Group 11 - Implementation Part II (HW7)

Casey Bennink, Tanya Haddad, Sarah Harber, Aaron Peressini, Nathan Zimmerman

Introduction: Our project is a volunteering platform for matching small neighborhood volunteering opportunities with volunteers. Features include the ability to search for volunteer opportunities, to sign up for opportunities, and post volunteer opportunities. Our client is Sean Soldberg.

The URL where our project can be tried out is here: http://flip3.engr.oregonstate.edu:7700/

Special instructions for accessing the page
In order to access the website you must be vpn'd into the OSU VPN. Information concerning the OSU VPN can be found here: http://oregonstate.edu/helpdocs/network-and-phone/virtual-private-network-vpn
Once connected enter the URL as identified above.

Using the Web Page
The homepage identifies an option for you to enter an address to search for events. Currently our database has only a few events present. To test the search functionality use the address 800 NE Oregon Street, Portland, OR, 97232 or zip code 97232 and select a range. Events in our database are within range of this address (increase the distance selected if none show up). You can test other addresses in this area as well, as long as they are within the mile radius of the events present in our database then you will get results. Click on one of the event results to be brought to an event information page.

Event Information Page
Click the volunteer button and enter in an email address to register for that event.

Navigate
Use the navbar to go to our home page, find skilled volunteer page, find host by category page, browse volunteers page, and browse event hosts page.

Find a Skilled Volunteer Page
Use the drop down to select a skill. If our database contains a volunteer with that skill then you will get a result.

Find Host by Category Page
Use the drop down to select a category. If our database contains an event host with that category then you will get a result.

Browse Volunteers Page
Click on one of the hyperlinked names to view information about that volunteer.

Browse Event Hosts Page

Click on one of the hyperlinked names to view information about that event host.

User Stories Status

Name	Tasks Checklist With Actual Time Taken	Status
View volunteer event	**Story: Users can click on a volunteer event to find out more information** • ~~Create MySQL Table of Events **(complete)** estimate: 0.5u actual: 0.25u~~ • ~~Create Queries to extract set of Event results **(complete)** estimate:1u actual: 0.25~~ • ~~Create Web Page to show list of results **(complete)** estimate:1u actual: 3u~~ • Create Event detail page to show to User estimate:1.5u actual: 2u **(complete)** **Time estimate:** 4u	**Teammates:** Nathan, Tanya **Current Status:** Implemented **Time required:** 5.5u **Relevant unit tests:** Test event DB insert, delete, and select statements for valid and invalid queries. Test web page on multiple browsers. **Problems encountered:** Event information title was not being displayed in the correct position of the webpage. Had to change the tag of the html element.
Locational Search	**Story: User can search for volunteer opportunities in their local area** • ~~Create Web Page to function as a Homepage **(complete)** estimate: 2u actual: 3u~~ • ~~Incorporate Lat Long into Event Table **(complete)** estimate: 1u actual: 0.5u~~ • ~~Create "Filter by Geography" Query **(complete)** estimate: 2u actual: 4u~~ • ~~Link Geo-Filter to Event Results Query built above **(complete)** estimate:2u actual: 1u~~ • Create Map page to view local events on a map estimate: 2u actual: 1u **(complete)** • Link Mapped events to Event Detail page built above estimate: 1u actual: 2u **(complete)** **Time estimate:** 8u -10u	**Teammates:** Nathan, Casey, Tanya, Aaron **Current Status:** Implemented **Time required:** 11.5u **Relevant unit tests:** Small distance, Medium distance, Zipcode search, Zero value from function, User enters full address into website search input, User enters zip code into website search input, User enters nothing into search input, User enters garbage into search input. **Problems encountered:** Map was not accurately zoomed when displaying data. Had to create a function to change zoom given different inputs.
Search for Skilled Volunteers	**Story: User needs a particular skill for an event, and searches for a local match** • ~~Create MySQL Table of Volunteers **(complete)** estimate: 0.5u actual: 0.5u~~ • ~~Create MySQL Table of Skills **(complete)** estimate: 0.5u actual: 0.5u~~ • ~~Link above 2 Tables in a many-to-many relationship table **(complete)** estimate: 0.5u actual: 0.5u~~ • ~~Create Skills Query **(complete)** estimate: 0.5u actual: 0.5u~~ • Create Web Page to show list of hyperlinked volunteers **(complete)** estimate: 1u actual: 2u+ • Create Volunteer detail page (if we need one?) **(complete)** estimate: 1u actual: 2u	**Teammates:** Aaron, Casey, Sarah, Nathan **Current Status:** Implemented **Time Required:** 4u **Relevant unit tests:** Insert, delete, select from both volunteer and skills table. Check for error handling when incorrect information is attempted to be inserted into the tables. Linking volunteers with events to ensure proper display of volunteer scheduled events on detail page. **Problems encountered:** No major problems were encountered.

