

Justin Schiller
Adrian Chen
Justin McBride
Zachary Lamb
Warren Ferrell

Project Summary

CoCal is a web application that allows users to create, attend, and share public or private events in a friendly user-facing calendar application. Users will have their own personal calendars hold events that they're interested in or are attending, allowing them to quickly view all upcoming and past events in an intuitive way. Furthermore, users can make these events available to the public, inviting others to also become involved. Even better, groups can be created that users can subscribe to, wherein events are shared between all members of the group, both on the private group calendar and on the users' private, personal calendars.

Table of Contents

Project Requirements:

User Requirements

ID	Requirement
US.01	As a user, I want to be able to create a new personal account on CoCal.
US.02	As a user, I want to be able to log in to my account.
US.03	As a user, I want to be able to recover my account password.
US.04	As a user, I want to be able to view my calendar.
US.05	As a user, I want to add an event to a calendar.
US.06	As a user, I want to share a calendar.
US.07	As a user, I want to be able to view events in a calendar shared with me.
US.08	As an admin, I want to remove any event.
US.09	As a user, I want to share events with other users.
US.10	As a user, I want to link a public event with my calendar.
US.11	As a user, I want to remove an event from my calendar.
US.12	As a user, I want to be able to create a group.
US.13	As a user, I want to be able to join a group.
US.14	As a user, I want to filter events by type.
US.15	As a user, I want to filter events by cost.
US.16	As a user, I want to filter events by location.
US.17	As a user, I want to be able to log out of my account.
US.18	As a user, I want to report events.
US.19	As an admin, I want to receive emails about reported events.
US.20	As a user, I want to remove my events.
US.21	As a user, I want to edit my event’s details.
US.22	As an admin, I want to remove users from groups.

Business Requirements

ID	Requirement
BR.01	Account names must be a valid email address.

BR.02	No inappropriate public events.
BR.03	Events exist indefinitely, unless manually removed.

Functional Requirements

ID	Requirement
FR.01	Our system needs to maintain records about users, organized into accounts.
FR.02	Each user has a personal, private calendar.
FR.03	Each group has a private calendar.
FR.04	Each event must have a title, description, date/time, location, cost, type, and administrator(s)
FR.05	Our system must have a web-based user interface.
FR.06	System must be able to sort events by category.

Non-Functional Requirements

ID	Requirement	
NF.01	Supportability:	McBride maintains bugs.
NF.02	Reliability:	Viewing a deleted event shows an error message.
NF.03	Usability:	Easily usable by non-techy person.
NF.04	Reliability:	Events should not be lost once added
NF.05	Performance:	Events should auto-populate in under 30 seconds
NF.06	Implementation:	Web interface should work with most major browsers.

Use Cases:

Use Case ID:	**UC-00**		
Requirements:	US.01, BR.01, FR.01, FR.02		
Use Case Name:	Create a user account		
Description:	Creating an account requires filling out a form with personal details.		
Actors:	User		
Pre-conditions:	---		
Post-conditions:	User exists in database		
Freq. of Use:	Often		
Flow of Events:	**#**	**Actor Action**	**System Response**
	1	Type in desired user name	---
	2	Type in desired password	---
	3	Verify desired password	---
	4	Type in email	---
	5	Press “Create”	Validate that user name is unique and alphanumeric; validate that password meets requirements; send confirmation email to email account; create database entry for user
Variations:	**#**	**Actor Action**	**System Response**
	5	Press “Create”	Validate that user name is unique and alphanumeric; validate that password meets requirements; information fails validation; prompt user for better information
Notes and	When creating a user, the information needs to be validated to		

Issues:	ensure that the user is real. A CAPTCHA may be used eventually to facilitate this validation. User names can only be alpanumeric characters.
Developer Notes:	Check the database to verify that the user name is unique. Run a password regex expression to validate password meets requirements. Make API call to send email to user containing their user name.

Use Case ID:	**UC-01**		
Requirements	US.02, US.04		
Use Case Name:	Log In		
Description:	User can log in to their CoCal account.		
Actors:	User		
Pre-conditions:	User must have an account.		
Post-conditions:	User signed in to their personal account. User presented with personal calendar view.		
Freq. of Use:	Very frequent use.		
Flow of Events:	**#**	**Actor Action**	**System Response**
	1	Enter email.	N/A
	2	Enter password.	System validates credentials. System directs user to personal calendar.
Variations:	**2**	Enter password	System fails to validate credentials. System displays invalid credentials message. System clears fields for user to retry.
Notes and Issues:			
Developer Notes:			

<table>
<tr><td><b>Use Case ID:</b></td><td colspan="3"><b>UC-02</b></td></tr>
<tr><td><b>Requirements:</b></td><td colspan="3">US.05, FR.04, BR.03</td></tr>
<tr><td><b>Use Case Name:</b></td><td colspan="3">Adding an Event</td></tr>
<tr><td><b>Description:</b></td><td colspan="3">A user needs to be able to add an event.</td></tr>
<tr><td><b>Actors:</b></td><td colspan="3">Users, Admin</td></tr>
<tr><td><b>Pre-conditions:</b></td><td colspan="3">User logged in.</td></tr>
<tr><td><b>Post-conditions:</b></td><td colspan="3">Event saved to calendar.</td></tr>
<tr><td><b>Freq. of Use:</b></td><td colspan="3">Frequently throughout the day by users.</td></tr>
<tr><td rowspan="6"><b>Flow of Events:</b></td><td><b>#</b></td><td><b>Actor Action</b></td><td><b>System Response</b></td></tr>
<tr><td>1</td><td>User clicks the “Add Event” button.</td><td>New Event window pops up.</td></tr>
<tr><td>2</td><td>User fills in event name.</td><td></td></tr>
<tr><td>3</td><td>User fills in event details.</td><td></td></tr>
<tr><td>4</td><td>User fills in event descriptions.</td><td></td></tr>
<tr><td>5</td><td>User clicks “Accept” button.</td><td>System saves the event.
System gives confirmation.</td></tr>
<tr><td rowspan="2"><b>Variations:</b></td><td colspan="3">Before User clicks accept, User can assign event to a group.</td></tr>
<tr><td colspan="3">Before User clicks accept, User can click a checkbox to make an event private.</td></tr>
<tr><td><b>Notes and Issues:</b></td><td colspan="3"></td></tr>
<tr><td><b>Developer Notes:</b></td><td colspan="3"></td></tr>
</table>

Use Case ID:	**UC-03**
Requirements:	US.03
Use Case Name:	Recover user password
Description:	User resets

Actors:	Users		
Pre-conditions:	User is on login page User does not remember password		
Post-conditions:	User’s password is changed		
Freq. of Use:	Weekly by each user		
Flow of Events:	**#**	**Actor Action**	**System Response**
	1	Click forgot password button.	Server sends forgot password page.
	2	Enter email address.	n/a
	3	Click reset password button.	Server sends reset password link to user’s email
	4	Click reset password link in email.	Server sends password reset page.
	5	Enter new password.	n/a
	6	Click reset password button.	Database changes user’s password. Server sends login page.
Variations:			
Notes and Issues:			
Developer Notes:			

Use Case ID:	**UC-04**
Requirements:	US.14-16, FR.06
Use Case Name:	Looking at events near user
Description:	Users will be able to see events based on location, type, cost
Actors:	Users
Pre-conditions:	Login,have access to calendar events, and have calendar open
Post-conditions:	User sees list of events by remaining categories

Freq. of Use:	Often		
Flow of Events:	**#**	**Actor Action**	**System Response**
	1.	User clicks Find Events Near Me button	Show events on calendar within 10 miles
Variations:	2.	User selects filter(s) from column dropdowns	Show events that correspond to selected filter
Notes and Issues:	User can filter by type, cost, change distance, and/or date range as well		
Developer Notes:	Make sure we include attribute for tag in database table		

Use Case ID:	**UC-05**		
Requirements:	US.07		
Use Case Name:	Look at user calendar		
Description:	A user should be able to look at another user’s public calendar.		
Actors:	User		
Pre-conditions:	User is logged in.		
Post-conditions:	A user’s calendar is shown.		
Freq. of Use:	Frequently throughout the day by users.		
Flow of Events:	**#**	**Actor Action**	**System Response**
	1	User clicks “Find User” button.	System requests an email address.
	2	User fills in the desired user’s email address.	
	3	User clicks “OK”.	System shows the public calendar associated with the email address.
Variations:	**#**	**Actor Action**	**System Response**
	1	User clicks “Home” button.	System shows the signed in user’s calendar.
Notes and			

Issues:	
Developer Notes:	

Use Case ID:	**UC-06**		
Requirements:	US.10		
Use Case Name:	Signing up for an event (mark as going)		
Description:	User can link public calendar event to their private calendar and show intent to attend		
Actors:	User		
Pre-conditions:	User is logged in, user is on public calendar page, user clicks on view event in which a popup with event details comes up		
Post-conditions:	public event is on private calendar of user, user is in list of attendees		
Freq. of Use:	often		
Flow of Events:	**#**	**Actor Action**	**System Response**
	1.	User clicks on button “Attend Event”	Shows message that says “added event to your calendar”. Show message that says “you are attending” + event. Updates event attendees to include user. Adds event to user calendar database table which is then displayed on user calendar
Variations:	Event is already added into user’s calendar,		
Notes and Issues:			
Developer Notes:	Make sure to add in link in order to delete event on user’s calendar if public calendar is deleted		

