Proximity

Project Report 1

Group 16: Ryan Wortmann (Leader), Song Vu Nguyen, Ryan Rottmann, Nathan Kulczak, John Oatey

Group Website: https://github.com/Rdubya54/Software_Engineering_Project

Individual Contributions Breakdown

******Each team member has contributed equally in the planning of this project and the drafting of reports parts 1-3.

Table of Contents

1) Customer Statement of Requirements

(a)Problem Statement

There are many different social media platforms, most of which implement some form of networking, messaging, and posting. These platforms let people talk to each other, share life events, share personal information, find communities for the like-minded or for those they share interests with, and many other features. However, while these platforms let us talk with friends and family online, they do not encourage face to face interactions. These types of platforms do not offer a realistic view of the lives of users to those that they have connected with. Whether this is through user's curating their content to control their public image, like Instagram, or the platform using uniquely tailored algorithms to decide what content they deem most relevant, like Facebook or Twitter.

Our project is to create a new type of social media platform that will address these issues and provide its users with a better representation of the local community. In our social media platform the focus will be moved away from forming networks with other users and moved towards geographically based communities. Posts will be flagged internally with approximate locations so that users can be shown posts from other users near them. This will have many benefits over traditional social media. It will show its users a representative view of their local communities than they could otherwise never see. This will in turn help foster a more tight knit online community as its users will be able to connect with others with many shared life experiences. Connecting locals online can also help people grow closer at already existing communities, such as college campuses.

There are many issues that need to be addressed when developers are making a social media platform. All users expect a number of standard features to be implemented well and easy to use. These features include the ability to post media or text, commenting on other user's posts, direct messaging, and much more. Developers of the social media platform must ensure that these features behave as a user would expect them too from their experience with other platforms. Most of these features, and most others that will be implemented, will be enhanced by only showing users local content. A lot of content users on popular social media platforms are shown is largely irrelevant to their lives unless they have gone through the effort of tailoring the users they have networked in or the groups they have joined to be relevant to them. Posts made near a user have a much higher chance of being relevant information. For example, someone could post about a street being shut down or a good new restaurant in town. People visiting from out of town could get a good feel for the local atmosphere or if there's an expected influx of out-of-towners, such as during a football game, locals put post helpful advice.

The way to measure the overall success of a social media platform is, at least for a high level approach, quite simple. A social media platform is successful if it has an active user base. An active user base means that not only does the platform have a steady or increasing number of users, but the users that are signed up are posting content and interacting with each other.

There are other metrics that our platform will be judged by. The user client must be responsive with a well designed user interface. The backend server must implement an algorithm that decides which posts to serve to which users so that users find the posts interesting, helpful, or relevant in some way. The developers must keep all private data private by using industry standards such as using TLS to secure all communication between the client and the server. Stored confidential information must also be kept secure. For example, industry standard hashing algorithms for passwords or using an industry standard identity provider service such as Cognito through AWS or Active Directory B2C through Azure Cloud to handle user accounts.

(b)Glossary of Terms

Technical Terms:

Progressive Web App - Web applications that load like regular web pages or websites but can offer the user functionality such as working offline, push notifications, and device hardware access traditionally available only to native mobile applications.

Database - A large table where user information will be stored.

Geolocation - The process or technique of identifying the geographical location of a person or device by means of digital information processed via the Internet.

Algorithm- A process or set of rules to be followed

Transport Layer Security (**TLS**) – Cryptographic protocol that provides communications security over a computer network.

Amazon **Cognito** - An Amazon Web Services (**AWS**) product that controls user authentication and access for mobile applications on internet-connected devices.

Non Technical Terms:

Posts - These are things that the users share with everyone on the app. These could be text, pictures, or location.

Social Media - Websites and applications that enable users to create and share content or to participate in social networking.

Users - The people who have user accounts with the application

Interaction - An exchange between two or more individuals.

Friend - A user who is connected with a given user in some way in the app. This usually means that they see each other’s posts.

News feed - A data format used for providing users with frequently updated content

2. System Requirements (User Stories)

(a)Functional Requirements

Identifier	User Story	Size Points
ST-1	As an user, I can create an account by providing a username, password and other necessary information	5
ST-2	As an user, I can login to my account by using my registered username and password	5
ST-3	As a logged user, I can share my thought by posting it online as text or media	5
ST-4	As a login user, I can view and comment on other people posts	4
ST-5	As a login user, I can choose which information I want to share to friend or the public	4
ST-6	As a logged in user, I can add people to be my friends	3
ST-7	As a logged in user, I can send messages to people that I added as friend	3
ST-8	As a user, I can create public or private groups for other users to join	2
ST-9	As a user, I can join public or private groups	3
ST-10	As a member of a group, I can post on the group’s page	3
ST-11	As a member of a group, I can view and comment on posts made by other group members on the group’s page	2
ST-12	As the creator of a group, I can remove posts from the groups page	2

ST-13	As the creator of a group, I can remove members of a group and prevent them from rejoining	2

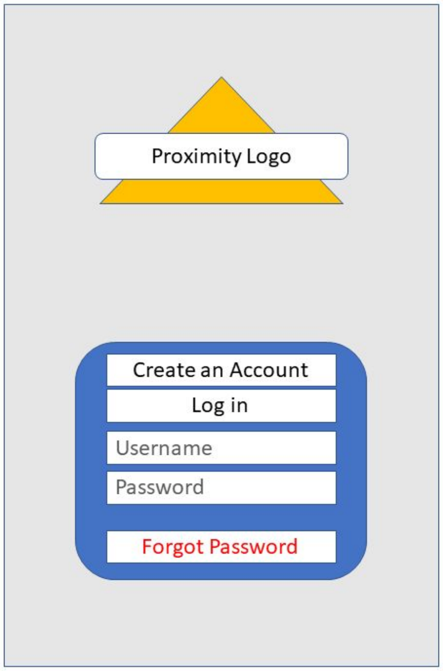
(b)Nonfunctional Requirements

Identifier	User Story	Size Points
ST- 14	As a user, I can easily navigate the web app and find what I want quickly	3
ST- 15	As a user, I can view a post as soon as it was posted by people	2
ST- 16	As a user, I can expect that my personal information are store privately and securely	5
ST- 17	As a user, I can expect the web app's map is giving me pertinent data about events that happen near me	5

(c)On-Screen Appearance Requirements

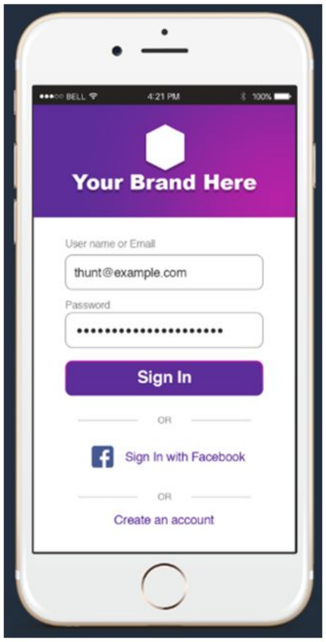
Identifier	User Story	Size Points
ST-18	As a user, the there should be a profile page that lets me customize some information about myself as well as have a profile picture of me	3
ST-19	As a user, the app should have a map where my general geolocation is presented as well as other information	2
ST-20	As a user, the app should have a page dedicated to messaging other users	3

