VolunteerCloud: A Cloud-Based Software Application for Volunteer Recruitment

Andres Pico

Report submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science and Applications

Osman Balci, Chair

Stephen H. Edwards

Calvin J. Ribbens

May 1, 2018

Blacksburg, Virginia

*Keywords and phrases:* Software engineering, cloud software application, requirements engineering, model-view-controller architecture, client-server architecture, web services, volunteer recruitment, webchat

VolunteerCloud: A Cloud-Based Software Application for Volunteer Recruitment

Andres Pico

ABSTRACT

In the modern world tragedies still occur everywhere, there are refugee crises, homelessness is on the rise, and funding is being taken away from good-hearted programs like ‘Meals on Wheels’. For this reason, it is essential for people to be active members of the community, and to contribute to the causes they support.

However, even though the world is more interconnected than ever, there are so many different charities and organizations that it’s hard to keep up with everything out there. This results in difficulty for volunteers to find the right opportunities to contribute to sometimes. And in even much more difficulty for organizations to find volunteers and promote their events, even though plenty of them might share their cause.

VolunteerCloud offers volunteers and non-profits a way to come together by providing a solution that allows organizations to improve their volunteer attraction capabilities. In VolunteerCloud, volunteers are able to look for opportunities and subscribe to them. These activities are added to their records of volunteering activities, and are ready for exportation any time. Furthermore, organizations are able to search and identify potential volunteers when creating a new volunteering event, and then get in contact with them for recruitment purposes.

ACKNOWLEDGMENTS

I would like to thank Dr. Balci for the support and guidance he provided me during the course of this project. The invaluable information delivered through his courses helped me grow as a professional and has led me to many opportunities.

I am also thankful to my family, and all the people whom I’ve met across this journey, who have influenced me into becoming the person I am today.

**TABLE OF CONTENTS**

[ABSTRACT ii](#_Toc510953238)

[ACKNOWLEDGMENTS iii](#_Toc510953239)

[LIST OF FIGURES vi](#_Toc510953240)

[LIST OF TABLES viii](#_Toc510953241)

[LIST OF ACRONYMS ix](#_Toc510953242)

[CHAPTER 1: Introduction 1](#_Toc510953243)

[1.1 Related Work 2](#_Toc510953244)

[1.2 Statement of the Problem 2](#_Toc510953245)

[1.3 Statement of Objectives 3](#_Toc510953246)

[1.4 Overview of Report 3](#_Toc510953247)

[CHAPTER 2: VolunteerCloud Requirements Specification 4](#_Toc510953248)

[2.1 Use Case Diagrams (Check figure numbers & use of ‘will’) 4](#_Toc510953249)

[2.2 Use Cases and Functional Requirements 15](#_Toc510953250)

[2.2.1 Use Case 1: Manage Account 15](#_Toc510953251)

[2.2.1.1 Use Case 1.1 Documentation 15](#_Toc510953252)

[2.2.1.2 Functional Requirements Associated with Use Case 1.1 16](#_Toc510953253)

[2.2.2 Use Case 1.2: Update an Account 16](#_Toc510953254)

[2.2.2.1 Use Case 1.2 Documentation 16](#_Toc510953255)

[2.2.2.2 Functional Requirements Associated with Use Case 1.2 17](#_Toc510953256)

[2.2.3 Use Case 1.3: Deactivate Account 17](#_Toc510953257)

[2.2.3.1 Use Case 1.3 Documentation 18](#_Toc510953258)

[2.2.3.2 Functional Requirements Associated with Use Case 1.3 18](#_Toc510953259)

[2.2.4 Use Case 2: Search Volunteers 19](#_Toc510953260)

[2.2.4.1 Use Case 2 Documentation 19](#_Toc510953261)

[2.2.4.2 Functional Requirements Associated with Use Case 2 20](#_Toc510953262)

[2.2.5 Use Case 3: Search Organizations 20](#_Toc510953263)

[2.2.5.1 Use Case 3 Documentation 20](#_Toc510953264)

[2.2.5.2 Functional Requirements Associated with Use 3 21](#_Toc510953265)

[2.2.6 Use Case 4: Search Opportunities 21](#_Toc510953266)

[2.2.6.1 Use Case 4 Documentation 22](#_Toc510953267)

[2.2.6.2 Functional Requirements Associated with Use Case 4 23](#_Toc510953268)

[2.2.7 Use Case 5: Manage Volunteering Opportunities 23](#_Toc510953269)

[2.2.7.1 Use Case 5.1 Documentation 23](#_Toc510953270)

[2.2.7.2 Functional Requirements Associated with Use Case 5.1 25](#_Toc510953271)

[2.2.7.3 Use Case 5.2: Create Volunteering Opportunity 25](#_Toc510953272)

