Volunteer Management System (VMS)

Amanda Priscilla Araujo da Silva
araujodasilva.a@husky.neu.edu
Northeastern University
College of Computer Science
Boston, Massachusetts 02115

Abstract
This document describes the implementation of the Volunteer Management System (VMS). This software provides the assignment of volunteers to projects for a non-profit organization. The system supports the management of volunteers and volunteer leaders. Staff members of the organization will administrate the projects and user permissions or roles, while volunteer leaders will manage the attendance to projects. The first part of this paper outlines the motivation, basic concepts involved in the system and use cases. The second part specifies the design, implementation of the system and conclusion including the overview of achieved goals.

Keywords
Volunteer, management, non-profit organization, volunteer opportunities

Introduction
Non-profit organizations are frequently looking for volunteers to reduce their costs with human resources and focus on making better use of their budget investing in the services provided to the community.

The Volunteer Management System (VMS) is an open-source web portal that allows a non-profit organization to manage its projects and volunteers. The users of the system will be the staff members of the organization, working as administrators, the volunteers and project leaders.

The motivation to develop this software started while observing the system for a non-profit organization. I noticed what the system was missing and needed to improve, like for example the functionality for volunteer leaders to choose the projects they want to lead. A second reason for choosing this topic was a personal project to spread the community services culture in my hometown where there is no similar system available.

The goal achieved in the proposed software was the ability for the staff to manage projects and volunteers, and the volunteers and leaders' ability to assign themselves to the available projects.

Requirements
This system originally proposed have the requirements of 10 Use Cases and 4 different actors. See below the description of the system requirements.

System:

Volunteer Management System

Name: Volunteer Management System **[VMS]**
Authors: Amanda Priscilla Araujo da Silva
Description: The VMS is a system for an organization that manages volunteers and projects in which they participate. The interaction between actors and the VMS is assumed to occur on an authenticated website. Activities such as login and user creation are therefore not included in the use cases. We assume every user already has an account.
Organizations: Northeastern University
Creation Date: October 15, 2015

System Actors:

General User **[User]**

Description: A user of the VMS is anyone with an account in the system. Users can only view the projects basic information. The general user needs approval from a staff member in order to pursue other roles like staff, volunteer or volunteer leader.

Staff Member **[Staff]**

Description: User that is an employee of the organization using the VMS. The Staff is responsible for managing the projects and users. A general user becomes a staff after approval from a staff member. We assume the system will have an initial staff member.
KindOf: User

Volunteer Member **[Volunteer]**

Description: User that will participate on projects as volunteer. This user is able to sign up for projects and follow up his own history of projects.
KindOf: User

Volunteer Leader **[VL]**

Description: Volunteer responsible for the execution of a project and management of volunteers' attendance. A volunteer member becomes a volunteer leader after approval of a staff member. He can still attend projects as volunteers and do everything a volunteer does.
KindOf: Volunteer

Use Cases:

UC-01: Delegate User Roles **[DelegateRoles]**

Description: The delegation of user roles will determine if the user is a staff, volunteer and/or volunteer leader. A staff member performs this task.
Step-by-step Description: 1. **[#Staff]** A staff member requests to delegate user roles; 2. **[#VMS]** The system returns the list of users with the possible roles (Volunteer, VL or Staff) listed for each user; 3. **[#Staff]** The staff adds/removes roles for the user(s) and submits changes; 4. **[#VMS]** The system performs the requested updates and informs staff the changes.

UC-02: Query Projects **[QueryProjects]**

Description: The search for a subset of projects using some query values to filter results.
Step-by-step Description: 1. **[#User]** The user submits a combination of query values. These can include project name, date interval and/or status; 2. **[#VMS]** The system returns the list of projects that match the query parameters. If no parameters defined, system will return the next projects ordered by date from today.

UC-03: Create Project **[CreateProject]**

Description: Create a new volunteer project.
Step-by-step Description: 1. **[#Staff]** Staff requests to create a new project; 2. **[#VMS]** System returns a blank form for the project data; 3. **[#Staff]** Staff fills out and submits the form; 4. **[#VMS]** System creates the project and notifies staff.