Log in Page



Create Account Page

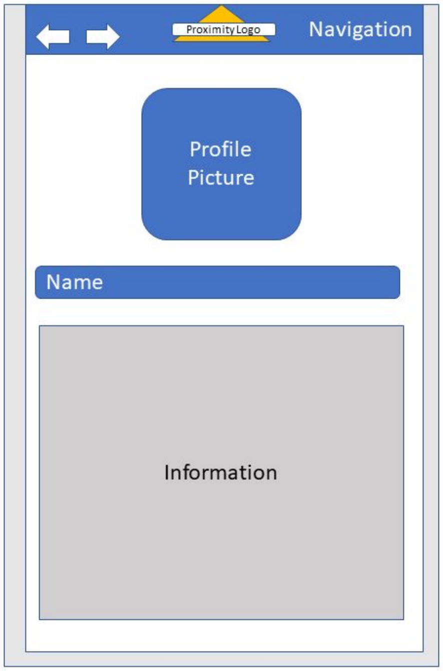
We plan on linking to Amazon Cognito services to create and manage user accounts for our Web App. This is the sample image Amazon provides to show how what this page looks like



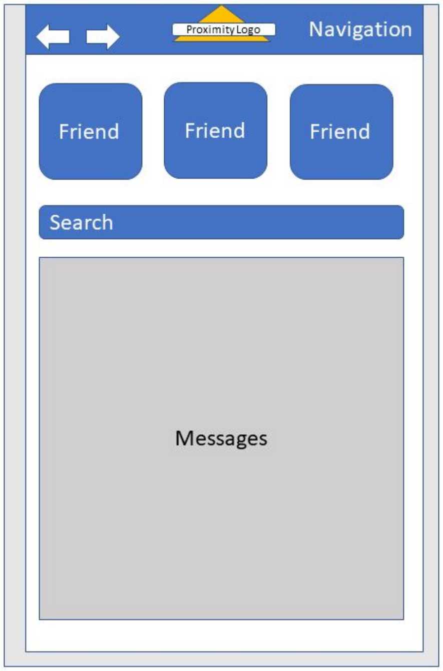
User name or Email: thunt@example.com

Password: ••••••••••••••••••••

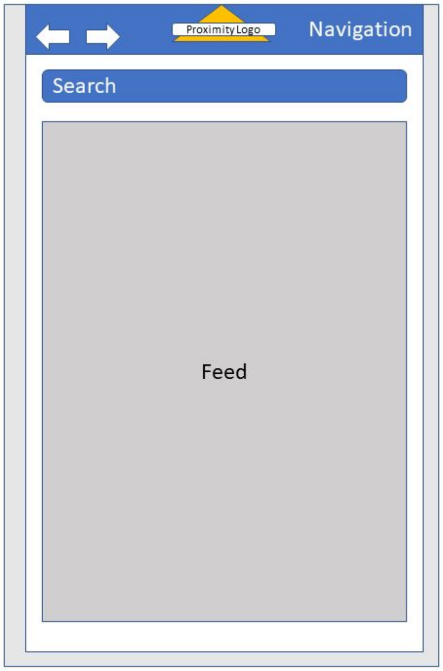
Profile Page



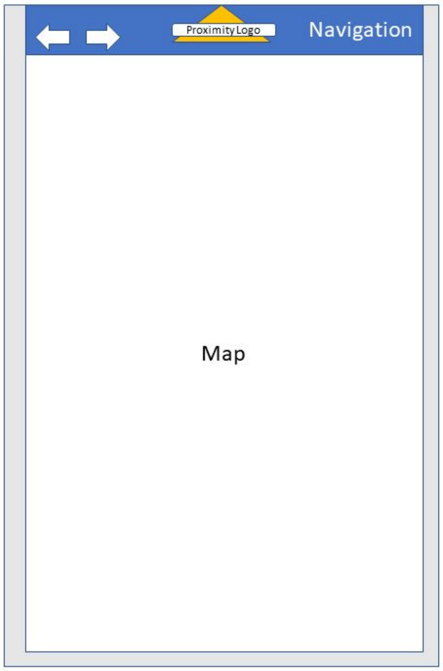
Messages



Feed


Proximity Logo
Navigation
Search
Feed

Maps


Proximity Logo
Navigation
Map

3) Functional Requirement Specification

(a)Stakeholders

Our stakeholders include anyone who is interested in using the social media platform. Social media platforms with an active user base have quite a few different parties who would be likely to use it.

These include:

- Individuals who want to make a user account and use the platform to connect with other people in their area, friends, and local groups of users.
- A business who wants to use the platform to advertise to local users. The nature of the platform means that businesses can make themselves visible to local users easily just by tagging their location for events they may be hosting.
- An organization who wants to reach out to potential new members. This could be any sort of organization such as a volunteer group, clubs, teams, or anything else.

(b)Actors and Goals

Initiating Actors

- **User**
    - Role - A human using the app.
    - Goal - Connect with friends via messaging, posts, event sharing, and location sharing.
- **Group**
    - Role - A private environment created by users to share information only within the bounds of a group.

    - Goal - Let the group connect with each other via messaging, posts, event sharing, and location sharing.

Participating Actors

- **Database**
    - Role - The object that stores information needed for the system.
    - Goal - Store and modify information related to users, groups, and events.

(c)Use Cases

Casual Description

Use Case 1: A user can create and sign in to a user account. Creating an account is requirement in order for user to access our webapp. All user logins are through Amazon Cognito and user informations is store privately in a secure database. User stories cover: ST-1, ST-2, ST-14, ST-16, ST-18.