[2.2.7.4 Use Case 5.2 Documentation 25](#_Toc510953273)

[2.2.7.5 Functional Requirements Associated with Use Case 5.2 26](#_Toc510953274)

[2.2.7.6 Use Case 5.3: Cancel Volunteering Opportunity 26](#_Toc510953275)

[2.2.7.7 Use Case 5.3 Documentation 26](#_Toc510953276)

[2.2.7.8 Functional Requirements Associated with Use Case 5.3 27](#_Toc510953277)

[2.2.8 Use Case 6: Manage Volunteering Areas of Interests 28](#_Toc510953278)

[2.2.8.1 Use Case 6 Documentation 28](#_Toc510953279)

[2.2.8.2 Functional Requirements Associated with Use Case 6 29](#_Toc510953280)

[2.2.9 Use Case 7: View History Report 30](#_Toc510953281)

[2.2.9.1 Use Case 7 Documentation 30](#_Toc510953282)

[2.2.9.2 Functional Requirements Associated with Use Case 7 31](#_Toc510953283)

[2.2.10 Use Case 8: Linking Organization to VolunteerMatch’s Account 31](#_Toc510953284)

[2.2.10.1 Use Case 8 Documentation 31](#_Toc510953285)

[2.2.10.2 Functional Requirements Associated with Use Case 8 32](#_Toc510953286)

[2.2.11 Use Case 9: Web Chat 33](#_Toc510953287)

[2.2.11.1 Use Case 9 Documentation 33](#_Toc510953288)

[2.2.11.2 Functional Requirements Associated with Use Case 9 34](#_Toc510953289)

[2.3 Non-Functional Requirements 35](#_Toc510953290)

[CHAPTER 3: VolunteerCloud Architecture Specification 36](#_Toc510953291)

[3.1 Client-Server Architecture 36](#_Toc510953292)

[3.2 Model-View-Controller Architecture, MVC 37](#_Toc510953293)

[3.3 VolunteerCloud’s Architecture 38](#_Toc510953294)

[3.4 The Department of Defense and Architecture Framework (DoDAF) 40](#_Toc510953295)

[3.4.1 OV-2: Operational Resource Flow Description 40](#_Toc510953296)

[3.4.2 SV-1: Systems Interface Description 42](#_Toc510953297)

[3.4.3 SV-4: Systems Functionality Description 43](#_Toc510953298)

[CHAPTER 4: VolunteerCloud Design Specification 45](#_Toc510953299)

[4.1 Package Diagrams 45](#_Toc510953300)

[4.1.1 VolunteerCloud Package Diagram 45](#_Toc510953301)

[4.2 Component Diagrams 46](#_Toc510953302)

[4.2.1 VolunteerCloud Component Diagram 46](#_Toc510953303)

[4.3 Class Diagrams 47](#_Toc510953304)

[4.3.1 VolunteerCloud Class Diagram 47](#_Toc510953305)

[4.4 Statechart Diagrams 48](#_Toc510953306)

[4.4.1 Volunteer Statechart Diagram 48](#_Toc510953307)

[4.4.2 Organization Statechart Diagram 49](#_Toc510953308)

[4.5 Activity Diagrams 50](#_Toc510953309)

[4.5.1 Organization-Volunteer Activity Diagram 50](#_Toc510953310)

[4.6 Sequence Diagrams 51](#_Toc510953311)

[4.6.1 Search Volunteer Sequence Diagram 51](#_Toc510953312)

[4.6.2 Subscribe to Opportunity Sequence Diagram 52](#_Toc510953313)

[4.7 Collaboration Diagrams 53](#_Toc510953314)

[4.7.1 Search Volunteer Collaboration Diagram 53](#_Toc510953315)

[4.7.2 Subscribe to Opportunity Collaboration Diagram 54](#_Toc510953316)

[4.8 Deployment Diagrams 55](#_Toc510953317)

[4.8.1 VolunteerCloud Deployment Diagram 55](#_Toc510953318)

[CHAPTER 5: VolunteerCloud Functionality 56](#_Toc510953319)

[5.1 User Registration 56](#_Toc510953320)

[5.2 Question Repository 57](#_Toc510953322)

[5.3 Question Management 57](#_Toc510953324)

[5.4 Exam Creation 58](#_Toc510953329)

[5.5 Exam Question Management 59](#_Toc510953330)

[5.6 Publish Exam 60](#_Toc510953331)

[5.7 Report Exam 62](#_Toc510953332)

[5.8 Take Exam 62](#_Toc510953333)

[5.9 View Exam Report 64](#_Toc510953334)

[5.10 Administrator Dashboard 65](#_Toc510953335)

[CHAPTER 6: Conclusions and Future Development 67](#_Toc510953336)