Use Case ID:	**UC-07**
Requirements:	US.12, FR.03

<table>
<tr><td><b>Use Case Name:</b></td><td colspan="3">Con</td></tr>
<tr><td><b>Description:</b></td><td colspan="3">Users can create a group which other users can later join. Each group has its own calendar of events, and an admin.</td></tr>
<tr><td><b>Actors:</b></td><td colspan="3">User</td></tr>
<tr><td><b>Pre-conditions:</b></td><td colspan="3">Log In</td></tr>
<tr><td><b>Post-conditions:</b></td><td colspan="3">A group exists with the specified name; the admin of that group is the user who created it.</td></tr>
<tr><td><b>Freq. of Use:</b></td><td colspan="3">Often</td></tr>
<tr><td rowspan="3"><b>Flow of Events:</b></td><td><b>#</b></td><td><b>Actor Action</b></td><td><b>System Response</b></td></tr>
<tr><td><b>1</b></td><td>Enter group name</td><td>--</td></tr>
<tr><td><b>2</b></td><td>Press “Create”</td><td>A new database entry is created with the group name, and creating user as the admin of that group.</td></tr>
<tr><td rowspan="2"><b>VariConations:</b></td><td><b>#</b></td><td><b>Actor Action</b></td><td><b>System Response</b></td></tr>
<tr><td><b>2</b></td><td>Press “Create”</td><td>Group name already exists; display error to user; prompt for new group name.</td></tr>
<tr><td><b>Notes and Issues:</b></td><td colspan="3">Creating a group inserts a new group into the database. This group has its own unique name and calendar. The creating user is the admin of the group and is given special privileges.</td></tr>
<tr><td><b>Developer Notes:</b></td><td colspan="3">Verify that the group name is unique.</td></tr>
</table>

Use Case ID:	**UC-08**
Requirements:	US.21
Use Case Name:	Editing an Event
Description:	User who created event or Admin can edit created event
Actors:	User who created event/Admin
Pre-conditions:	User logged in. Event exists. User has permission to edit event.

	Viewing event.		
Post-conditions:	Modifications to event persist to public calendar.		
Freq. of Use:	Varies by event: medium frequency.		
Flow of Events:	**#**	**Actor Action**	**System Response**
	1	User clicks “Edit Event” button.	System directs user to event description with ability to edit.
	2	User edits desired event information (date, time, location, etc…).	None.
	3	User clicks “Save Updates” button.	System updates event on relevant calendars with new information.
Variations:	**1**	User clicks “Edit Event” button without permission to edit.	System displays lack of permission message. Use case ends.
Notes and Issues:			
Developer Notes:	Potentially use mutex to disable multiple users editing event concurrently.		

Use Case ID:	**UC-09**
Requirements:	US.08, US.20
Use Case Name:	Remove Event
Description:	User deletes an event.
Actors:	Users, Admins
Pre-conditions:	User is logged in Event exists. User is viewing event. User has write access to an event.
Post-conditions:	Event information is removed from the database. Requests for event page return event deleted page.

<table>
<tr><td><b>Freq. of Use:</b></td><td colspan="3">Weekly by each user.</td></tr>
<tr><td><b>Flow of Events:</b></td><td><b>#</b></td><td><b>Actor Action</b></td><td><b>System Response</b></td></tr>
<tr><td></td><td><b>1</b></td><td>User clicks remove event button.</td><td>Pop up confirmation dialog.</td></tr>
<tr><td></td><td><b>2</b></td><td>User clicks yes.</td><td>Removes event information.
Toggle event as deleted.</td></tr>
<tr><td><b>Variations:</b></td><td><b>2</b></td><td>User clicks no.</td><td>n/a</td></tr>
<tr><td><b>Notes and Issues:</b></td><td colspan="3"></td></tr>
<tr><td><b>Developer Notes:</b></td><td colspan="3">User should only see remove event button if they have write access.
Navigating to deleted events returns event deleted page.
Admins can delete any event.</td></tr>
</table>

<table>
<tr><td><b>Use Case ID:</b></td><td colspan="3"><b>UC-10</b></td></tr>
<tr><td><b>Requirements:</b></td><td colspan="3">FR.05</td></tr>
<tr><td><b>Use Case Name:</b></td><td colspan="3">Viewing Event Details</td></tr>
<tr><td><b>Description:</b></td><td colspan="3">When a user clicks on an event, they should be taken to an event page unique to that event that lists the full details.</td></tr>
<tr><td><b>Actors:</b></td><td colspan="3">User</td></tr>
<tr><td><b>Pre-conditions:</b></td><td colspan="3">User logged in.
Event exists in calendar visible to user.</td></tr>
<tr><td><b>Post-conditions:</b></td><td colspan="3">User can view event details.</td></tr>
<tr><td><b>Freq. of Use:</b></td><td colspan="3">Very high frequency.</td></tr>
<tr><td><b>Flow of Events:</b></td><td><b>#</b></td><td><b>Actor Action</b></td><td><b>System Response</b></td></tr>
<tr><td></td><td><b>1</b></td><td>User clicks on event.</td><td>System redirects user to event page</td></tr>
<tr><td><b>Variations:</b></td><td colspan="3"></td></tr>
<tr><td><b>Notes and Issues:</b></td><td colspan="3"></td></tr>
<tr><td><b>Developer Notes:</b></td><td colspan="3"></td></tr>
</table>

<table>
<tr><td><b>Use Case ID:</b></td><td colspan="3"><b>UC-11</b></td></tr>
<tr><td><b>Requirements:</b></td><td colspan="3">US.10</td></tr>
<tr><td><b>Use Case Name:</b></td><td colspan="3">Adding a public event to personal calendar</td></tr>
<tr><td><b>Description:</b></td><td colspan="3">Public events that are found on other user’s calendars, group’s calendars, or public calendars can be added to a user’s personal calendar. This addition refers to the original event and contains all the same information.</td></tr>
<tr><td><b>Actors:</b></td><td colspan="3">User</td></tr>
<tr><td><b>Pre-conditions:</b></td><td colspan="3">User Logged In
User is viewing a specific event</td></tr>
<tr><td><b>Post-conditions:</b></td><td colspan="3">User’s personal calendar contains reference to added event</td></tr>
<tr><td><b>Freq. of Use:</b></td><td colspan="3">Often</td></tr>
<tr><td rowspan="2"><b>Flow of Events:</b></td><td><b>#</b></td><td><b>Actor Action</b></td><td><b>System Response</b></td></tr>
<tr><td><b>1</b></td><td>Clicks “Interested”</td><td>Copy event reference ID into user’s personal calendar.
Display confirmation message.
Increment interested counter in event.</td></tr>
<tr><td><b>Variations:</b></td><td colspan="3">---</td></tr>
<tr><td><b>Notes and Issues:</b></td><td colspan="3">---</td></tr>
<tr><td><b>Developer Notes:</b></td><td colspan="3">Ensure that the event is still valid when the event is referenced to be added. A user may sit on a specific event’s page for multiple hours, wherein it may have been deleted in the meantime.</td></tr>
</table>

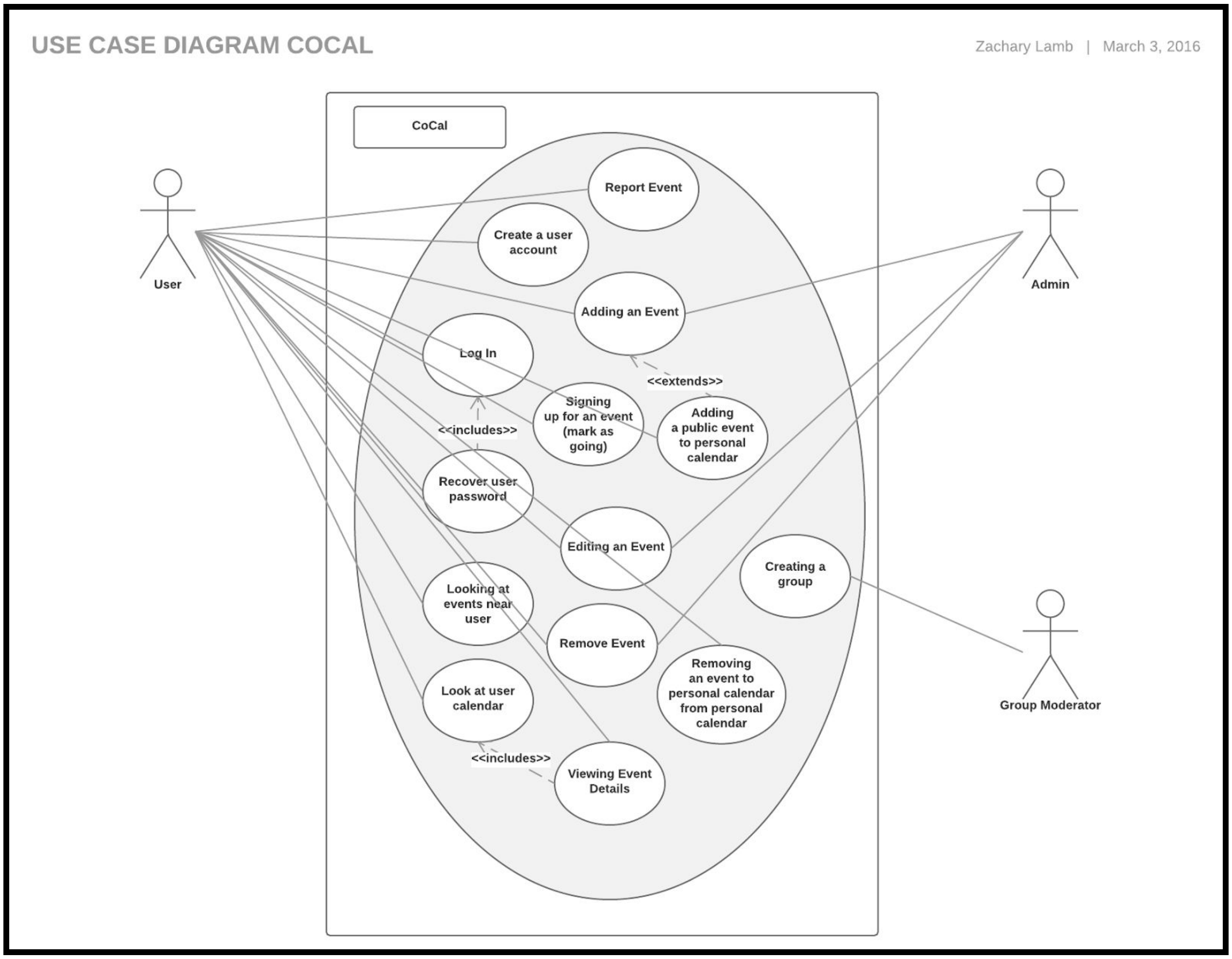
Use Case ID:	**UC-12**
Requirements:	US.11
Use Case Name:	Removing an event added to personal calendar from personal calendar.
Description:	Sometimes a user will want to remove an added event from
Actors:	User

Pre-conditions:	User logged in. User added an event to their personal calendar.		
Post-conditions:	Deleted event is no longer in the user's personal calendar.		
Freq. of Use:	Semi-frequently		
Flow of Events:	**#**	**Actor Action**	**System Response**
	1	Views a specific event	---
	2	Clicks "Remove"	Removes the reference to the event from the user's personal calendar
Variations:	---		
Notes and Issues:	---		
Developer Notes:	Ensure that the event is still valid when removing the event.		