Use Case 2: A user can add other users as 'friends' to their account. User then can choose to share their post privately only to friends or view their friends posts. User stories cover: ST-5, ST-6, ST-7, ST-14

Use Case 3: A user can contact other users. User can use either direct message for friends, posts or comments for general public, or group message to people that in a group. User stories cover: ST-3, ST-7, ST-11, ST-14, ST-20

Use Case 4: A user can create and manage a group of users. User can sent invitation to other users to join their group and do group activity. User stories cover: ST-8, ST-9, ST-10, ST-11, ST-12, ST-13, ST-14

Use Case 5: A user can join and interact with other users in the group. User can choose to joined either public or private group and communicate with other people in the group via messaging or posts. User stories cover: ST-9, ST-10, ST-11, ST-14

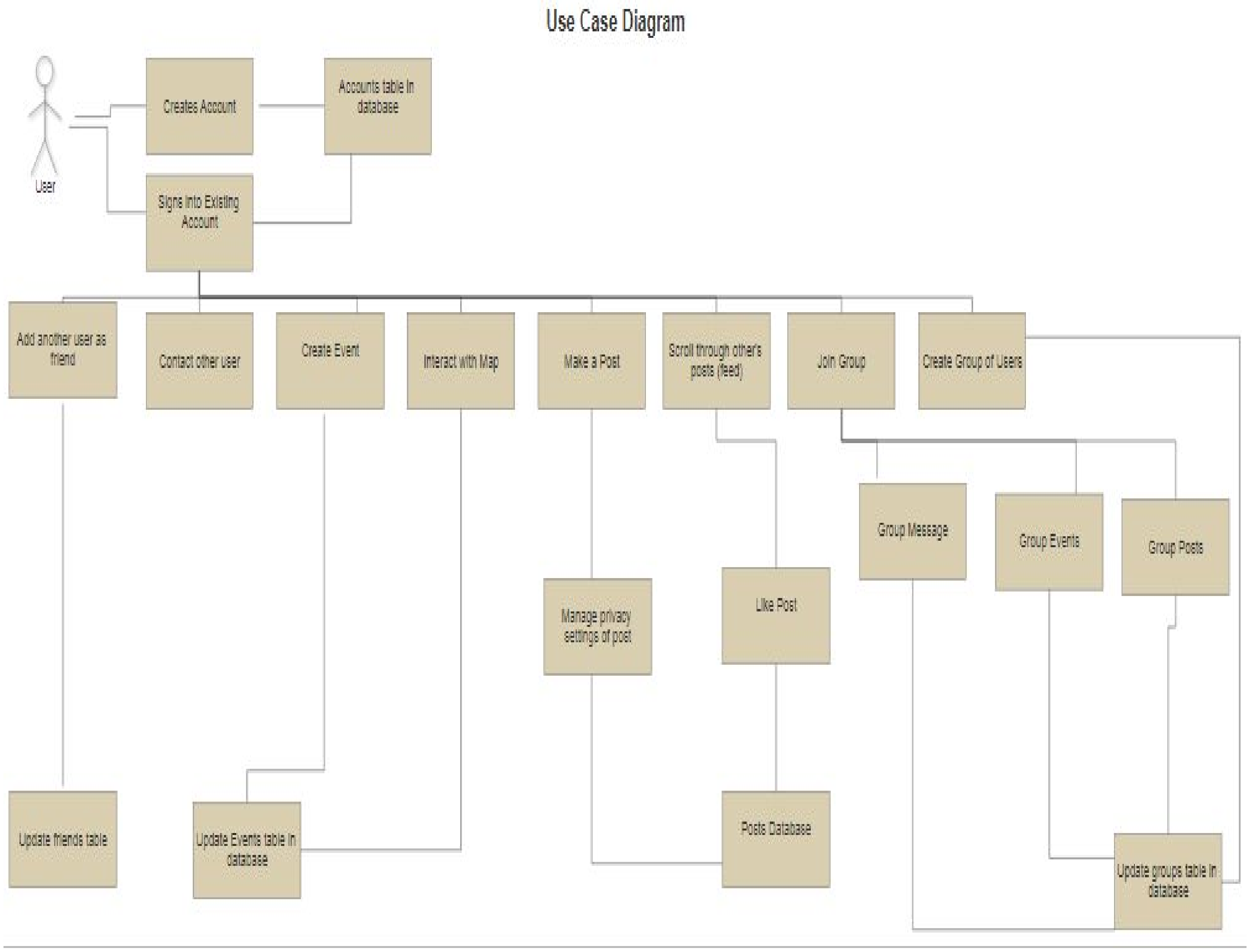
Use Case 6: A user can create and manage a post that will be viewable to others of their selection. User can have text, media in their posts and they can choose to share it with general public, their friends or group. User stories cover: ST-3, ST-5, ST-10, ST-11, ST-14

Use Case 7: Users can view and interact with other user’s posts on a feed. User can like, comment and reply to comment on the post. User stories cover: ST-4, ST-10, ST-11, ST-12, ST-14

Use Case 8: A user can create and manage an event that is visible on the map to other users. User can choose to host an event and posted its location on the map to share with other users nearby who interested. User stories cover: ST-14, ST-19

Use Case 9: A user can look at the map and interact with events that other users have created. User can view events events that are posted near their location. By clicking on that event on the map, it will give the user more detail on the event like what kind of event is this, who invited and and what time does it start etc. User stories cover: ST-4, ST-14, ST-15, ST-17, ST-19

Use Case Diagram:


Use Case Diagram
User
Creates Account
Accounts table in database
Signs into Existing Account
Add another user as friend
Contact other user
Create Event
Interact with Map
Make a Post
Scroll through other's posts (feed)
Join Group
Create Group of Users
Group Message
Group Events
Group Posts
Manage privacy settings of post
Like Post
Update friends table
Update Events table in database
Posts Database
Update groups table in database

Traceability Matrix

User Stories	Size Points	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8	UC9
ST-1	5	X								
ST-2	5	X								
ST-3	5			X			X			
ST-4	4							X		X
ST-5	4		X				X			
ST-6	3		X							
ST-7	3		X	X						
ST-8	2				X					
ST-9	3				X	X				
ST-10	3				X	X	X	X		
ST-11	2			X	X	X	X	X		
ST-12	2				X			X		
ST-13	2				X					
ST-14	3	X	X	X	X	X	X	X	X	X
ST-15	2									X
ST-16	5	X								
ST-17	5									X
ST-18	3	X								
ST-19	2								X	X
ST-20	3			X						
Max Size Points		5	4	5	3	3	5	4	3	5
Total Size Points		21	13	15	17	11	17	14	5	16

Fully-Dressed Description

Use Case UC-1:	**User Account**
Related Requirements:	**Trace Matrix**
Initiating Actor:	General User
Actor’s Goal	To have a permanent account in application
Participating Actors:	Database, WebApp System
Preconditions:	1. No database entry for user
Postconditions:	Compete database entry for user User can manage their data
Flow of Events for Main Success Scenario:	