[6.1 Conclusions 67](#_Toc510953337)

[6.2 Future Work 67](#_Toc510953338)

[REFERENCES 68](#_Toc510953339)

LIST OF FIGURES

[Figure 1. ARENAS Use Case Diagram 5](#_Toc440833620)

[Figure 2. Account Use Case Diagram 6](#_Toc440833621)

[Figure 3. Question Repository Use Case Diagram 7](#_Toc440833622)

[Figure 4. Exam Creation Use Case Diagram 8](#_Toc440833623)

[Figure 5. Exam Publishing Use Case Diagram 9](#_Toc440833624)

[Figure 6. Exam Report Generation Use Case Diagram 10](#_Toc440833625)

[Figure 7. Taking an Exam Use Case Diagram 11](#_Toc440833626)

[Figure 8. Exam Report Viewing Use Case Diagram 12](#_Toc440833627)

[Figure 9. Administrator Dashboard Use Case Diagram 13](#_Toc440833628)

[Figure 10. A two-tier Client-Server Architecture 34](#_Toc440833629)

[Figure 11. A multi-tiered Client-Server Architecture 35](#_Toc440833630)

[Figure 12. A Peer-to-Peer Architecture 36](#_Toc440833631)

[Figure 13. A Service-Oriented Architecture 37](#_Toc440833632)

[Figure 14. A Java EE-based Client-Server Architecture 38](#_Toc440833633)

[Figure 15. OV-2: Operational Resource Flow Description 39](#_Toc440833634)

[Figure 16. SV-4: Systems Functionality Description 40](#_Toc440833635)

[Figure 17. ARENAS Class Diagram 42](#_Toc440833636)

[Figure 18. User-Exam-Database Package Diagram 43](#_Toc440833637)

[Figure 19. Exam-Question Component Diagram 44](#_Toc440833638)

[Figure 20. Instructor Statchart Diagram 45](#_Toc440833639)

[Figure 21. Student Statchart Diagram 45](#_Toc440833640)

[Figure 22. Instructor and Student Activity Diagram 46](#_Toc440833641)

[Figure 23. Exam Creation Sequence Diagram 47](#_Toc440833642)

[Figure 24. Exam Taking Sequence Diagram 48](#_Toc440833643)

[Figure 25. Exam Creation Collaboration Diagram 48](#_Toc440833644)

[Figure 26. Exam Taking Collaboration Diagram 49](#_Toc440833645)

[Figure 27. ARENAS Deployment Diagram 50](#_Toc440833646)

[Figure 28. User Registeration Interface 51](#_Toc440833647)

[Figure 29. Question Repository Interface 52](#_Toc440833648)

[Figure 30. Question Management Interface 53](#_Toc440833649)

[Figure 31. Exam Creation Interface 54](#_Toc440833650)

[Figure 32. Exam Question Management Interface 55](#_Toc440833651)

[Figure 33. Publish Exam Interface 56](#_Toc440833652)

[Figure 34. Report Exam Interface 57](#_Toc440833653)

[Figure 35. Take Exam Interface 58](#_Toc440833654)

[Figure 36. View Exam Report Interface 59](#_Toc440833655)

[Figure 37. Administrator DashBoard Interface 60](#_Toc440833656)

LIST OF TABLES

[Table 1. Use Case 1 Documentation **Error! Bookmark not defined.**](#_Toc440833660)

[Table 2.Use Case 2 Documentation 17](#_Toc440833661)

[Table 3. Use Case 3 Documentation 19](#_Toc440833662)

[Table 4. Use Case 4 Documentation 20](#_Toc440833663)

[Table 5. Use Case 5 Documentation 22](#_Toc440833664)

[Table 6. Use Case 6 Documentation 24](#_Toc440833665)

[Table 7. Use Case 7 Documentation 25](#_Toc440833666)

[Table 8. Use Case 8 Documentation 26](#_Toc440833667)

LIST OF ACRONYMS

API Application Programming Interface

CSA Client-Server Architecture

CSS Cascading Style Sheets

EJB Enterprise JavaBean

GUI Graphical User Interface

HTML HyperText Markup Language

HTTP HyperText Transfer Protocol

JDK JAVA Development Kit

JSF JavaServer Faces

JSON JavaScript Object Notation

JSP JavaServer Pages

JRE JAVA Runtime Environment

MVC Model-View-Controller Architecture

RDBMS Relational Database Management System

RMI Remote Method Invocation

SDK Software Development Kit

SOA Service-Oriented Architecture

SQL Structured Query Language

URI Uniform Resource Identifier

URL Uniform Resource Locator

VC A Cloud-Based Software Application for Volunteer Recruitment

VolunteerCloud A Cloud-Based Software Application for Volunteer Recruitment

VM VolunteerMatch

XHTML Extensible Hypertext Markup Language

XML Extensible Markup Language

# Introduction

We live in an age where the world is more interconnected than ever, and where social problems ought to ideally receive enough attention to attract people to contribute to their solution, either by engaging in volunteering activities or by other means. Volunteering has previously been defined as “freely chosen and deliberate helping activities that extend over time, and that are engaged in without any expectation of reward or other compensation, often through formal organizations, and that are performed on behalf of causes or individuals who desire assistance” [[Snyder and Omoto 2008](#Snyder_And_Omoto2008)]. These formal organizations, mainly non-profit, are generally composed of non-paid volunteers who join efforts towards a shared cause.