	Time estimate: 4u	
Volunteer Event Registration	**Story: User signs up to volunteer at events of interest** ● Create Add "volunteer" button to Event detail page **(complete)** estimate: 0.5u actual:0.5u ● Create many-to-many relationship table between volunteers and events **(complete)** estimate: 0.5u actual: 0.5u ● Volunteer button adds relationship to above table **(complete)** estimate: 1u actual:1.5u **(complete)** **Time estimate:** 2u	**Teammates:** Nathan, Casey, Sarah **Current Status:** Implemented **Time required:** 2.5u **Relevant unit tests:** Insert, select, delete from volunteer and event tables. Query database to return only events for which a particular volunteer is registered. Checking event detail page for proper functionality of "volunteer" button. **Problems encountered:** Event eid was not being carried over properly from previous page. Had to change html form data. Registering multiple times and entering repeat information was not being handled correctly by the queries. Wrote queries to check for duplicate entries before additional queries.
Category Search	**Story: Select category (e.g. programming) to find event hosts hosting events that match user interest** ● Create MySQL Table of Categories **(complete)** estimate: 0.5u actual: 0.5u ● Add Category ID attribute as a foreign key to the Event Host table **(complete)** estimate: 0.5u actual: 0.5u ● Create category query **(complete)** estimate: 0.5u actual: 0.5u ● Create web page to show list of hyperlinked event hosts **(complete)** estimate: 1u actual: 1u ● Create event host detail page **(complete)** estimate: 1u actual: 1u **Time estimate:** 3.5u	**Teammates:** Aaron, Tanya, Nathan **Current Status:** Implemented **Time Required:** 3.5u **Relevant unit tests:** Database query for event host with a particular category. Insert, delete, select from both host and category tables. Checking proper handling of no results. Checking event host detail page for proper display of attributes or lack thereof. **Problems encountered:** No major problems were encountered.
Browse Results	**Story: User Views Event details** ● Create page to display search query results **(complete)** estimate: 1u actual: 1u ● Use google map API to insert map **(complete)** estimate: 2u actual: 2u ● List hyperlinked results on page after map **(complete)** estimate: 1u actual: 1.5u **Time estimate:** 4u	**Teammates:** Nathan, Aaron **Current Status:** Implemented **Time Required:** 4.5u **Relevant unit tests:** Input location close to events, input faraway location. Add events to the database, remove events. Add events that are far apart from each other. Click on search results and click on map results. **Problems encountered:** Map results were not displaying tags. Had to change the variable being used to a different one. Also, list results were not generating the proper form request. Updated html to fix.
Browse Volunteers	**Story: Hosts can view user profiles that match desired skills** ● Create query to get attribute information of volunteer **(complete)** estimate: 1u actual: 1.5u ● Create page to display attribute information **(complete)** estimate: 1.5u actual: 1.5u	**Teammates:** Aaron, Tanya, Nathan **Current Status:** Implemented **Time Required:** 3u **Relevant unit tests:** Database query for all volunteers. Select different volunteers