Use Case ID:	**UC-13**		
Requirements:	US.18, US.19, BR.02		
Use Case Name:	Report Event		
Description:	Inappropriate content should be removed from the public calendar by admins		
Actors:	Users		
Pre-conditions:	User logged in. Event exists. User is viewing event.		
Post-conditions:	Admins receive an inappropriate event email.		
Freq. of Use:	Rarely		
Flow of Events:	**#**	**Actor Action**	**System Response**
	1	User clicks report event button.	Database increments inappropriate marker System sends email to all

			admins
Variations:		Event already reported	Database increments inappropriate marker
	1	Admin clicks mark okay button	Database resets inappropriate marker to 0
Notes and Issues:			
Developer Notes:	Admins see mark okay button instead of report event		

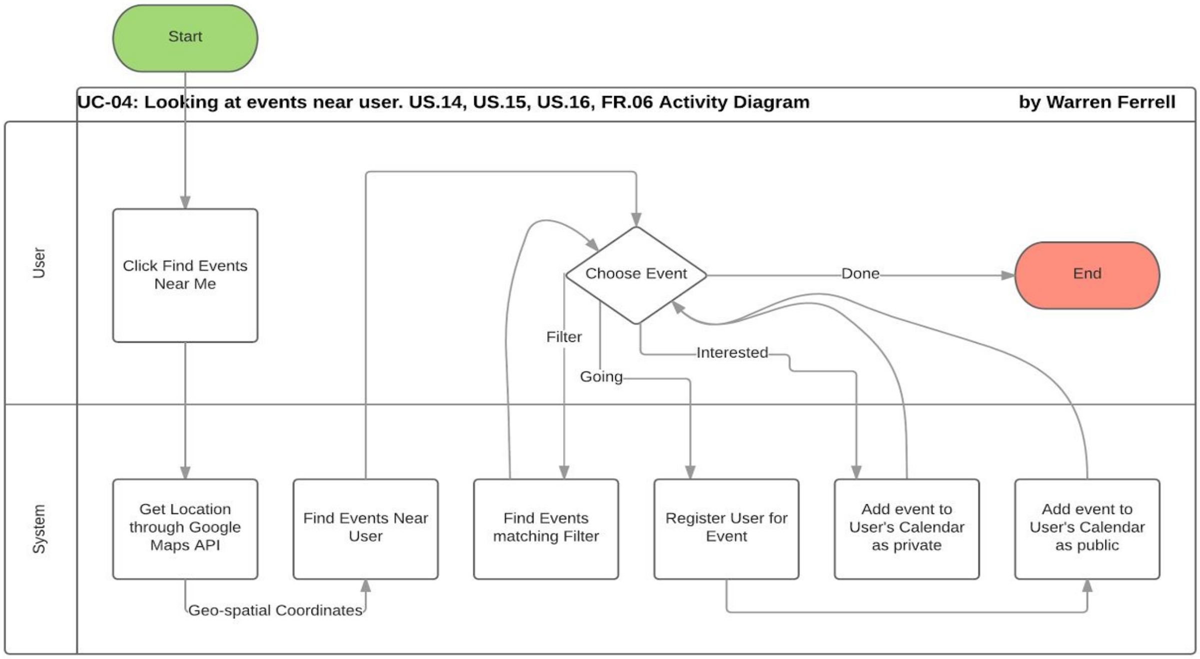
Use Case Diagram:



USE CASE DIAGRAM COCAL

Zachary Lamb | March 3, 2016

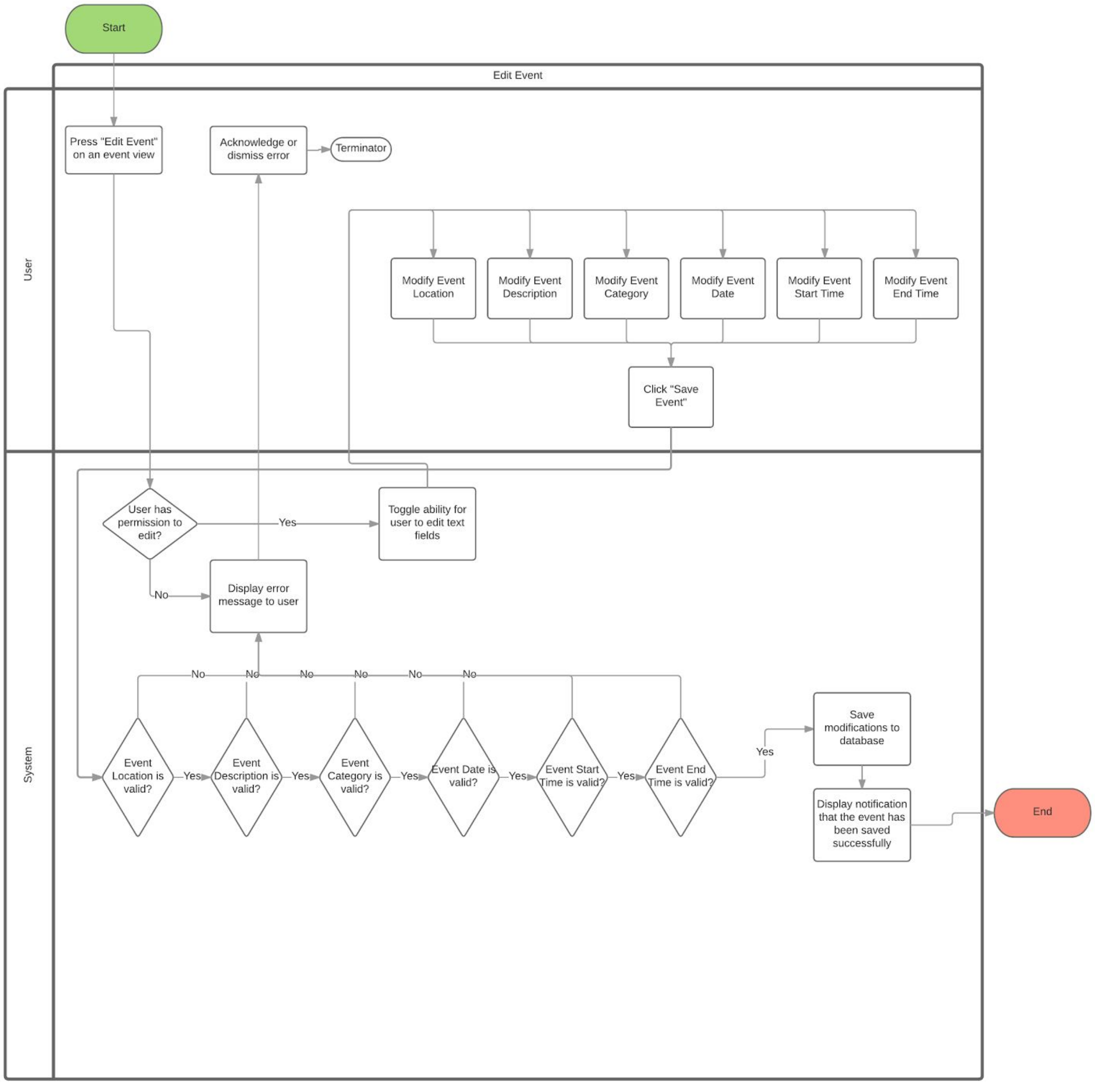
Activity Diagrams:


Start
UC-04: Looking at events near user. US.14, US.15, US.16, FR.06 Activity Diagram
by Warren Ferrell
User
System
Click Find Events Near Me
Choose Event
Done
End
Filter
Going
Interested
Get Location through Google Maps API
Find Events Near User
Find Events matching Filter
Register User for Event
Add event to User's Calendar as private
Add event to User's Calendar as public
Geo-spatial Coordinates

Event Search Activity Diagram: User Requirements 14, 15, and 16. Use Case 4: Looking at events near user. Users want to be able to find events nearby to their current location and view their details. Implemented by Warren Ferrell.