UC-04: Update Project **[UpdateProject]**

Description: The update of an existing volunteer project.
Step-by-step Description: 1. Includes: **[#QueryProjects]** 2. **[#Staff]** Selects the project to be updated; 3. **[#VMS]** The system returns a form with the project data; 4. **[#Staff]** The staff updates the fields and save changes; 5. **[#VMS]** The system performs the requested updates, informs staff member the changes and returns the list of projects with applicable updates.

UC-05: Sign up to Volunteer in a Project **[VolunteerSignUp]**

Description: Volunteers can sign up to attend a project.
Step-by-step Description: 1. Includes: **[#QueryProjects]** 2. **[#Volunteer]** Volunteer selects the project to see details; 3. **[#VMS]** The system returns details of the specific project with the option to sign up; 4. **[#Volunteer]** Volunteer requests to sign up for the project; 5. **[#VMS]** The system adds volunteer to the attendance list of the project and informs the volunteer that he is registered for the project.

UC-06: Sign up to Lead a Project **[LeaderSignUp]**

Description: Volunteer Leader (VL) can sign up to lead a project that has no volunteer leader associated with it yet.
Step-by-step Description: 1. **[#VL]** VL requests the list of projects available for leaders; 2. **[#VMS]** The system returns the list of projects the user can select to lead; 3. **[#VL]** VL selects one of the projects; 4. **[#VMS]** The system returns details of the specific project with the option to lead it; 5. **[#VL]** VL requests to lead the project; 6. **[#VMS]** The system sets the project's leader and notifies the VL.

UC-07: Quit Project **[QuitProject]**

Description: Volunteers or Volunteer Leaders (VL's) can remove themselves from a project they are going to attend or lead, respectively, prior to its execution date.
Step-by-step Description: 1. Includes: **[#QueryProjects]** 2. **[#VL/#Volunteer]** Volunteer or VL requests the project details; 3. **[#VMS]** The system returns the projects details with option to self-remove; 4. **[#VL/#Volunteer]** Volunteer or VL requests to be removed from project; 5. **[#VMS]** The system removes the user's specified association with the project, and confirms the removal to the volunteer/VL.

UC-08: View Volunteer Profile **[VolunteerProfile]**

Description: Volunteers can view their performance profile, which includes their history of projects, future projects they signed up for and total hours of community services.
Step-by-step Description: 1. **[#Volunteer]** Requests to see his performance profile; 2. **[#VMS]** The system calculates the amount of hours served, queries the projects the volunteer attended and will attend in the future and displays the results.

UC-09: View Leader Profile **[LeaderProfile]**

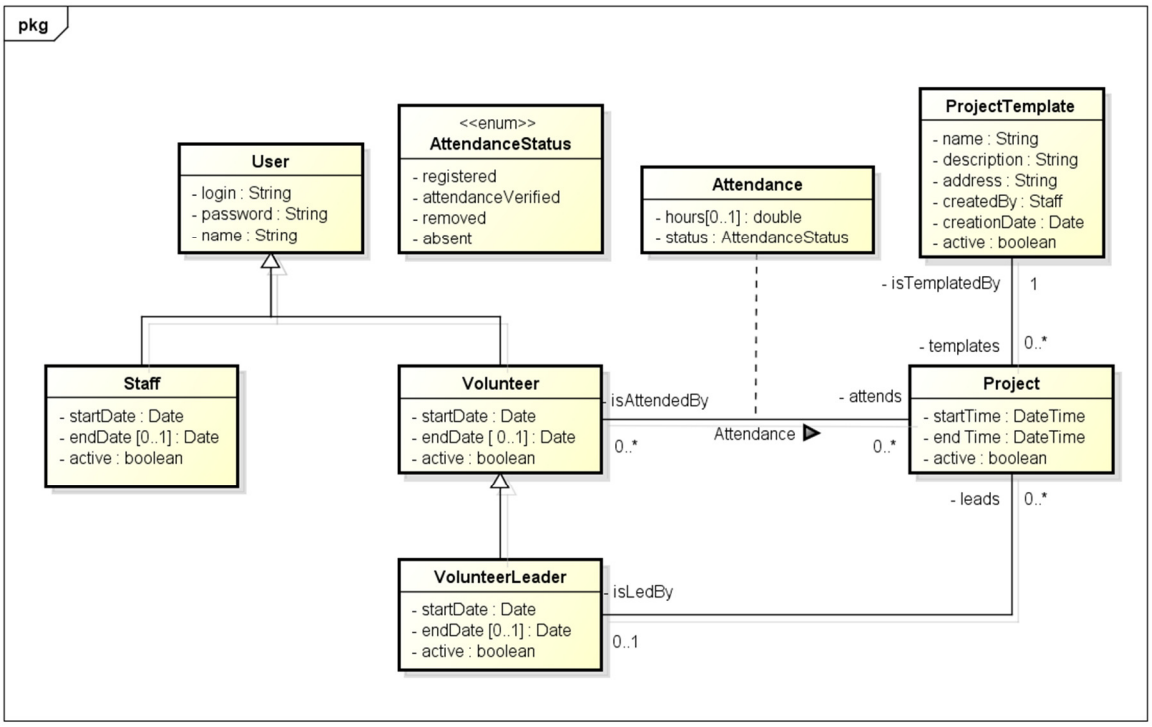
Description: Volunteer Leaders can view their leadership performance, which includes their led projects history, future projects they sign up to lead and total hours leading projects.
Step-by-step Description: 1. **[#VL]** Requests to see his performance profile; 2. **[#VMS]** The system calculates his hours leading projects, queries the led projects and the projects he signed up to lead in the future and displays the results.