	Time estimate: 2.5u	from browse page to ensure the proper volunteer information page is displayed. **Problems encountered:** No major problems were encountered.
Browse Hosts	**Story: Users can view hosts that post events in their area of interest** ● Create query to get attribute information of event manager **(complete)** estimate: 0.5u actual: 0.5u ● Create page to display attribute information **(complete)** estimate: 1u actual: 1u **Time estimate:** 1.5u	**Teammates:** Aaron, Tanya, Nathan **Current Status:** Implemented **Time Required:** 1.5u **Relevant unit tests:** Database query for all hosts. Select different hosts from browse page to ensure the proper host information page is displayed. **Problems encountered:** No major problems were encountered.
Post Event	**Story: Event host posts event details with location and list of required skills** ● Create page with forms to enter in event information **(complete)** estimate: 2u actual: 5u ● Check if user is registered estimate: 0.5u actual: ● Create a new Event in the Event table with an associated Event Host ID added as a foreign key estimate: 1u actual: **Time estimate:** 3.5u	**Teammates:** Sarah, Casey **Current Status:** Partially Implemented **Tasks Left:** Check if user is registered, Create a new Event in the Event Table with an associated Event Host ID added as a foreign key **Time Required:** 5.0u+ **Relevant unit tests:** Add event with no name, complete name, no location, full location. Add duplicate events. Add different events with duplicate attributes testing duplicates of each attribute separately. **Problems encountered:** Group did not properly allocate resources to complete this user story.
Delete Event	**Story: Hosts can take down events that are cancelled or not happening** ● Create delete button on my events page estimate: 0.5u actual: ● Create query to delete event from Event Table estimate: 1u actual: ● Create success page to read and display query result estimate: 1u actual: **Time estimate:** 2.5u	**Teammates:** **Current Status:** Not Implemented **Tasks Left:** Create delete button on my events page, Create query to delete event from Event Table, Create success page to read and display query result **Time Required:** N/A **Relevant unit tests:** N/A **Problems encountered:** Group did not properly allocate resources to complete this user story.
Subscribe	**Story: User can sign up for notification of events of interest** ● Under registered user page create an enable notifications 0.5u button. ● Create notify users function that queries the Volunteer Table when a new Event is posted. Query should check location and a notification attribute set to true. 2u ● Create function to send email to list of users returned by query 2u	**Teammates:** **Current Status:** Not Implemented **Time Required:** **Relevant unit tests:** **Problems encountered:**

	Time estimate: 4.5u	
Follow	**Story: User can follow event hosts that they like or that frequently post in their area of interest** • Create a Following many-to-many relationship table linking the Volunteer Table with the Event Manager table 1u • Add query to return a list of volunteers who have notifications enabled and who are following this event manager 1u • Include result in the function to email notification to volunteers 1u **Time estimate:** 3u	**Teammates:** **Current Status:** Not Implemented **Time Required:** **Relevant unit tests:** **Problems encountered:**
Thank Volunteers	**Story: Hosts are notified of volunteers that should be thanked for participating in successful events** • Create "Event has Ended" addition to Event Detail Page estimate: 0.5u actual: • Create query of the host and volunteers who contributed at a specific event estimate: 0.5u actual: • Send Email To Host with email addresses of volunteers to thank. estimate: 0.5u actual: • Add Display of above list to Event Detail Pages for events that have ended estimate: 1u actual: **Time estimate:** 2.5u **Time required:**	**Teammates:** **Current Status:** Not Implemented **Relevant unit tests:**. **Problems encountered:**
Register Host	**Story: User can register to post events that need volunteers** • Create register page with form fields to enter data 2u • Create query to add user to Event Manager table 0.5u **Time estimate:** 2.5u	**Teammates:** **Current Status:** Not Implemented **Time Required:** **Relevant unit tests:** **Problems encountered:**
Register Volunteer	**Story: User can register for website using social media profile** • Research how social media logins work **(complete)** estimate: 1u actual: 1u • Determine what needs to be done from there estimate: 1u actual: • Create MYSQL Table to store usernames & passwords estimate: 0.5u actual: • Create page to display registration form estimate: 0.5u actual: • Upon Submission, check to make sure user is not already in the system. If they are not registered add information to MYSQL Table, otherwise display error message. estimate: 0.5u actual:	**Teammates:** Casey, Sarah, Aaron **Current Status:** Not Implemented, Deprioritized based on customer feedback **Time Required:** **Relevant unit tests:**

	Time estimate: 3.5u **Time required: 1u + remaining tasks**	

UML Diagram Discussion

User Story: Volunteer Event Registration
The UML diagram acted as a general outline for the requirements. While useful as a general guideline, the diagram was a pretty straight forward description of the process and was likely not necessary for development. The main difficulty came from the implementation of these steps. The outline was a helpful reminder of the steps needed to accomplish the story, but it was far from necessary.