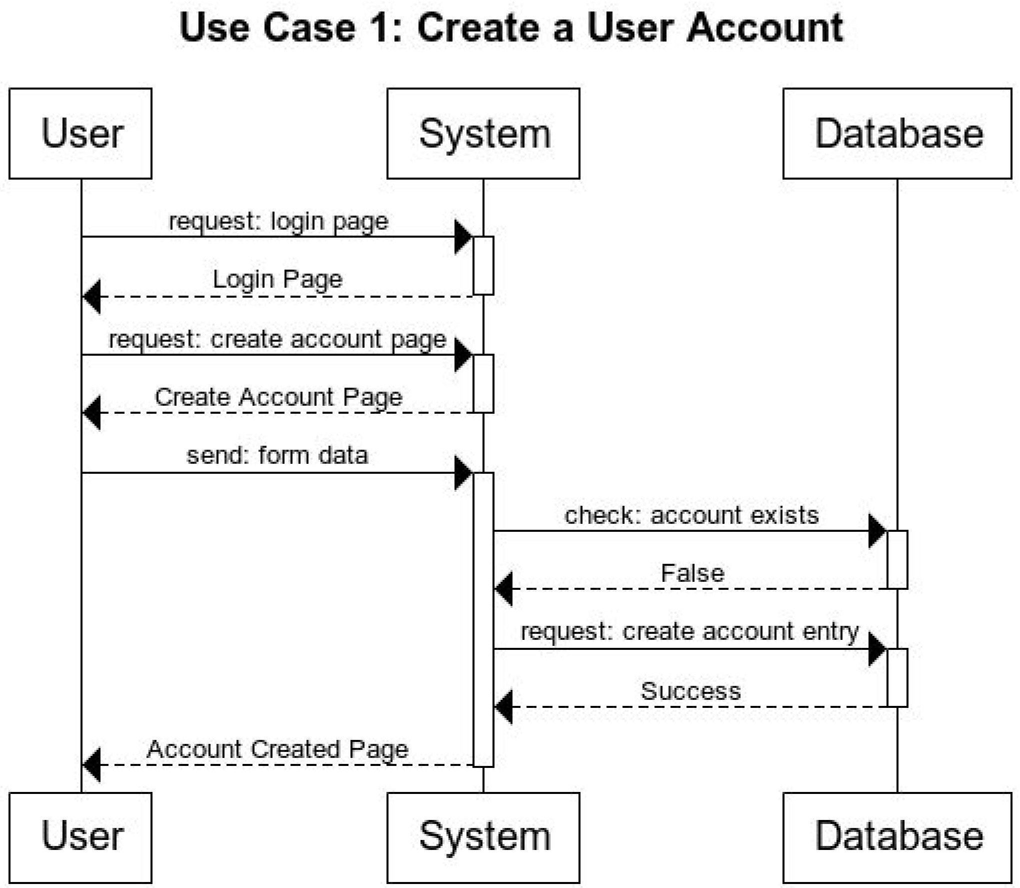
→	**1)**	**General User** loads application to see login page
→	**2)**	**General User** clicks create an account
←	**3)**	**System** displays form for **General User** to fill out
→	**4a)**	**General User** fills out form and submits data
←	**5a)**	**System** returns a success for account create page as well as creating a database entry for new **General User**
Alternative flow of Events		
→	**4b)**	**General User** fills out form and submits email tied to another account
←	**5b)**	**System** returns a failure for account create page due to email being found in the database already

Use Case UC-4:	**Group Creation and Management**
Related Requirements:	**Trace Matrix**
Initiating Actor:	General User
Actor’s Goal	To be able create and manage a group of other users

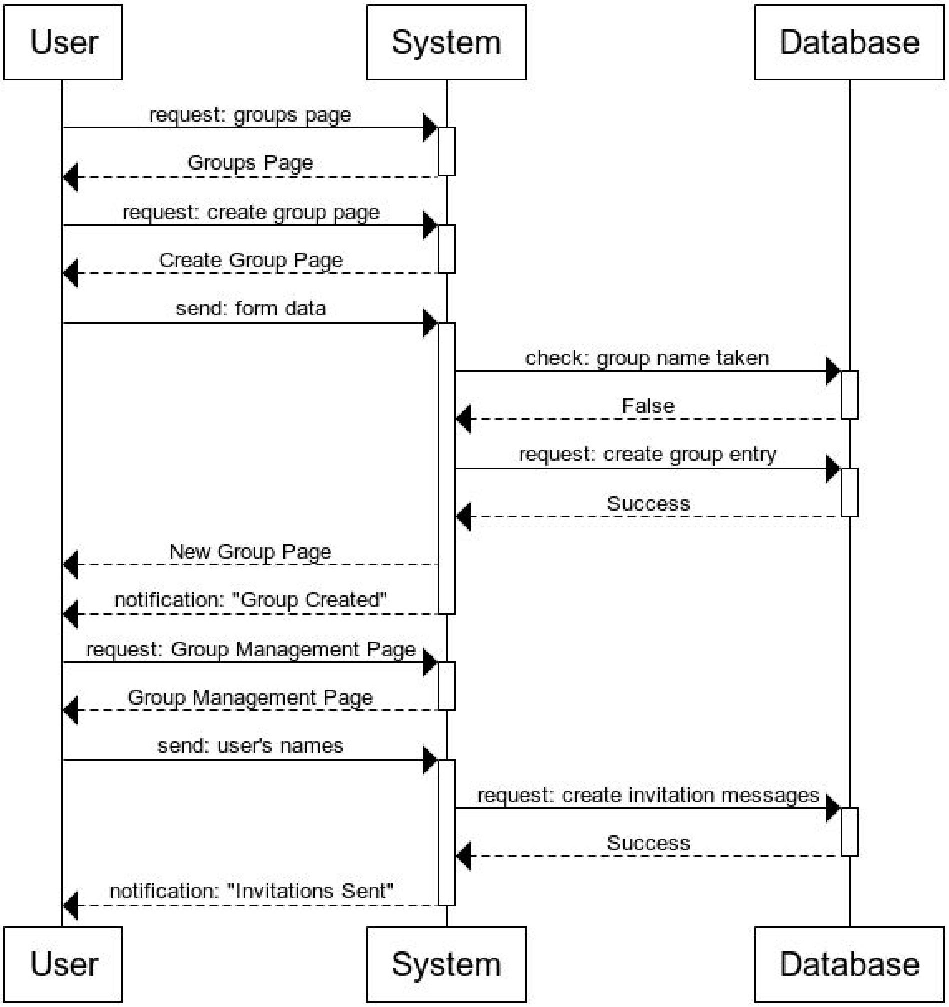
Participating Actors:		Webapp System, database
Preconditions:		1. The user doesn’t have any data about belonging to a group in the database
Postconditions:		1. The user has data in the database that identify them with a group and link them with other users that belong to the same group 2. System enables user to have various actions to interact with their group
Flow of Events for Main Success Scenario:		
→	**1)**	**General User** select the group option on the web app menu
→	**2)**	**General User** clicks to create new group
←	**3)**	**System** displays Create Group form for **General User** to fill out.
→	**4)**	**General User** fills out form and submits data
←	**5a)**	**System** checks the database and verifies that no groups with this name exists. Then add this newly created group to the database
←	**6a)**	**System** displays the new group page and a notification alerting the **General User** that the group has been created.
→	**7)**	**General User** clicks the “Manage Group Members” button.
←	**8)**	**System** displays the Manage Group Members page.
→	**9)**	**General User** types in the names of other users to add to the group.
←	**10)**	**System** adds invitations for these users to the message database.
←	**11)**	**System** displays a notification alerting the **General User** that the invitations have been sent.
Alternative flow of Events		
←	**5b)**	**System** checks the database and sees that a group with this name already exists.
←	**6b)**	**System** displays an error alerting the **General User** that this group name has been taken.