UC-10: Manage Project Attendance **[ManageAttendance]**

Description: Volunteer Leader (VL) will have to confirm which volunteers were present or not in the execution of the project.
Step-by-step Description: 1. **[#VL]** VL requests to manage the projects he led; 2. **[#VMS]** The system returns the list of projects he led from the most recent to oldest; 3. **[#VL]** VL selects the project he wants to manage; 4. **[#VMS]** The system returns details of the specified project including a form with its registered volunteers and options to confirm presence or absence for each volunteer; 5. **[#VL]** VL verifies absence or presence for each volunteer and submits changes; 6. **[#VMS]** The system saves changes and informs VL about the changes.

Design

The figure below presents the Class Diagram for the software.



The users were designed as the following hierarchy:

- **User**: The most basic level of user represented by login, which later decided to be the email, and password.
- **Staff**: The administrator level of user. Staff can do anything an user can do.
- **Volunteer:** The type of user with most occurrences, since we expect to have many volunteers per project. Most of the users will be volunteers.
- **Volunteer Leader:** A subset of the volunteer type. A VL is a volunteer with leader privileges and duties. The system needs one leader per project.

Below is a brief explanation of the other classes of the system:

- **Project Template**: In the context of volunteer projects, there are situations where you have an opportunity that repeats periodically. To avoid user input for the entire project information, I decided to define the title, description and location of the project in a different class that will be associated with every project occurrence.
- **Project**: It is a project occurrence given by start and end times and a date. It also contains some common information among a group of occurrences, defined above as Project Template.
- **Attendance**: This is the association class between Volunteer and Project, where the Volunteer Leader will define the status of the attendance and for how many hours the volunteer was present in the project. The hours are usually the complete duration of the project, except when volunteer arrives late or needs to leave earlier.
- **Attendance Status:** Represents the different states of attendance:
    - Registered: user just signed up);
    - Attendance Verified: Volunteer Leader confirmed volunteer's presence
    - Removed: Volunteer left the project through the system
    - Absent: Volunteer was registered by did not attend the project in person