User Story: Register Volunteers
Based on customer feedback, we deprioritized the Register Volunteers user story and reprioritized and adjusted other user stories, some of which did not have a UML diagram. While unused in terms of development, the UML diagram for Register Volunteers was helpful in making the decision to deprioritize the registration function. The technology required to properly implement a full registration database was deemed too difficult based on our limited time and resources, but the primary decision to deprioritize this task was based on the Sean’s feedback of creating a simple user experience. It seemed unnecessary to develop an entire registration system involving stored passwords and user information when an email address could instead be tied to each event. This design choice saved an excessive amount of development time and allowed for more development in critical areas.

User Story: Search for Skilled Volunteers
Last week we intended to focus on implementing the Register Volunteer story before finalizing the implementation for skilled volunteers. After a discussion with the customer, we pivoted and dropped development on Register Volunteers and instead utilized a dummy database to illustrate the implementation of Searching for Skilled Volunteers. The UML diagram continued to aid in this implementation, although much of the development had already been accomplished in the previous week.

User Story: Locational Search
The UML diagram for Locational Search was less helpful this week, now that we were stuck on the implementation of a few specifics of the story rather than the general outline of the story as a whole. Again, spike would likely have been useful in the initial design phase.

Other Diagrams
Like last week, the UML diagrams we created covered most of the development work accomplished this week, although some additions to the diagrams may have facilitated the work on the incomplete user stories we had scheduled for this week. While all the UML diagrams may not have been entirely necessary, they didn’t hinder the development, except for the time it took

to create them. We again had some difficulty with how familiar we were with the technology required for implementation, so spike diagrams would have continued to aid us in the completion of those tasks.

Some expansion on the requirements in the UML diagrams may have facilitated a faster development cycle, but most of these more specific descriptions would have been difficult to anticipate prior to the start of their implementation. Perhaps reevaluating/recreating some diagrams between weeks would have been helpful, but typically once a hurdle was reached it was quickly identified; the delay was generally caused by how to overcome the difficulty rather than identifying the specifics requirements needed to accomplish the task.

We did not utilize any spike diagrams in our initial diagrams, and this was again a detriment to our development. Several stories we implemented took considerably longer than anticipated, in no small part due to some gaps in knowledge surrounding the technology involved. There was difficulty in using tools we had used in the past, both due to new implementation of those tools, but also due to time gaps from the last time we needed to use some tools. Utilizing a spike diagram to relearn several of the required technologies would have likely sped up our development and we may have accomplished more of our planned checklist.

The implementation of displaying the local events on a map and linking the mapped events to an event detail page continued to prove difficult. Utilizing the API's necessary across multiple development environments caused difficulties in consistency in testing environments. Review of several web development tools, such as implementing node.js on an external server, and some refreshers on basic bootstrap development may have saved us some time and ramped up our development speed.

Again, it is difficult to say how useful the spike diagrams would have been had they been created in advance. Some of the technological hurdles were not anticipated at all so it's difficult to say how much we could have outlined in a planned document without the experience of how we utilized the technology in the project. These shortcomings may have been discovered with a spike for the overarching technology, but may have simply been glossed over like some of the brief descriptions in the UML diagrams.

Refactoring

As with last week's refactoring, we primarily looked at the functions being used in our javascript files to ensure that they meet the criteria for simple code. Our system is a web based one so most of our code has taken the form of either javascript or html. With both our html and javascript we have sought to avoid duplicate code and overly complicated functions.