ACTIVITY DIAGRAM -- EDITING AN EVENT

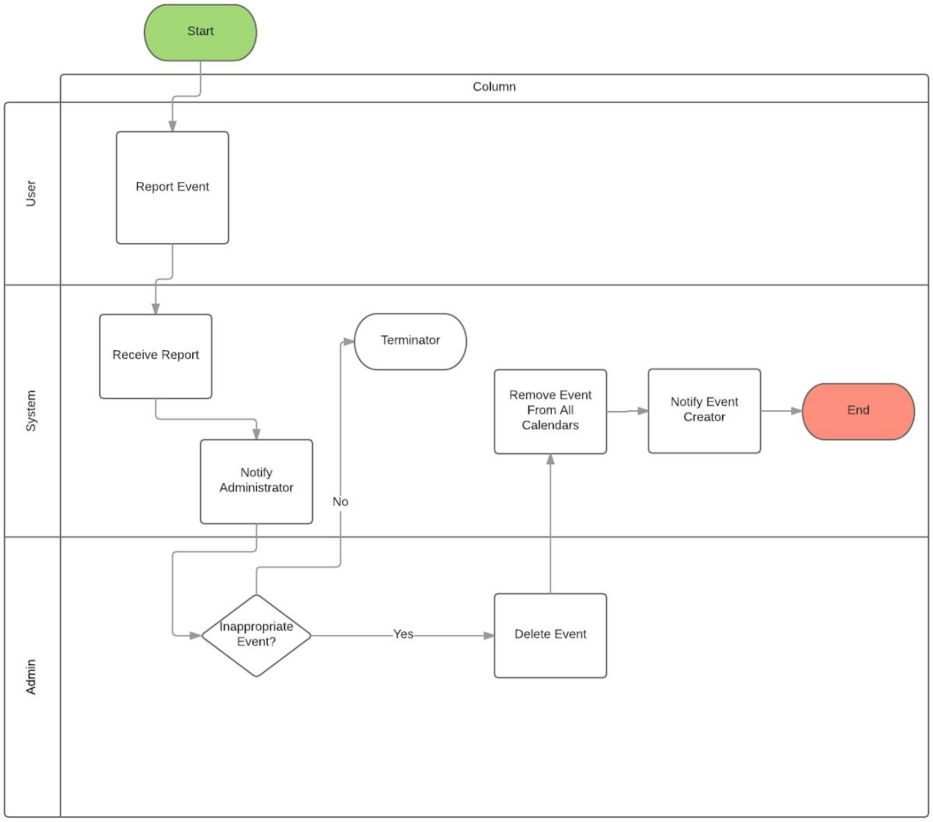
Justin McBride | March 4, 2016



Edit Event Activity Diagram: User Requirement 21. Use Case 8: Edit Event. Users want to be able to edit an event’s details and have it reflected indefinitely. Implemented by Justin McBride.

ACTIVITY DIAGRAM: REPORT EVENT

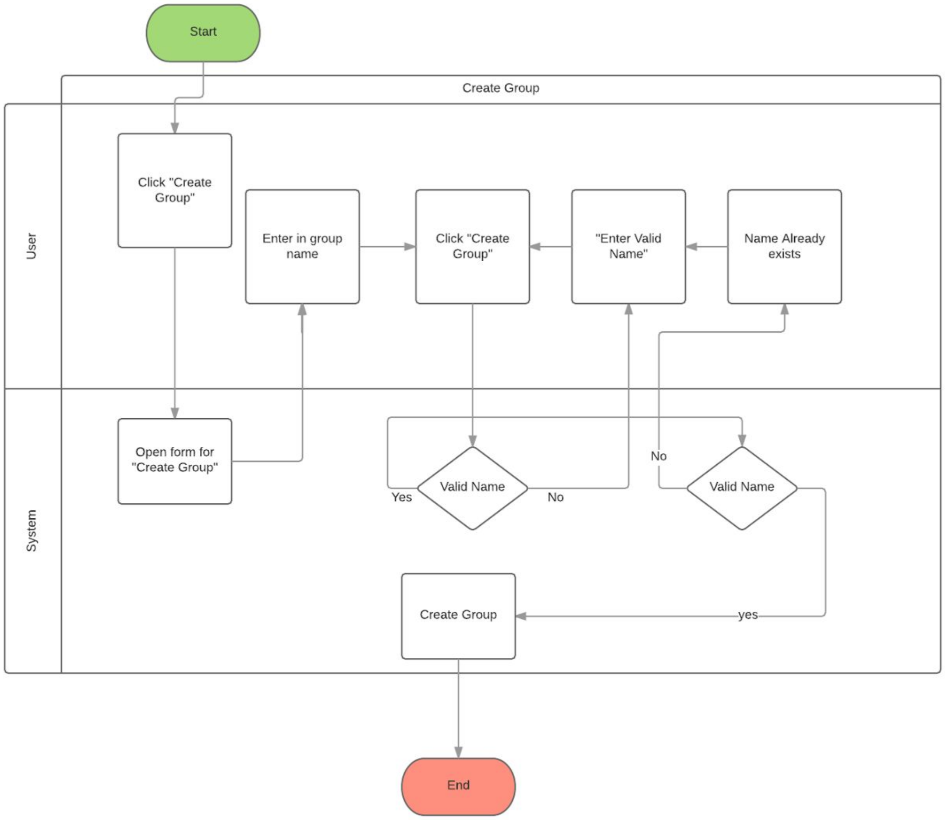
Justin Schiller | March 4, 2016



Report Event Activity Diagram: User Requirements 18 and 8. Use Case 13: Report Event. Users want to be able to report an event for inappropriate content and have it removed from the public calendar. Implemented by Justin Schiller.

ACTIVITY DIAGRAM: UC-07: CREATE A GROUP

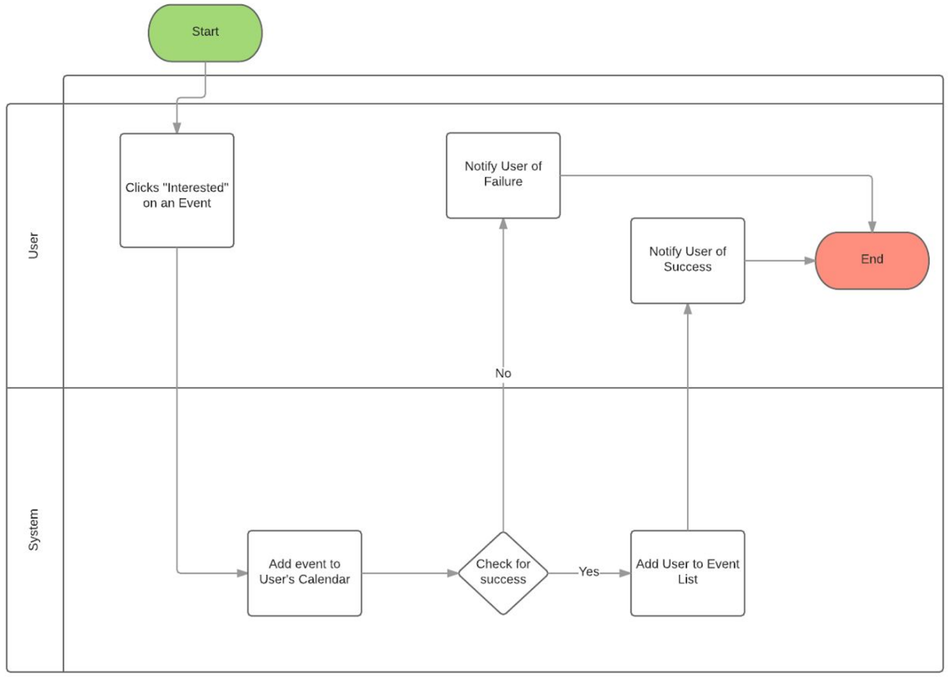
Zachary Lamb | March 9, 2016



Create a Group Activity Diagram: User Requirements 12 and Functional Requirement 03. Use Case 07: Create a Group. User wants to create a group and add other users to it. Implemented by Zachary Lamb.

ACTIVITY DIAGRAM: ADD A PUBLIC EVENT TO A PERSONAL CALENDAR

Adrian Chen | March 9, 2016



Add a Public Event to a Personal Calendar Activity Diagram: User Requirement 10 and Use Case 11. A User wants to subscribe to an event to be able to track it within their own calendar. Implemented by Adrian Chen

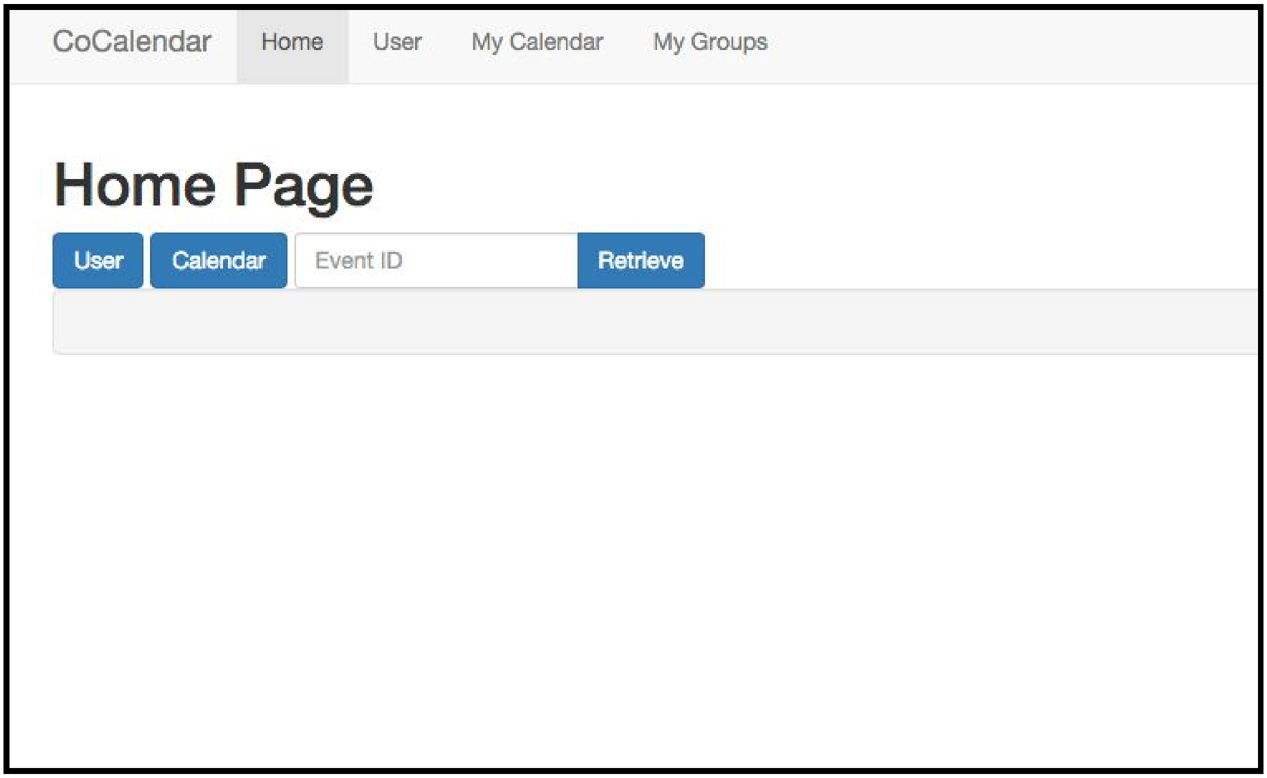
Data Storage:

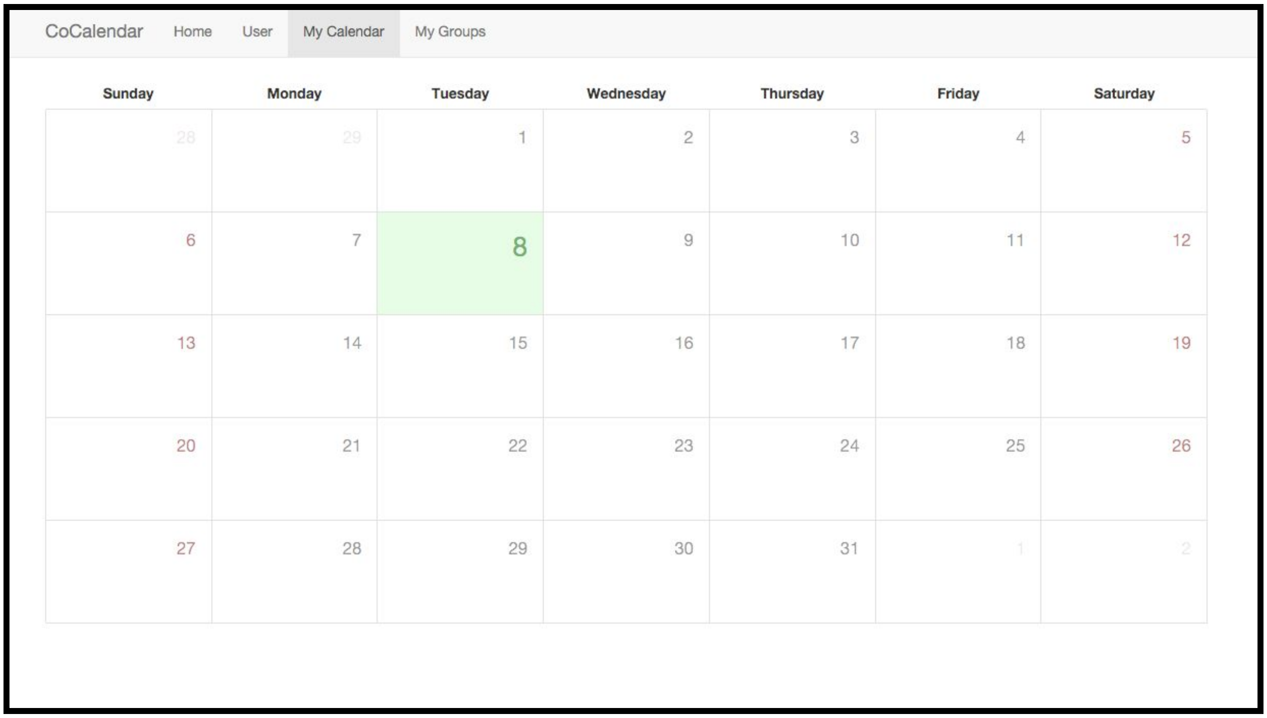
MongoDB will sustain all of our application data. Each individual entity will be stored in MongoDB through a NodeJS RESTful API. The reason that we wrap the MongoDB inside of our NodeJS is to filter out misinformation and to easily allow swapping of the database backend with no change to the graphical frontend.

Classes:

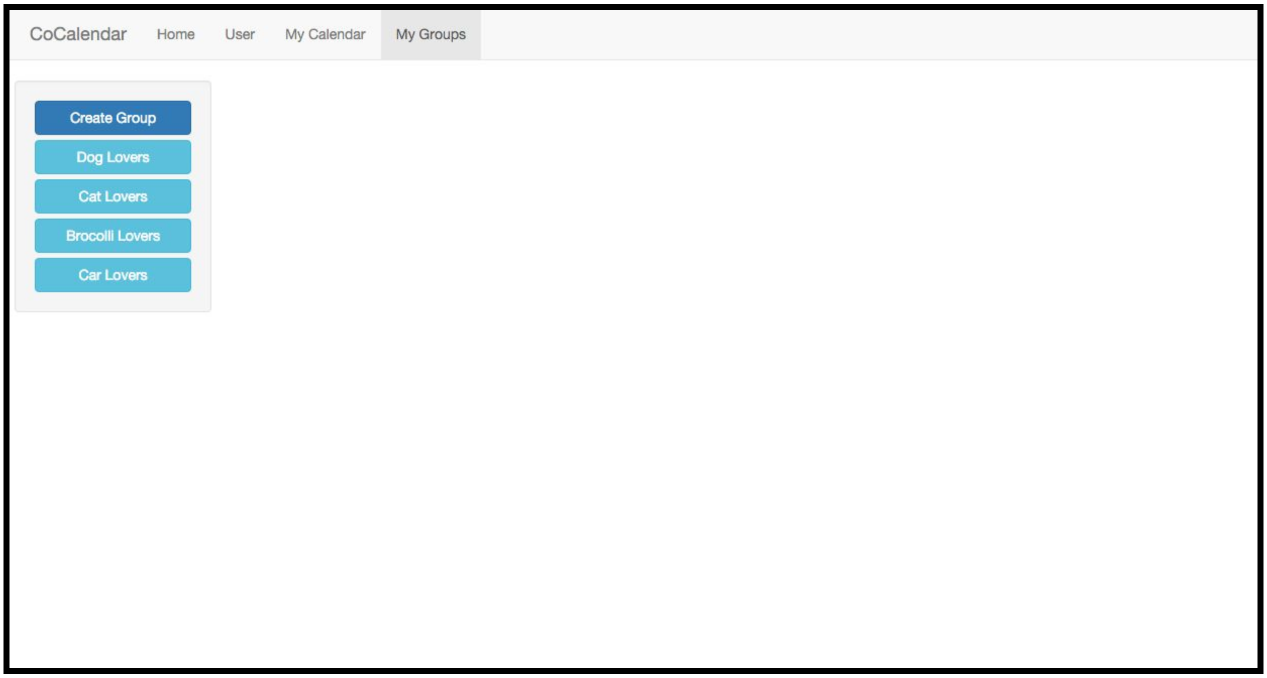
- User: An actual user of the system. This user might be an administrator, a moderator of an event, or a moderator of a group. They are all represented under this class.
- Group: A group will be a collection of users that has a name. This group can contain a single user, or many dozens, as long as there is at least one. The group also has a calendar owned by the group, which is only visible to members of that group.
- Event: A specific time and place that an activity is happening is represented by an event. It will contain details such as the start and end times, location, and description. The visibility of an event is decided by the creator.
- Calendar: It's simply a collection of events that are common to some entity, be that a group or a user.

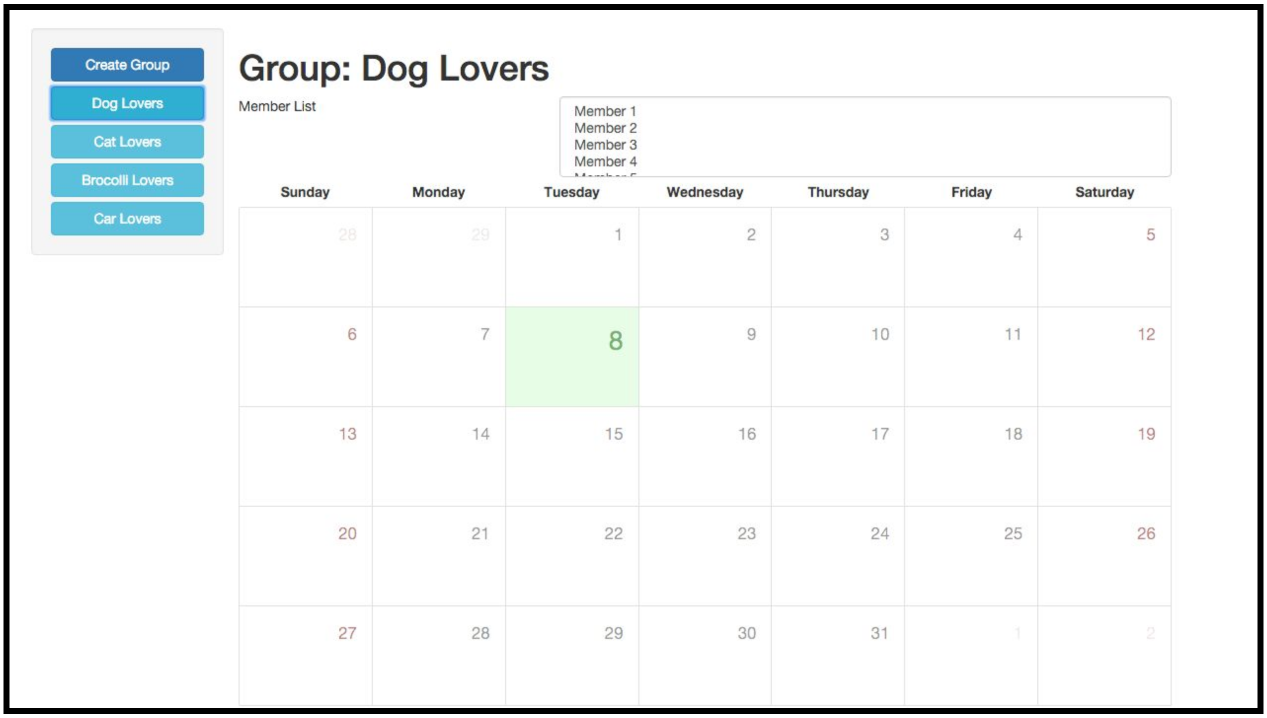
UI Mockups:





Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
28	29	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2





Group: Dog Lovers

Member List

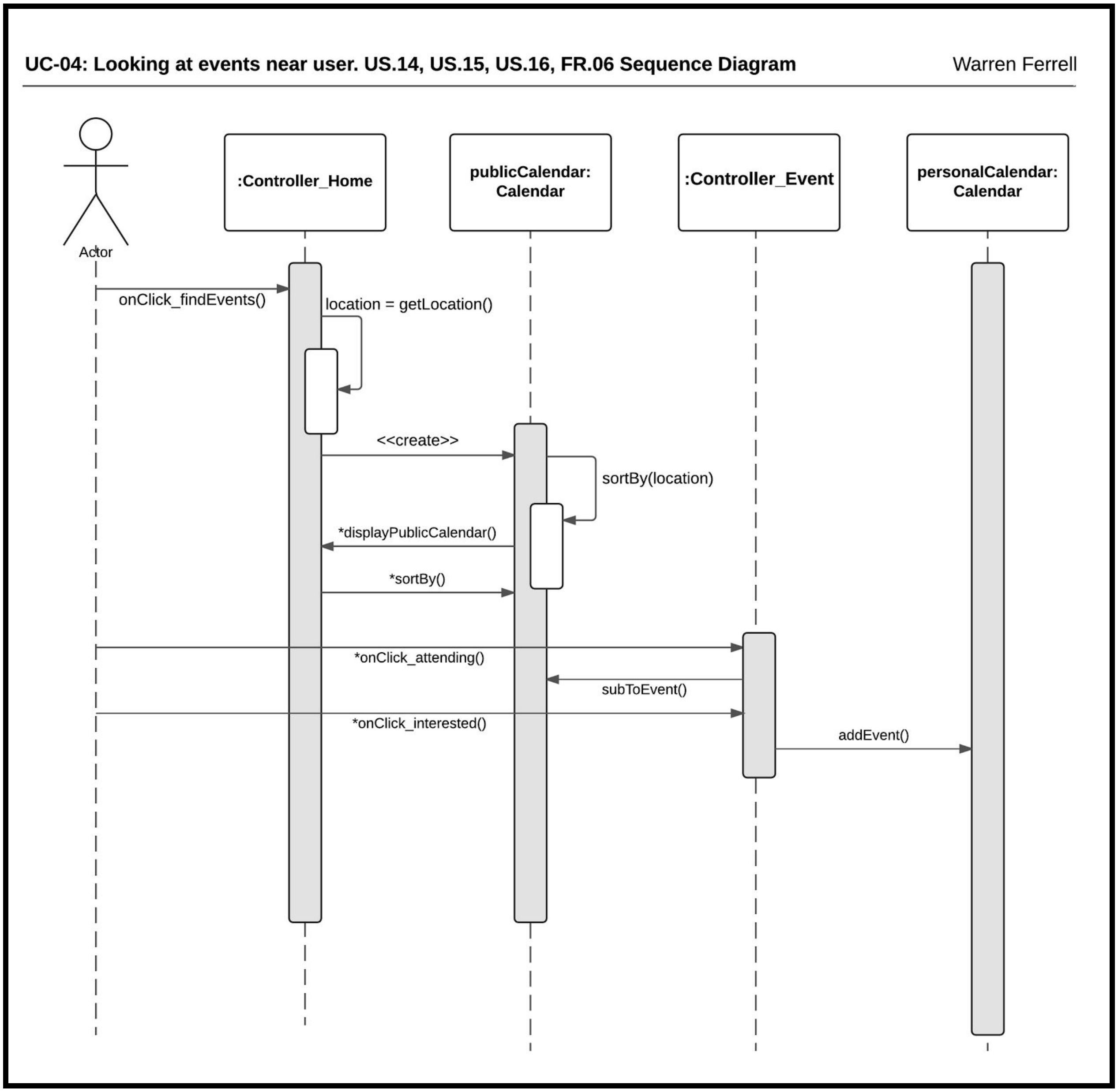
Member 1
Member 2
Member 3
Member 4

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
28	29	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2

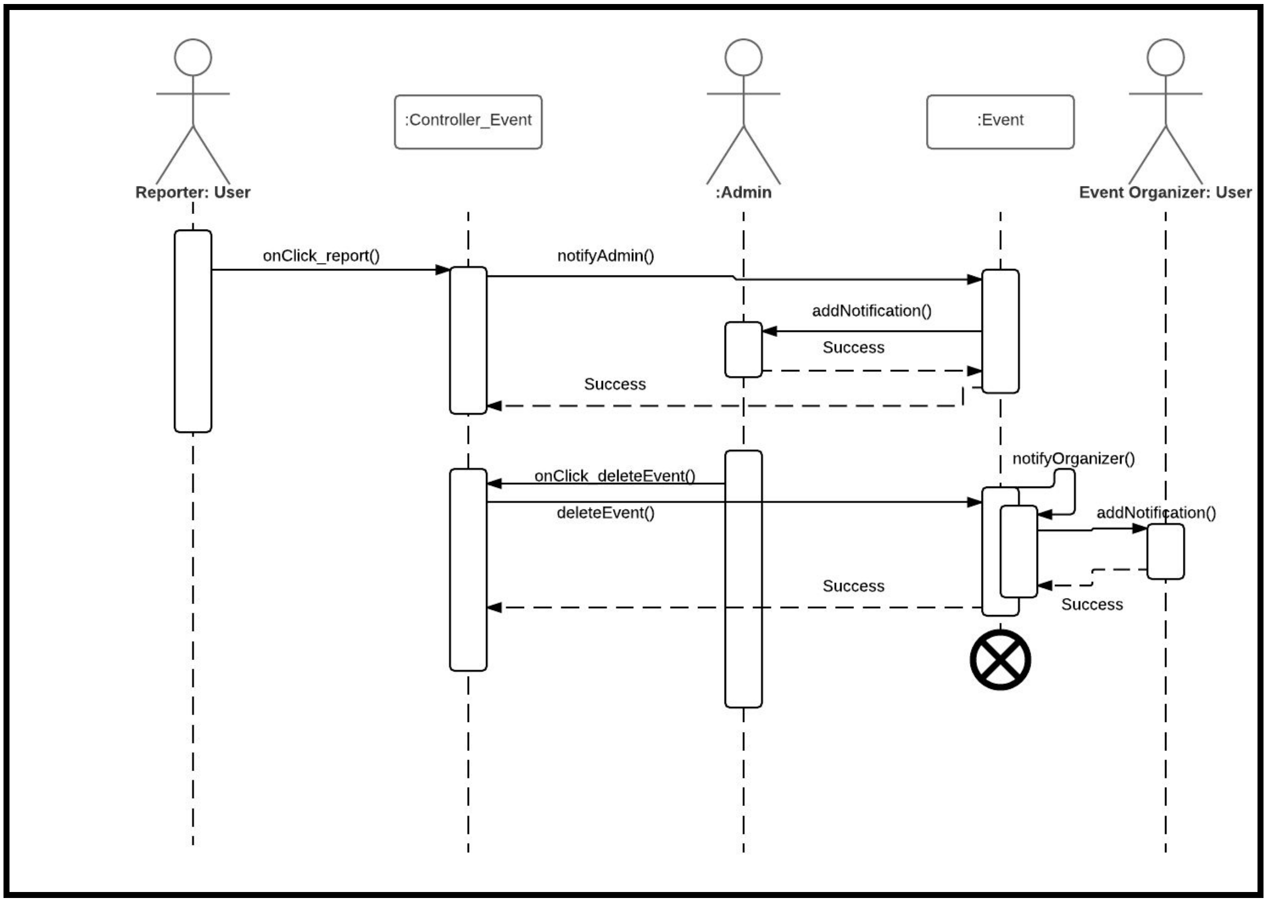
- **What will a user see as they work through the tasks identified in your use cases?**
    - There are five web pages or "views" if you will that the user will interact with. They will be given feedback as they click on listeners per standard web design.

- **What is the overall organization of your user interface?**
    - The user is greeted by a homepage which will then show them options to look at a visual overview of the calendar, the group that they are in ,and also it will show them personal info such as email and username
- **How will data be displayed?**
    - Event data will show up on the calendar, group data such as events will show up on a separate calendar under the Groups view. Members will of the group will show up as a list next to the calendar as well.
- **How will the user navigate from screen to screen?**
    - There will be standard navigation bar at the top of the page ,and we will have event listeners for creating events, groups, etc. for popups.