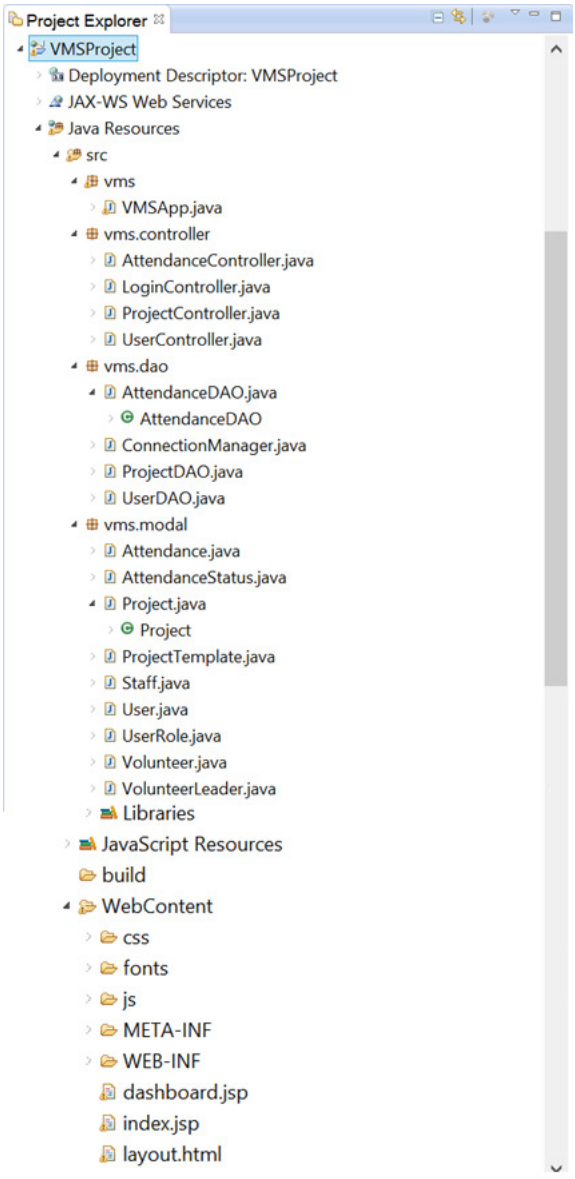
Implementation

Regarding technology, the VMS is a Web application built in Java using Java Server Pages (JSP), Servlets and Java Database Connectivity (JDBC) driver. See the table below for further details:

Technology	Version	Reference
MySQL Community Server (GPL) 5.6	5.6.26	[1]
MySQL Workbench 6.3	6.3.4	[2]
Eclipse Kepler	4.3	[3]
MySQL Connector/J	5.0.8	[4]
Java Runtime Environment 8	1.8.0_65	[5]
Apache Tomcat 7	7.0.67	[6]
Bootstrap 3	3.3.6	[7]
LayoutIt – Bootstrap interface builder	-	[8]

The figure below shows how the project was organized in Model-View-Controller (MVC) design pattern. The vms.modal package was representing the Modal layer, the Controller layer was represented by servlets, and the JSP was representing the View layer. There was also a package

dedicated for the database access, vms.dao, where there was a Connection Manager with the driver information to return a connection when needed.



The Users' inheritance (user, staff, volunteer, VL) was implemented using Joined Strategy.