Starting with our javascript code, during our refactoring sessions we first looked over the code to check that our variable names were descriptive. There were a number of instance when our variable names did not accurately reflect the nature of the data that was being stored. For instance we changed a variable from docVal to emailVal to accurately represent what specific information was being stored in the variable. Likewise, function names were assessed and corrected if they did not accurately describe their purpose. For instance, our initMap function was initially named map. By making the function more specific we have made our code more readable. Once variable and function names were corrected, we looked for unnecessary or repetitive sections of our code. One such instance we found was a case where several sections of our xmlhttprequest code would stringify data and then parse it unnecessarily. We shortened our code and made it more precise by removing these unnecessary functions.

New areas added to our javascript code were also subject to the refactoring process. In adding new functions to our javascript file we looked to ensure that the functions were simple and direct. Anytime a function was accomplishing too many separate tasks, we would separate out the tasks into separate functions. For instance, when creating our map functions we realized that we were combining too many tasks into one large function. As a result we separated the task of initializing the map and adding markers into two separate functions. By doing so we ensured that if a task like adding a marker needed to be completed at a later time, there would be a function available to address that one specific need. Likewise by separating out the function we were able to remove the need for repetitive code. Instead of repeating the same function code over and over in our map function, we separated out the specific task and used a function call to accomplish our goal. Calling a function instead of writing out a section of code over and over again, helped to shorten the length of our code and to make it more readable.

Like our javascript, our html code was subject to a number of refactorings. Our html refactoring focused on simplifying current code by removing unnecessary tags. For example, in one instance we created a container to change the background color of a certain page. The container turned out to be problematic and did not work as anticipated. Removing the tag and editing existing tags proved to be a more effective solution. By decreasing the number of tag elements and modifying the remaining tags we were able to make our html more readable and effective. Likewise tag ids that were not descriptive enough were modified so that they accurately described the data that the tag was creating. For instance, in reusing a lot of our code to create new templates, we would have tags with ids referred to the wrong page. Updating these ids helped organize our html and css page so that any changes made would only affect the correct page.

As with last week, all of these changes, once completed, were tested against our suite of unit tests to ensure that the changes did not break any aspect of our site. For instance, modifying the html code connected to an event listener, meant that we would test the event with our array of inputs to ensure that nothing broke.

Customer Feedback

We asked Sean the following questions:

We would like to know within the 10 priority 2 items, how you would rank the tasks so that we can project how far the priority 2 list we can get?

If you prefer we implement a greater number of priority 2 items, we suggest pushing logins to priority 3 (or at least to the bottom of priority 2 rankings). Are you ok with this change of priorities?

Related to the above, if login remains a high priority, we would like to know if we can negotiate Google logins in place of Facebook logins?

Sean's responses were as follows:

Sorry for the delayed response. I probably ranked the items without thinking about realistic time constraints in this course. After working on my group's project we are realizing that everything is taking longer for us too.

Regarding the questions:

1. I changed some of the priority 2 items to "2a" and tried to take into consideration the items that would be more straightforward to implement.

2. Yes, agree to push login to priority 3. It is not necessary to understand functionality.

3. N/A

Based on Sean's willingness to cede the importance of social media logins, we agreed to push this item to priority 3 status, meaning that we would no longer attempt to fit it into the remaining time available for the project. This was a major schedule boost as it allowed us to elevate most of the priority 2 items onto the "likely will complete" portion of our predicted tasks table.

<u>Descriptions of Relevant Implementation and Integration Tests</u>

The tables below represent the results of tests focused on the interface of our various unit test modules. Each table tests the interaction of two or more modules, where each module contains various unit tests. The table identifies a specific action taken by one module and the result of that action on another module. For instance, how does the altering of a table affect the appearance of search results on the website? The modules tested will be identified above the tables, where the specific actions of one module are related to the outcome expected in another module. Website display, javascript functions, and database modules are all tested.