Use Case UC-9:		**User Map Event**
Related Requirements:		**Trace Matrix**
Initiating Actor:		User
Actor’s Goal		To interact with an event
Participating Actors:		Event Owner
Preconditions:		1. Event Owner has created an event
Postconditions:		1. User has interacted with event 2. User can view data about the event
Flow of Events for Main Success Scenario:		
→	**1)**	**User** loads up map page
←	**2)**	**System** displays map and events based on **Users** location
→	**2)**	**User** selects an event
←	**3)**	**System** displays data about event (time, location, who is going)
→	**4a)**	**User** decides they want to attend by selected “going”
←	**5a)**	**System** tells **Event Owner** that **User** is attending
Alternative flow of Events		
→	**4b)**	**User** picks not attending
←	**5b)**	**System** will not let **Event Owner** know as well as hiding event from **User**

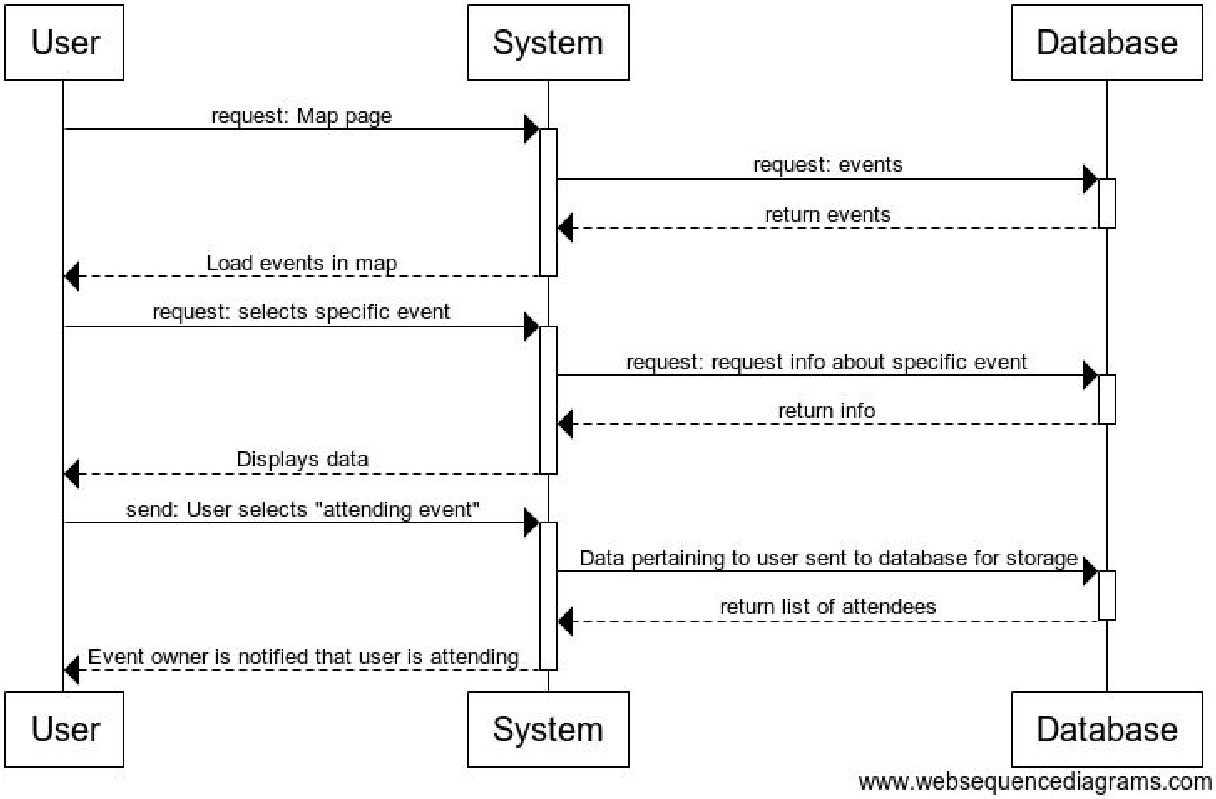
(d)System Sequence Diagrams



Use Case 4: Create and Manage a Group



Use Case 9: User can view map and interact with events that other users have created.

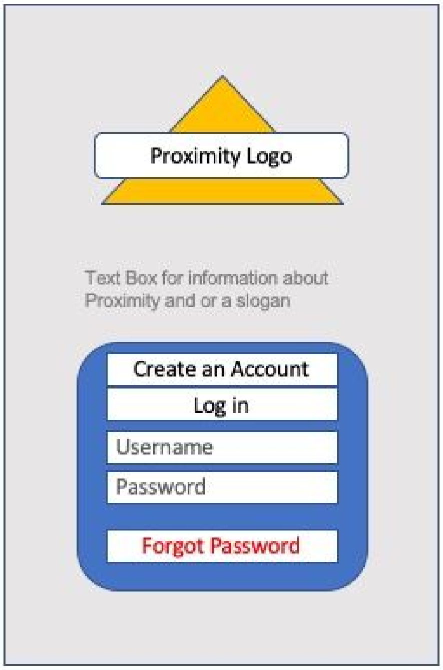

User
System
Database
request: Map page
request: events
return events
Load events in map
request: selects specific event
request: request info about specific event
return info
Displays data
send: User selects "attending event"
Data pertaining to user sent to database for storage
return list of attendees
Event owner is notified that user is attending
User
System
Database
www.websequencediagrams.com

4) User Interface Specification

(a)Preliminary Design

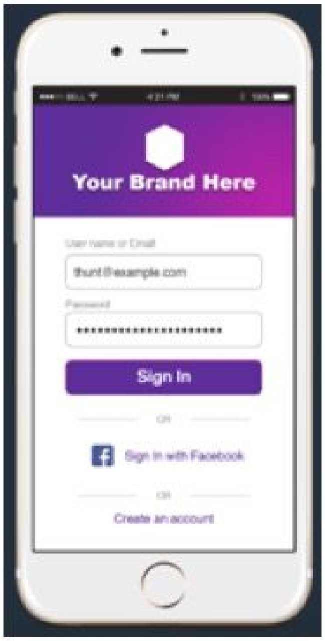
Log in Page

This is the first page that the user will be presented with when visiting our page. This will be where current users can log into our Web App. New users will be able to create an account for our app by pressing the Create an Account button. When users press the create an account button they will be directed to the create an account page.