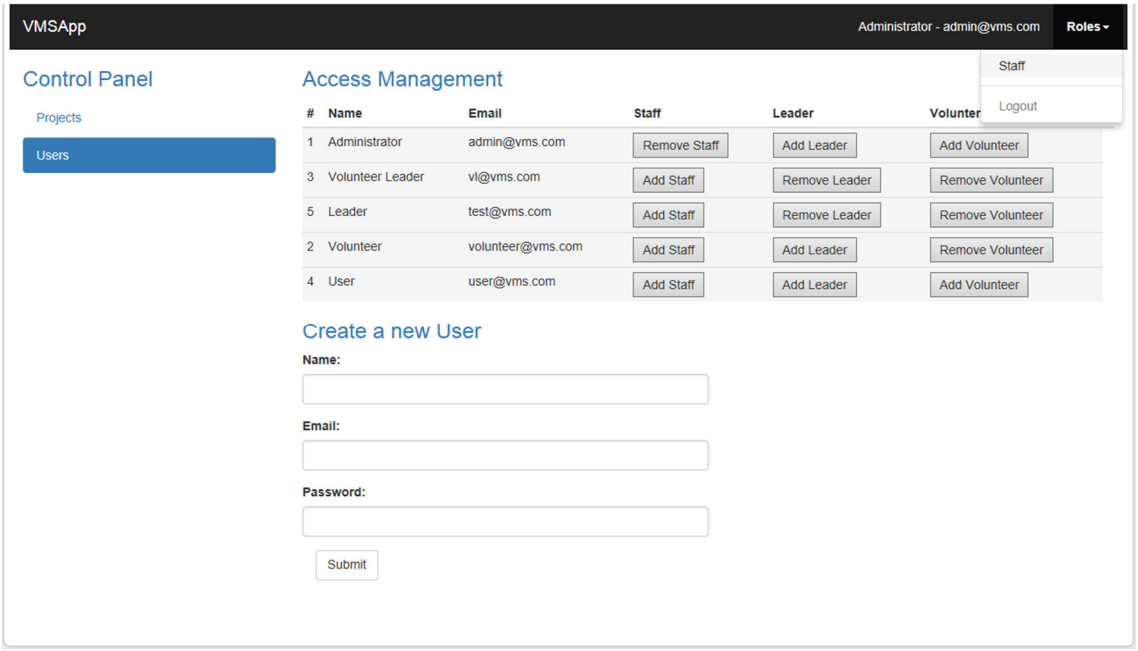
Discussion

The software was built as designed for this first version. However, during the implementation I noticed a not appropriated design decision: the users as a hierarchy or cascade inheritance. It was complicated to instantiate a new Volunteer Leader object as a property of a Project object only to set its id, which is the same as the user primary key, into the database. This generated useless work and unnecessary memory usage.

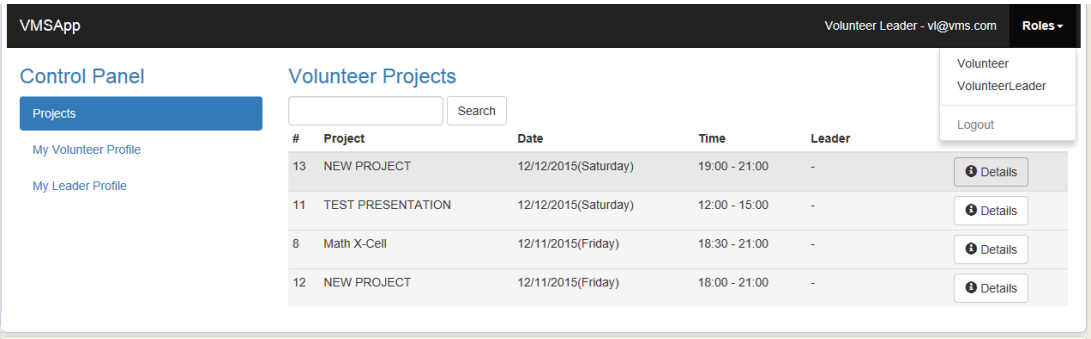
Current systems work with the concept of roles and permissions. I believe now that a new class Role with an enumerator type that could be Staff, Volunteer or Volunteer Leader with all the other attributes (start date, end date and active) should be used instead of the inheritance. The relationship between User and Role would be ManyToOne, one user could have multiple roles, but each role is for a single user with the pair (user, type) unique or as composite primary keys.

Regarding initial requirements, I decided to add new small functionalities: **create user** and **show current user roles** (Volunteer, VL and/or Staff) to facilitate the prototype's demo. The purpose was to create new users without executing extra scripts and to explain why the current user could access those functionalities without checking the database manually during the demo. The Staff Member is the only user that can perform this functionality in the page where he can view the current users to delegate permissions.

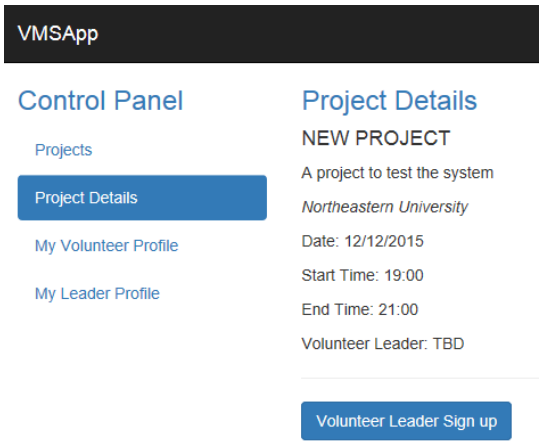
For the "**Delegate Permissions**" Use Case (UC), the Staff can assign whether a listed user is a volunteer, leader or staff. The user clicks on "Users" menu and the list of users will be displayed on the screen. The staff member can delegate more than one permission to a single user by simply clicking on the correspondent buttons in the user's row. The figure below shows the screen where the staff user (see the role in the top right of the screen) can delegate those permissions and immediately below the form to create a new user:



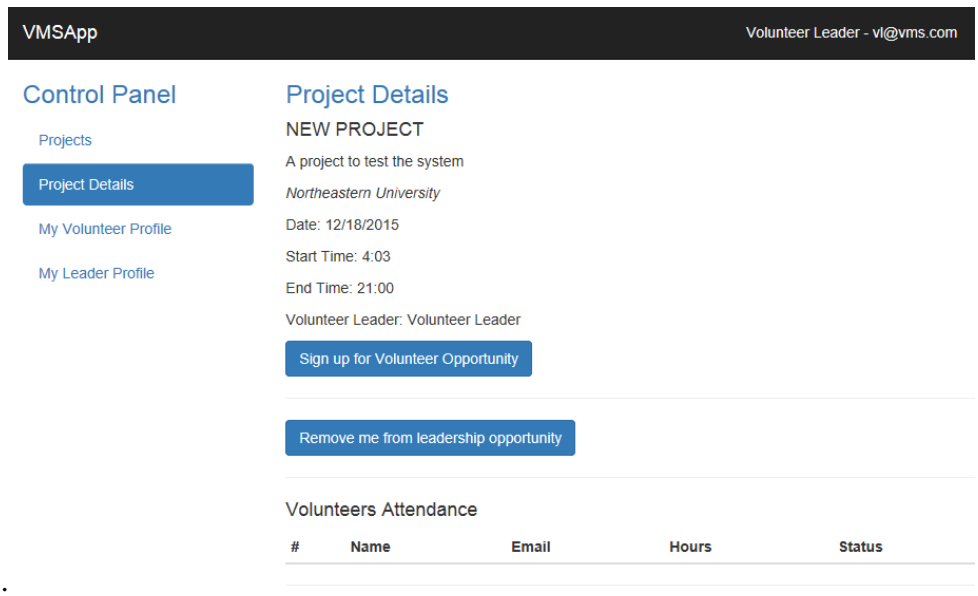
Regarding the UC to sign up and remove from a project, it was implemented in the details page of the project and is only displayed for the users with required permissions to be a volunteer and/or volunteer leader, see the permissions in the figure below on top right. To initiate the UC "**Query Projects**", a user only needs to click on "Projects" menu in the control panel. All available projects for the current user are listed as in the figure above. User can filter by typing into the text box on the top and press "Search" button.



Before the UC's for Volunteer or Leader Sign up, the user needs to select one of the listed projects and satisfy the following constraints:
1 – Leader only can sign up to lead a project if it has no leader assigned. The figure below shows the moment when a leader can sign up. Note that "Volunteer Leader: TBD". TBD stands for To Be Defined. This refers to the UC "**Sign up to Lead a Project**":