Test group 1: Register for Event
Test website module against database module

Test Case	Input Values	Purpose	Expected Outcome	Outcome

Click on first event and register	Click on first event in list of search results. Eid is 1.	Check that website action accurately updates the database volunteer_event table.	If volunteer does not exist new volunteer is created and added to the volunteer_event table. Eid of 1 is added.	Correct Eid added. Volunteer created and added to the correct table
Click on last event and register	Click on last event in list of search results. Eid is 3.	Check that website action accurately updates the database volunteer_event table.	If volunteer does not exist new volunteer is created and added to the volunteer_event table. Eid of 3 is added.	Error Eid 1 added instead of 3. Disconnect between web module and database. Correction: change form value from static 1 to read eid from search result query
Click on first event and register with email that does not exist	Click on first event in list of search results. Eid is 1. Register with email: test@test.com	Check that website action accurately updates the database volunteer table with a new volunteer.	Volunteer added to volunteer table with email matching website input.	Volunteer created with matching email.
Click on second event and register with email that already exists	Click on first event in list of search results. Eid is 2. Register with email: test@test.com	Check that website does not create a new volunteer, but instead registers the existing volunteer to the event.	New volunteer not created. Volunteer with email test@test.com registered to event.	Error: Volunteer not registered to event. Correction: Change query to check for existence of volunteer and then add them to volunteer_event table.
Register to same event twice	Click on second event and register with email that already exists	Check that website database does not add duplicate entry	New volunteer not created. New entry not added to volunteer_event table.	New volunteer not created. New entry not added to volunteer_event table.

Changes as a result of this test group 1
Changes were implemented to our database queries so that they handle the event of a duplicate email better. By checking first that a volunteer exists, we are able to add a user to the volunteer_event table regardless of if there email is already in the system. Additionally we also updated our queries so that they handle an attempt to register twice for the same event. Now the user will be notified of the error.

Test group 2: Find Host by Category
Test database module against website display module

Test Case	Input Values	Purpose	Expected Outcome	Outcome
Add category to database	INSERT INTO category (categoryname) VALUES (environmental);	Check that new category is displayed on the website drop down menu.	On the page Find Host by Category, environmental shows up in the drop down menu.	On the page Find Host by Category, environmental shows up in the drop down menu.
Delete category from database	DELETE FROM category WHERE categoryname=environmental;	Check that deleting item from the database is reflected on the website.	On the page Find Host by Category, environmental has been removed from the drop down menu.	On the page Find Host by Category, environmental has been removed from the drop down menu.
Add 10 categories to database	INSERT INTO category (categoryname) VALUES (environmental); etc...	The web display still accurately captures the various results in the dropdown menu.	Website display hold a robust list that is accurately displayed in the dropdown.	Website display hold a robust list that is accurately displayed in the dropdown.

Changes as a result of this test group 2
Changes for test group two focused on improving the look of the website display. We moved the drop down to make it better centered for the viewer. The actual interfacing of the two modules worked well, so no major changes were necessary.

Test group 3: Browse Volunteers
Test database against web display module

Test Case	Input Values	Purpose	Expected Outcome	Outcome
Add new volunteer with full name to database	INSERT INTO volunteer (firstname, lastname) VALUES (Fred Flintstone);	Check that update to the volunteer database is displayed on the website.	Fred Flintstone's name is displayed on the Browse Volunteers page.	Fred Flintstone's name is displayed on the Browse Volunteers page.
Add new volunteer with no name to the database	INSERT INTO volunteer (email) VALUES (frank@frank.com) ;	Check that update to the database if accurately handled by web page.	Volunteer is not displayed on the page because they do not have a name.	Volunteer is not displayed on the page because they do not have a name.
Remove volunteer with name from the database.	DELETE FROM volunteer WHERE firstname=Fred;	Check that update to the database is reflected on the website.	Fred's name is no longer displayed on the website.	Fred's name is no longer displayed on the website.

Changes as a result of this test group 3
As with the category page, the browse volunteer page resulted in changes to the look of the webpage. No major changes where needed to the implementation of the web site, as the result of the queries were accurately handled. Changes to the page focused on ensuring that the names present were large enough to be readable.