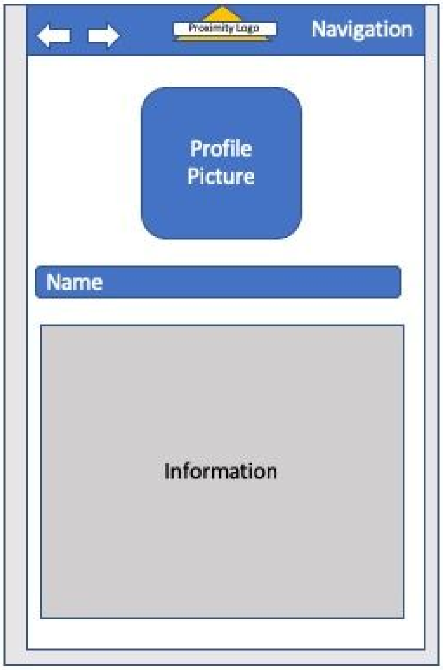
Create Account Page

We plan on linking to Amazon Cognito services to create and manage user accounts for our Web App. We chose to use amazon to handle our user account personal data because we believe that Amazon is the most secure way to store users personal information. This is the sample image Amazon provides to show how what the log in page looks Like.



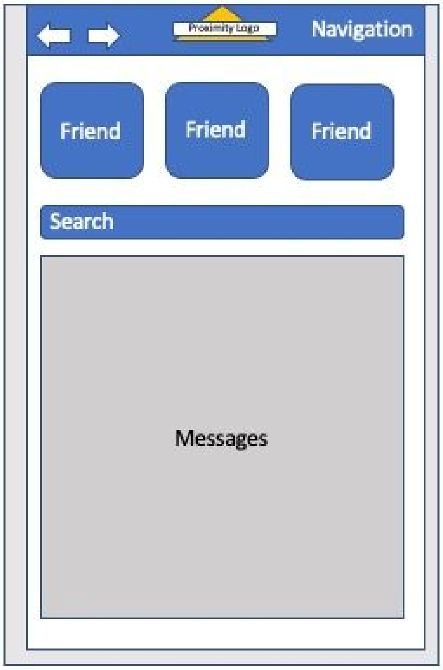
Profile Page

This is where the user will store a profile picture as well as personal information about themselves including email phone number etc. The user will be prompted for this information after they first create an account on the app. This information will be viewable to other users using the app. A user can decide which personal information other users can see depending on if they are in the same group or if they are friends.



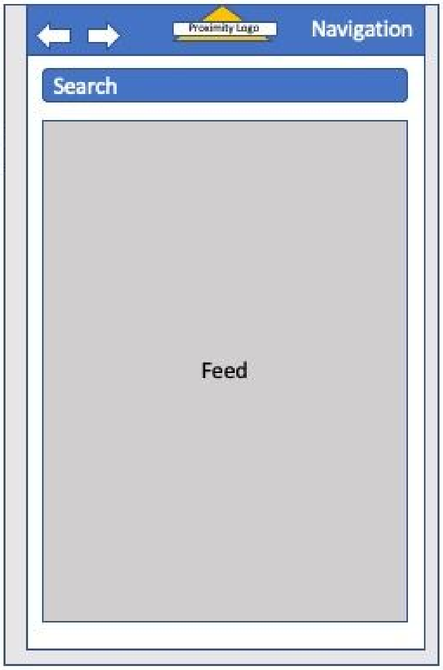
Messages

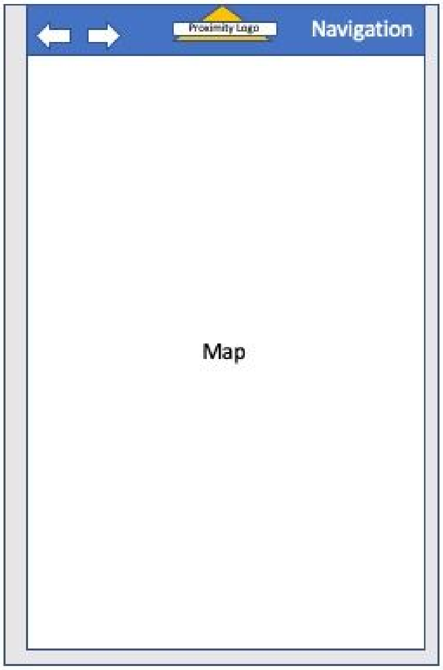
This will be the page that users can message other users on. Users can message other individual users on this page or they can message groups of people also.



Feed

This is where a user can view other users posts. The posts will appear on this page and a user can scroll down through other users posts indefinitely until there are no more posts to scroll through. A user can also search through their posts and filter the results by person.





Maps
This is where users will be able to view a map of their own posts as well as others posts that were chosen by the users to include post location. This will also be where users can view events created by users.

(b)User Effort Estimation

Use Case 1: A user can create and sign in to a user account. To sign in a user will need to fill in the username and password text boxes on the entry page to our app. This involves the user pressing into both of the text boxes, typing their username and password in, and pressing the login button. There are 3 button presses the user will need to make along with key presses to type their username and password.

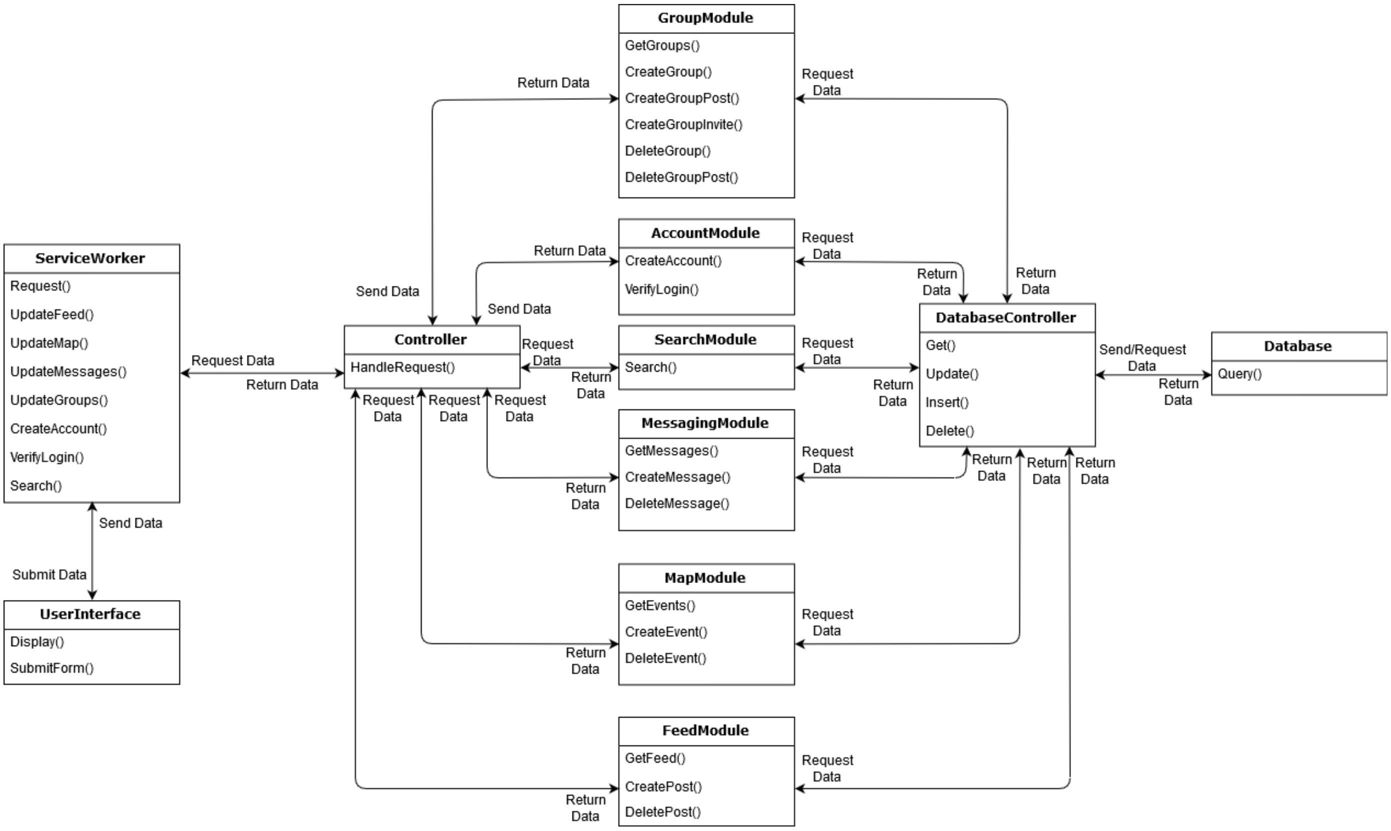
Use Case 4: A user can create and manage a group of users. To create a group the user first selects a friend this can be done by searching their name in the messages page and then selecting their profile. Once selected there will be a button where the user will be able to click add user to group. This will open all of your current groups which you can add the user to or you will be able to create and name a new group. There mouse presses a user needs to do is open search bar, search individual, select individual, add to group, and then select the group to add to.