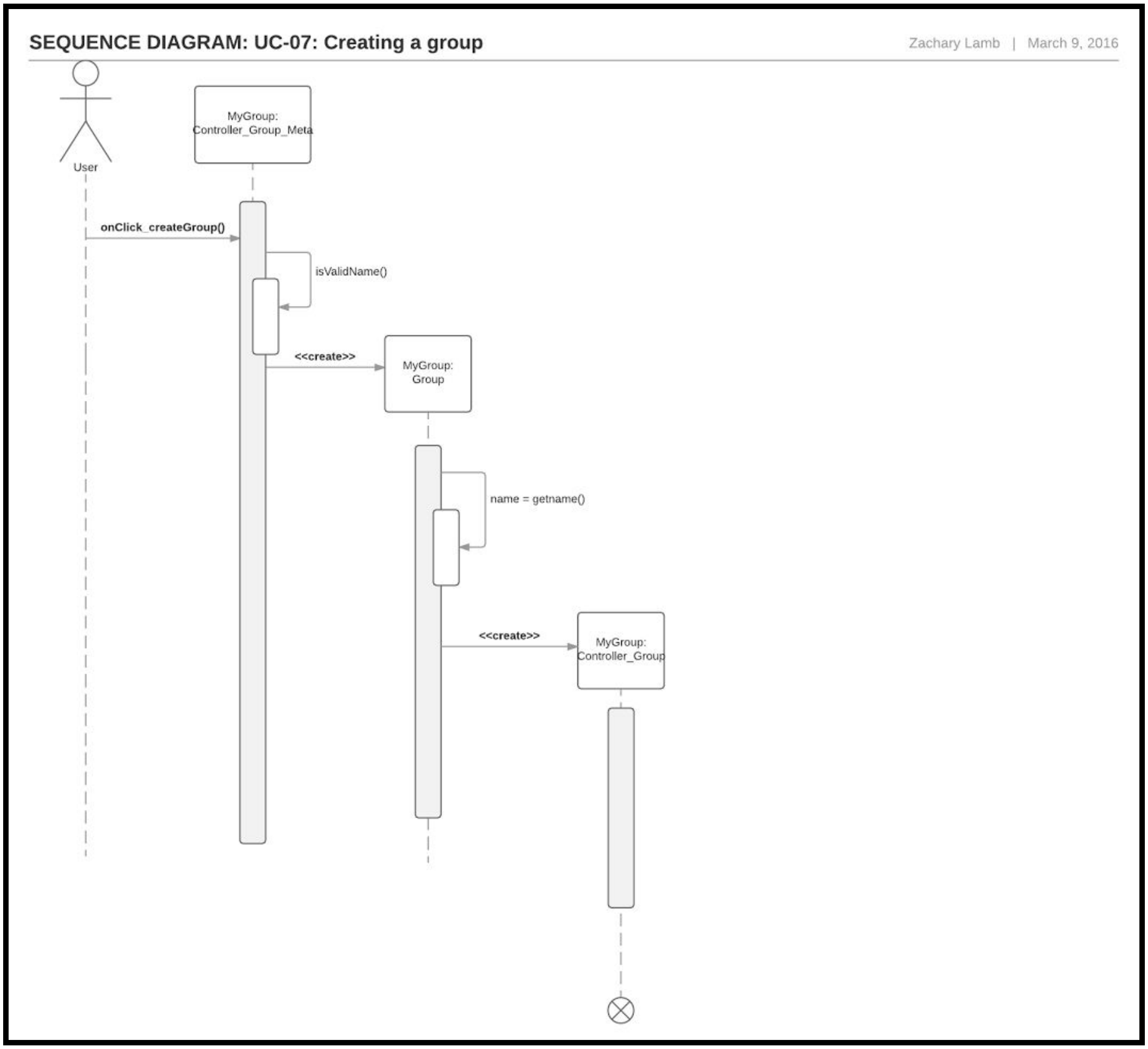
Sequence Diagrams



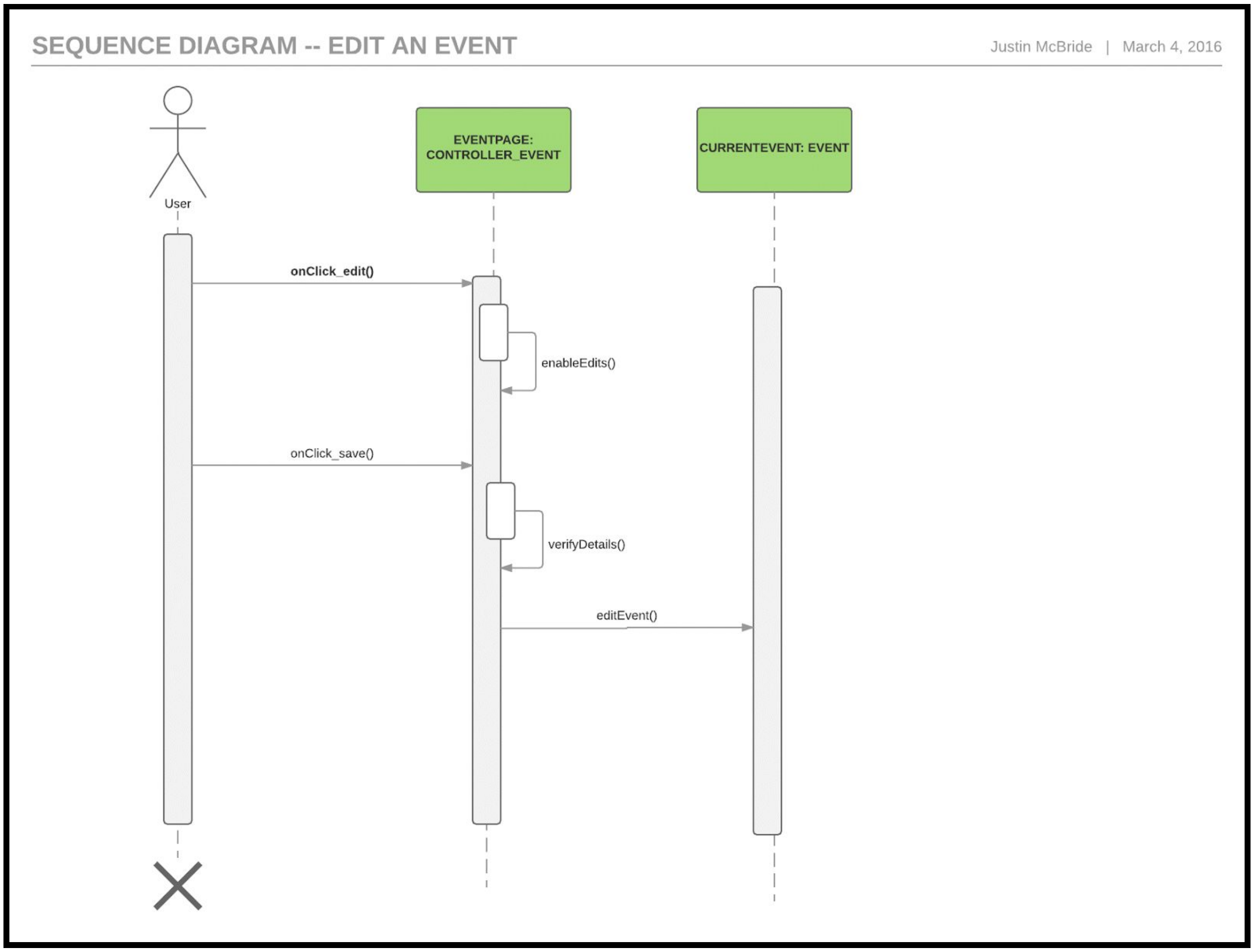
Event Search Sequence Diagram: User Requirements 14, 15, and 16. Use Case 4: Looking at events near user. Users want to be able to find events nearby to their current location and view their details. Implemented by Warren Ferrell.



Report Event Sequence Diagram: User Requirements 18 and 8. Use Case 13: Report Event. Users want to be able to report an event for inappropriate content and have it removed from the public calendar. Implemented by Justin Schiller.


SEQUENCE DIAGRAM: UC-07: Creating a group
Zachary Lamb | March 9, 2016
User
MyGroup: Controller_Group_Meta
onClick_createGroup()
isValidName()
<<create>>
MyGroup: Group
name = getname()
<<create>>
MyGroup: Controller_Group