Upcoming Schedule

We are aware that the course is ending and thus there is no time remaining in our budget. However if we were continuing for another week, the following user stories and tasks would be on our schedule. Our intent would be to build all remaining unique functional aspects first (marked in green), and if time allows, tackle registration tasks, (marked in orange):

Name	Tasks Checklist With Actual Time Taken
Post Event	**Story: Event host posts event details with location and list of required skills** ● Create page with forms to enter in event information 2u ● Check if user is registered 0.5u ● Create new Event in Event Table with associated Event Manager ID added as a foreign key 1u

	Time estimate: 3.5u
Delete Event	**Story: Hosts can take down events that are cancelled or not happening** ● Create delete button on my events page 0.5u ● Create query to delete event from Event Table 1u ● Create success page to read and display query result 1u **Time estimate:** 2.5u
Subscribe	**Story: User can sign up for notification of events of interest** ● Under registered user page create an enable notifications button. 0.5u ● Create notify users function that queries the Volunteer Table when a new Event is posted. Query should check location and a notification attribute set to true. 2u ● Create function to send email to list of users returned by query 2u **Time estimate:** 4.5u
Follow	**Story: User can follow event hosts that they like or that frequently post in their area of interest** ● Create a Following many-to-many relationship table linking the Volunteer Table with the Event Manager table 1u ● Add query to return a list of volunteers who have notifications enabled and who are following this event manager 1u ● Include result in the function to email notification to volunteers 1u **Time estimate:** 3u
Thank Volunteers	**Story: Hosts are notified of volunteers that should be thanked for participating in successful events** ● Create "Event has Ended" addition to Event Detail Page 0.5u ● Create query of the host and volunteers who contributed at a specific event 0.5u ● Send Email To Host with email addresses of volunteers to thank. 0.5u ● Add Display of above list to Event Detail Pages for events that have ended 1u **Time estimate:** 2.5u
Register Host	**Story: User can register to post events that need volunteers** ● Create register page with form fields to enter data 2u ● Create query to add user to Event Manager table 0.5u **Time estimate:** 2.5u
Register Volunteer	**Story: User can register for website using social media profile** ● ~~Research how social media logins work **(complete)** 1u~~ ● ~~Determine what needs to be done from there estimate: 1u~~ ● Create MYSQL Table to store usernames & passwords** 0.5u ● Create page to display registration form 0.5u ● Upon Submission, check to make sure user is not already in the system. If they are not registered add information to MYSQL Table, otherwise display error message. 0.5u ** reflects simplification of original idea as negotiated with customer **Time estimate:** 3.5u

Group Member Contributions

Client - Sean Solberg

Sean has been very good about responding to our questions in a timely manner. We have not had a problem reaching him for this homework.

Aaron Peressini

Created volunteer_skill and category tables in MySQL. Wrote code, debugged, and performed unit testing for 'Browse Volunteers', 'Browse Hosts', 'Search for Skilled Volunteers', and 'Category Search' user stories. Contributed to the 'User Story status' portion of the written document. Helped debug other user stories. Contributed updates to MySQL queries, HTML, CSS, and JavaScript code. Available for discussions and questions throughout the week.

Nathan Zimmerman
Implemented event information page and 'register for event' functionality. Also added map results to search. Wrote unit tests and debugged code for event information page and volunteer functionality. Handled the integration tests for the site. Wrote the integration test doc and refactoring doc. Wrote the instructions for using the system. Contributed to the user story section. Contributed to MySQL queries as well. Was available for questions throughout the week.

Tanya Haddad
Organized group documents and tasks early in the week, made Github commits related to status of all user stories required under HW7, created sample data for Event and Host tables in MySQL, maintained contacts with customer, and adjusted tasks according to customer feedback. Available throughout the week for questions and discussion.

Sarah Harber
Created Event Post html page & css. Helped with report. Was available to answer all questions, and participated in group discussions & hangouts.

Casey Bennink
Wrote up assessment of diagrams and missing diagrams. Available through hangouts and participated in group discussions. Assisted with proofreading and compiling final document.

<u>References</u>

Sample spatial data created with the help of:

- Google Maps - https://maps.google.com
- Geojson.io - http://geojson.io

Sample event ideas inspired by actual volunteer activities in Portland, Oregon, hosted by organizations such as:

- Children's Book Bank - http://www.childrensbookbank.org
- Friends of Tryon Creek State Park - http://www.tryonfriends.org
- Oregon Food Bank - https://www.oregonfoodbank.org/

- Store to Door - http://www.storetodooroforegon.org
- Potluck in the Park - http://www.potluckinthepark.org