Use Case 7: Users can view and interact with other user's posts on a feed. A user can view other users posts by navigating to the feed page. Once on the page the user can simply scroll through and select a post for further information about the post. This takes only one click to retrieve this information with the addition of a user scrolling through the feed

Use Case 9: A user can look at the map and interact with events that other users have created. The user can view events events that are posted near their location. By clicking on that event on the map, it will give the user more detail on the event like what kind of event is this, who invited and and what time it starts. It only takes one click to open information about the events if it is already visible on the map. If the event is not visible on the map the user can zoom out and move the map to find the event. This will add additional clicks for the user.

5) Domain Analysis

(a)Domain Model



Domain Model Diagram

- Concepts Definition

Responsibility	Type	Concept
R1: Accepts and handles request from the front end the application.	D	Controller
R2: Fetches new information from the back end server and updates the user interface.	D	ServiceWorker
R3: Formats data received from the ServiceWorker and displays it to the user. Accepts input from the	D	UserInterface

user and transmits it to the ServiceWorker.		
R4: Stores and provides access to data	K	Database
R6: Sits between database and the rest of the application and provides an easy interface.	D	DatabaseController
R7: Fetches data from the posts table using the DatabaseController and formats it to be sent to the user to display the user's home feed. Also uses the DatabaseController to add new post data to the database.	D	FeedModule
R8: Fetches a list of events to display on the map using the DatabaseController and formats it to be sent to the user to display the user's event map. Also uses the DatabaseController to add new event data to the database.	D	MapModule
R9: Fetches a list of messages to display using the DatabaseController and formats them to be sent to the user to display the user's messages. Also uses the DatabaseController to add new message data to the database.	D	MessagingModule
R10: Fetches group information from the DatabaseController and formats it to be sent to the user to display a group's page and new posts. Also uses the DatabaseController to add new group data to the database.	D	GroupModule
R11: Creates new account entries in the database and verifies sign ins for already existing accounts.	D	AccountModule
R12: Fetches information relating to a search query from the relevant databases using the DatabaseController and formats it to be sent to the user.	D	SearchModule

- Associate definitions

Concept pair	Association Description	Association Name
UserInterface<->ServiceWorker	The User Interface Displays information to the user that is provided by the ServiceWorker	Display

ServiceWorker<->Controller	The ServiceWorker communicates with the Controller request and send data back and forth	Request Communication
Controller<->GroupModule	The Controller handles requests from the GroupModule about updating group information	Request handling
GroupModule<->DatabaseController	The GroupModule Contacts the the DatabaseController to get or update group data in the Database	Data Interaction
Controller<->AccountModule	The Controller handles requests from the AccountModule about creating and verifying user accounts	Request handling
AccountModule<->DatabaseController	AccountModule sends create read, or delete requests to the DatabaseController	Data Interaction
Controller<->SearchModule	Controller handles requests from the SearchModule	Request handling
SearchModule<->DatabaseController	SearchModule sends create read, or delete requests to the DatabaseController	Data Interaction
Controller<->MessagingModule	Controller receives requests to send, read, and delete messages from the MessageModule	Request Handling
MessagingModule<->DatabaseController	MessageModule sends create read, or delete requests to the DatabaseController	Data Interaction
Controller<->MapModule	Controller handle request about map interaction by sending appropriate request through MapModule	Request Handling
MapModule<->DatabaseController	MapModule sends create, read, or delete requests to the DatabaseController	Data Interaction
Controller<->FeedModule	Controller handle request about feed interaction by sending appropriate request through FeedModule	Request Handling
FeedModule<->DatabaseController	FeedModule sends create, read, or delete requests to the DatabaseController	Data Interaction
DatabaseController<->Database	Database sends Queries to Update the database	Data handling

- Attribute definitions

Concept	Attribute	Attribute Description
Controller	Handle Request	Handle requests from the user by interact with appropriate module to update data in the database and update user interface
ServiceWorker	Request	Communicate user request from the interface to the controller
UserInterface	Display	Display information data to the user on screen
	SubmitForm	Send request base on user interaction on the screen through the ServiceWorker to the Controller for appropriate action
Database	Query	Modify data in the database, base on the request of the DatabaseController
DatabaseController	Get	Handle request to get and return data from the database
	Update	Handle request to update data in the database
	Insert	Handle request to insert new data into the database
	Delete	Handle request to delete data in the database
FeedModule	GetFeed	Get the post data that the user request from the database and format it to human readable format and send it back to interface to be display on screen
	CreatePost	Take user’s newly created post and added it to the database
	DeletePost	Delete the post that the user request from the database
MapModule	GetEvent	Get data about a map event from the database and send it to be display on the interface
	CreateEvent	Added data about a map event that is created by

		user into the database
	DeleteEvent	Delete data about a map event from the database
MessagingModule	GetMessage	Get data about user request messages from the database
	CreateMessage	Added user newly send message to the database
	DeleteMessage	Delete message that user requested from the database
GroupModule	GetGroups	Get data information about a group from the database
	CreateGroup	Add group data that user entered into the database
	CreateGroupPost	Add post data from a group to the database
	CreateGroupInvite	Add user's send group invitations to the database
	DeleteGroup	Remove a group data from the database
	DeleteGroupPost	Remove post data from a group from the database
AccountModule	CreateAccount	Add a newly user created account information into the database
	VerifyLogin	Check user entered login information with user's data in the database and allow user to access that information if it matched
SearchModule	Search	Check if a specific data that that the user enter is in the database