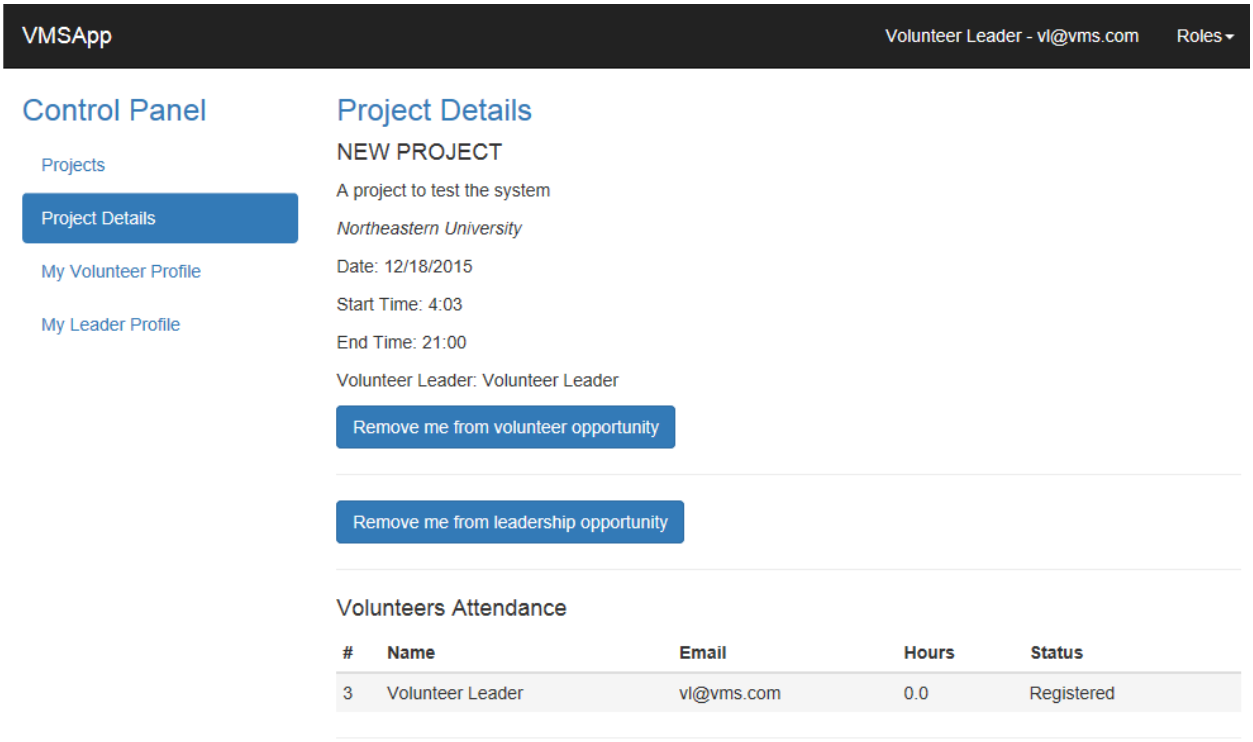


2 – Volunteer only can sign up to lead a project if there is a leader assigned.



.

The figure above is the implementation of the UC "**Sign up to volunteer in a Project**", the same project the leader signed up in step 1 now is available to be attended by volunteers because the project has a leader. In addition, since the current user is also the leader, it also shows the option to "Remove me from leadership opportunity", and after he signed up as volunteer, it also appears the option "Remove me from volunteer opportunity" which is where to trigger the UC "**Quit Project**". Right below the button to remove leader, you can see a list of volunteers attendance where the volunteers who sign up will be listed as in the figure below:



The only UC that was not fully implemented was the Verify Attendance. The update command for the database and methods in the back end were implemented, however some issues with the web project navigation increased the time spent on the other UCs, not allowing time to implement the interface for this UC specifically. However, the behavior should be the same as viewing details of a project. It opens the attendance details as a form and allow user to use a "Save" button for updates with the Status displayed as a Combo Box since it is an enumeration.

Conclusion

This project was developed to help a non-profit organization to manage the volunteers and leaders for its projects.

From the 10 initial Use Cases, 9 were successfully implemented and tested, 1 was missing to integrate with web interface in the JSP, but two new functionalities were added (create user and show current user roles) because of the need to get immediate feedbacks during the demo. Therefore, there were enough use cases implemented to validate the correct behavior and manipulation of the database, and the missing functionality for managing leaders were implemented and tested which was one of the main goals.

References

[1]	*MySQL 5.6 Reference Manual.* (2015, December 18). Retrieved from https://dev.mysql.com/doc/refman/5.6/en/
[2]	*Changes in MySQL Workbench 6.3.4* (2015, June 15). Retrieved from http://dev.mysql.com/doc/relnotes/workbench/en/wb-news-6-3-4.html
[3]	*Eclipse Kepler* (2013, June 26). Retrieved from http://www.eclipse.org/kepler/
[4]	*Changes in MySQL Connector/J 5.0.8: Documentation* (2007, October 9). Retrieved from http://dev.mysql.com/doc/relnotes/connector-j/en/news-5-0-8.html
[5]	*Java SE Runtime Environment 8 Downloads.* (n.d.). Retrieved from http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html
[6]	*Documentation Index: Apache Tomcat 7 Version 7.0.67* (2015, December 7) Retrieved from https://tomcat.apache.org/tomcat-7.0-doc/index.html
[7]	*Bootstrap 3* (n.d.). Retrieved from *http://getbootstrap.com/*
[8]	*LayoutIt – Interface Builder for Bootstrap.* (n.d.). Retrieved from http://www.layoutit.com/