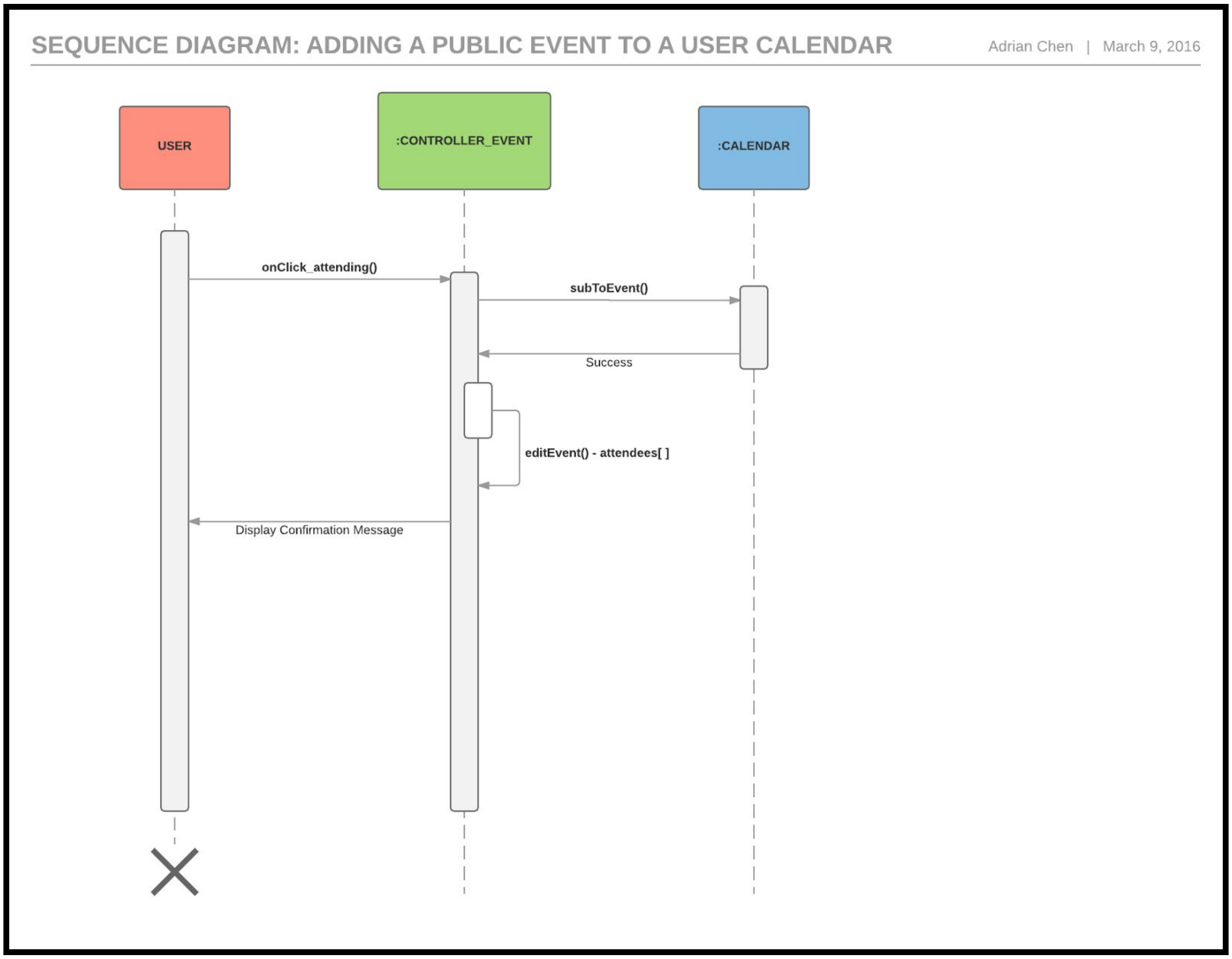
Create A Group Sequence Diagram: User Requirements 12 and Functional Requirement 03. Use Case 07: Create A Group. User wants to create a group and add other users to it. Implemented by Zachary Lamb.



SEQUENCE DIAGRAM -- EDIT AN EVENT

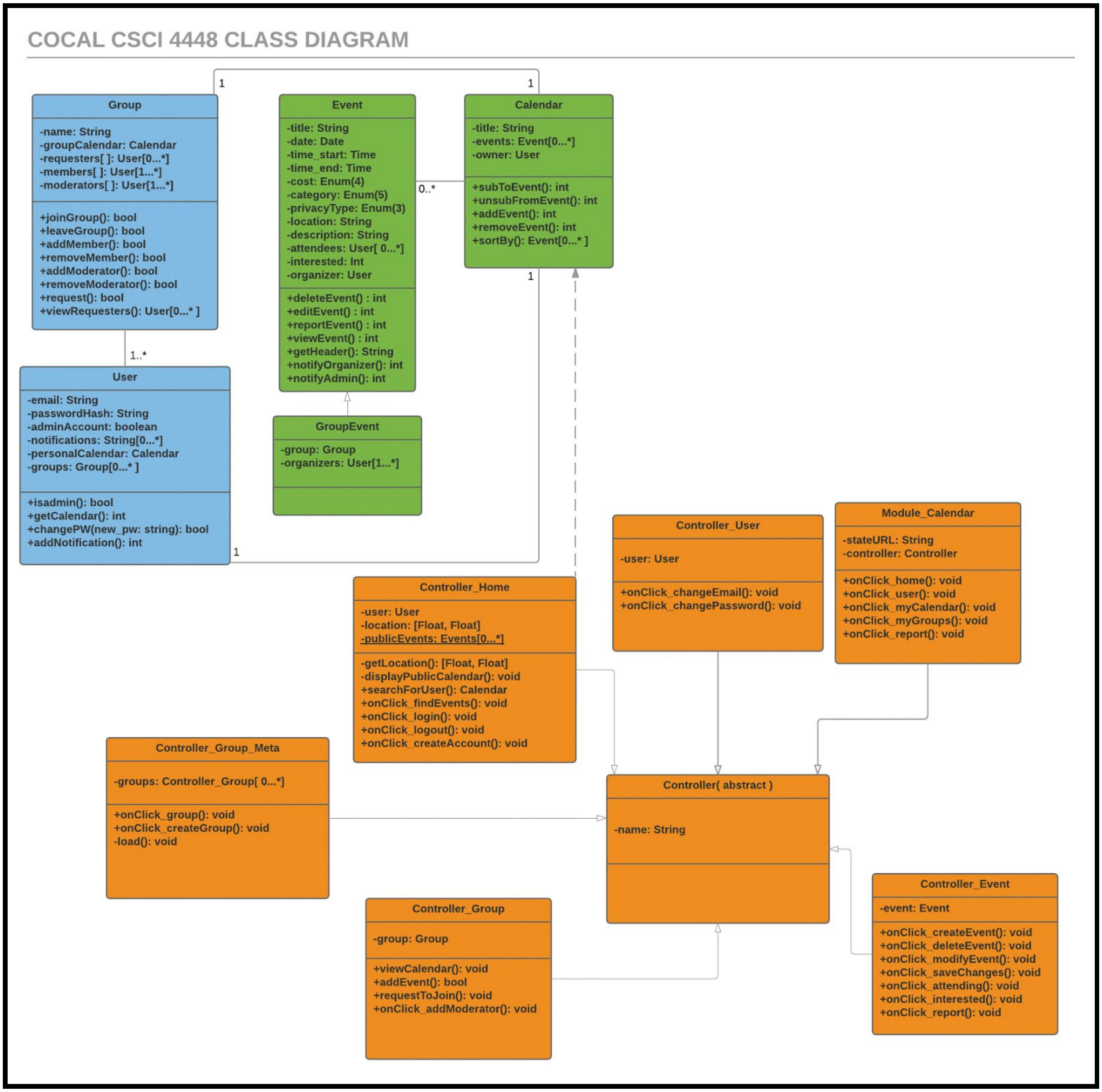
Justin McBride | March 4, 2016

Edit Event Sequence Diagram: User Requirement 21. Use Case 8: Edit Event. Users want to be able to edit an event’s details and have it reflected indefinitely. Implemented by Justin McBride.


SEQUENCE DIAGRAM: ADDING A PUBLIC EVENT TO A USER CALENDAR
Adrian Chen | March 9, 2016
USER
:CONTROLLER_EVENT
:CALENDAR
onClick_attending()
subToEvent()
Success
editEvent() - attendees[ ]
Display Confirmation Message

Adding a Public Event to a User Calendar Sequence Diagram: User Requirement 10, Use Case 11. A User wants to subscribe to an event to be able to track it within their own calendar. Implemented by Adrian Chen

Class Diagram:


COCAL CSCI 4448 CLASS DIAGRAM
Group
-name: String
-groupCalendar: Calendar
-requesters[ ]: User[0...*]
-members[ ]: User[1...*]
-moderators[ ]: User[1...*]
+joinGroup(): bool
+leaveGroup(): bool
+addMember(): bool
+removeMember(): bool
+addModerator(): bool
+removeModerator(): bool
+request(): bool
+viewRequesters(): User[0...*]
1
1..*
User
-email: String
-passwordHash: String
-adminAccount: boolean
-notifications: String[0...*]
-personalCalendar: Calendar
-groups: Group[0...*]
+isadmin(): bool
+getCalendar(): int
+changePW(new_pw: string): bool
+addNotification(): int
1
Event
-title: String
-date: Date
-time_start: Time
-time_end: Time
-cost: Enum(4)
-category: Enum(5)
-privacyType: Enum(3)
-location: String
-description: String
-attendees: User[0...*]
-interested: Int
-organizer: User
+deleteEvent() : int
+editEvent() : int
+reportEvent() : int
+viewEvent() : int
+getHeader(): String
+notifyOrganizer(): int
+notifyAdmin(): int
0..*
GroupEvent
-group: Group
-organizers: User[1...*]
Calendar
-title: String
-events: Event[0...*]
-owner: User
+subToEvent(): int
+unsubFromEvent(): int
+addEvent(): int
+removeEvent(): int
+sortBy(): Event[0...*]
1
1
Controller_User
-user: User
+onClick_changeEmail(): void
+onClick_changePassword(): void
Module_Calendar
-stateURL: String
-controller: Controller
+onClick_home(): void
+onClick_user(): void
+onClick_myCalendar(): void
+onClick_myGroups(): void
+onClick_report(): void
Controller_Home
-user: User
-location: [Float, Float]
-publicEvents: Events[0...*]
-getLocation(): [Float, Float]
-displayPublicCalendar(): void
+searchForUser(): Calendar
+onClick_findEvents(): void
+onClick_login(): void
+onClick_logout(): void
+onClick_createAccount(): void
Controller_Group_Meta
-groups: Controller_Group[0...*]
+onClick_group(): void
+onClick_createGroup(): void
-load(): void
Controller(abstract)
-name: String
Controller_Event
-event: Event
+onClick_createEvent(): void
+onClick_deleteEvent(): void
+onClick_modifyEvent(): void
+onClick_saveChanges(): void
+onClick_attending(): void
+onClick_interested(): void
+onClick_report(): void
Controller_Group
-group: Group
+viewCalendar(): void
+addEvent(): bool
+requestToJoin(): void
+onClick_addModerator(): void