- Traceability matrix

Use Case	Size Points	Domain Concepts										
		Controller	Service Worker	UserInterface	Database	DatabaseController	FeedModule	MapModule	MessagingModule	Group Module	AccountModule	Search Module
UC-1	21	X	X	X	X	X					X	
UC-2	13	X	X	X	X	x						X

UC-3	15	X	X	X	X	X			X			
UC-4	17	X	X	X	X	X				X		
UC-5	11	X	X	X	X	x				X		
UC-6	17	X	X	X	X	X	X					
UC-7	14	X	X	X	X	X	X					
UC-8	5	X	X	X	X	X		X				
UC-9	16	X	X	X	X	X		X				

(b)System Operation Contract

Use Case 1:

- Preconditions: The application’s interface is displaying a login screen that will prompt the user to click a link if they do not have an account.
- Postconditions: The user will have a username and account in the database. The user will now be able to use the app.

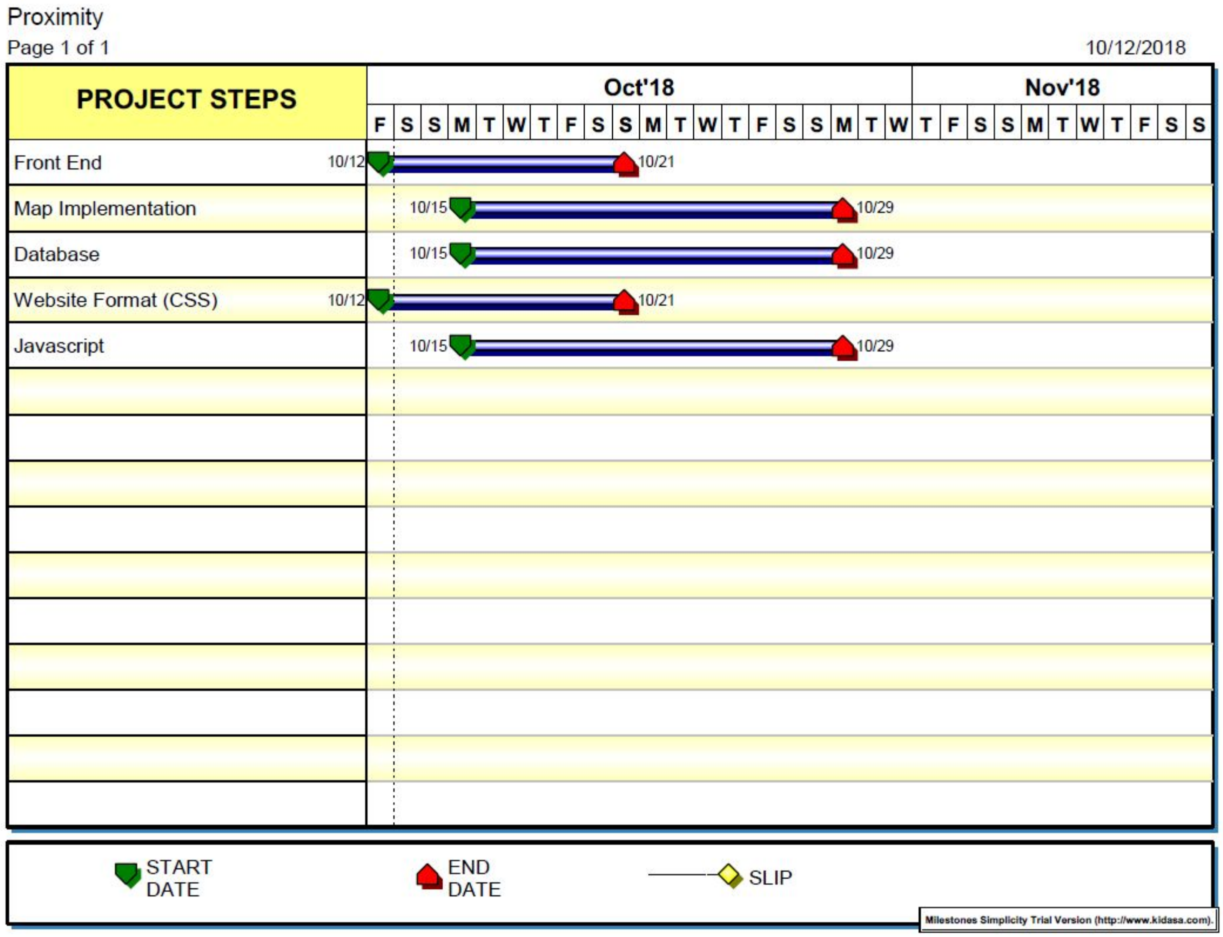
Use Case 4:

- Precondition: The application’s interface is displaying a “groups page”, listing all of the groups the user belongs to. On the same page is a create group button.
- Postcondition: New group data is added to the database. The user now can interact directly with his group or create new groups.

Use Case 9:

- Preconditions: The application’s interface is displaying a map with user created events on it.
- Postconditions: The event’s information on the database was fetched and displayed on the screen when user click on a specific event. The user can now interact with the map to find events near them.

6) Plan of Work



Plan of Work chart

So far:
So far each team member has contributed equally in the planning of this project and the drafting of reports parts 1-3.

Currently:
John is currently working on coding for the front-end this will be done 10/19. Ryan Wortmann is currently working on writing the code for the map this will be done on 10/19 as well. Nathan is currently writing code for the back end (database) this will take longer and is not expected to be done until 10/19. Ryan Rottman will likely write code for the CSS (format) of the site once John is done working on the javascript/html for the front end. Song Vu is going to assist John with the front-end where he needs extra help. What Song Vu is going to do will be determined once John

has finished the skeleton of the front-end. Before our first demo, we hope to at least have user cases 1,4, and 9 complete.

After Demo 1:

After Demo 1 is complete and we have user cases 1,4, and 9 complete, we will begin coding for user cases 2,3,5,6,7,9. Who will do what for each user case will be decided after Demo 1.

7) References

1. This Project Report 1 is made follow Lecture Slides and example Project Samples that posted on Canvas

2. The Domain Analysis section follow example of Project Samples: HealthMonitoring.pdf that posted on Canvas

3. “Software Engineering” book by Ivan Marsic
http://www.ece.rutgers.edu/~marsic/books/SE/book-SE_marsic.pdf

4. System Sequence Diagrams made using tool from:
https://www.websequencediagrams.com/

5. Use Case Diagram made using tool from:
https://www.smartdraw.com/

6. Using Amazon Cognito for User Accounts
https://aws.amazon.com/cognito/

7. Domain Model diagram made using tool from:
https://www.draw.io/

8. Milestones Simplicity 2017 for Plan of Work chart
https://kidasa.com/