Even though there have been some initiatives that utilize technology to increase activity in the volunteering community, such as in the case of VolunteerMatch, these solutions might not be equipping organizations with all the adequate recruitment tools that may be needed to attract non-volunteers towards their events.

Previous research in the volunteering community, has shown that “providing non-volunteers with information about the support provided by the organization induced anticipated feelings of respect, which enhanced attraction to the volunteer organization” [[Boezeman and Ellemers 2008](#Boezeman_And_Ellemers2008)]. It is important to note that the same study also suggested that task and emotional support coming from the volunteer organization, was a much more relevant source of anticipated respect than the support received from co-volunteers. Ironically, organizations with the highest rates of attraction and retention, are the ones concerned about satisfying their volunteers [[Hager and Brudney 2004](#Hager_And_Brudney2004)]. This means that although volunteers are willing to contribute to a cause during their free time, they are more inclined towards organizations that they feel support and care for their needs.

Recent research on the use of chat services and its implications in the behavior of users online, has found that most customers feel more satisfied when using chat services as compared to other models of communication, such as email and phone calls [[Mero 2018](#Mero2018)]. The same findings suggested that users also increasingly preferred chat services for its real-time nature and instant responses. Although this study focused on user satisfaction when shopping online, it is suspected that the same satisfaction preferences might hold true for other online interaction environments.

VolunteerCloud therefore provides a cloud software solution to unite volunteers and organizations interested in fighting for a shared cause. VolunteerCloud also attempts to improve the organizations’ recruitment efforts by providing a real-time chat service that allows them to communicate with volunteers back and forth about opportunities, logistics, and more importantly, the support and tools that the organization can provide to the volunteer during the experience.

Following the demands of the modern-day world, VolunteerCloud is offered as a cloud software application following the “Software as a Service”(SaaS) paradigm. Software as a service allows users to use cloud-based apps over the Internet, usually with a web browser. All of the underlying infrastructure, including middleware, app software, and app data, are located in the service’s provider data center. In other words, the service provider manages both the hardware and the software remotely [[Microsoft 2018](#Microsoft2018)].

## Related Work

Previous research in the volunteering community has focused on exploring the reasons and relationships of poor volunteer attraction and retention rates, and on how to improve them. [Snyder and Omoto [2008]](#Snyder_And_Omoto2008) found that volunteering involvement is influenced by some factors such as the social distance between the volunteer and the entity asking for involvement, and the size of the contribution whether in the form of time or money. The study also found that people who showed higher rates of volunteering and monetary contributions had higher levels of education, and earned higher wages. [Boezeman and Ellemers [2008]](#Boezeman_And_Ellemers2008) found evidence that volunteer attraction and feelings of anticipated respect towards an organization are increased when non-volunteers are provided with information about the task and emotional support that the organization provides. Their research also showed that this information is much more effective when the support comes from the organization rather than the co-volunteers. When it comes to volunteer retention, [Hager and Brudney [2004]](#Hager_And_Brudney2004) suggest that organizations interested about keeping their volunteers should invest in making volunteers feel recognized and valued. In fact, they discovered that organizations that adopted practices directly concerned with satisfying volunteers had the highest rates of retention, while organizations that focused more on their management needs were unlikely to motivate volunteers.

[Mero J. [2018]](#Mero2018) recently studied the implications of offering chat services to users and its impact on consumer-retailer relationships. Mero’s survey of 6783 participants found that two-way communication has major effects on the levels of trust of satisfaction between online users and retailers. His findings suggested that the extent to which the retailer fosters reciprocal conversation, affects both attitudes and behavior intentions. Mero also pointed out how previous research indicated an increased preference in using chat services for obtaining customer support, over other popular alternatives such as emails and phone conversations.

Currently, there exists some software solutions that attempt to make societal impact through the use of technology, such are the cases of VolunteerMatch, Samaritan, Bedder, and Copia. [VolunteerMatch [2018]](#VolunteerMatch2018) attempts to strengthen the community by allowing volunteers to find opportunities near their locations. However, this solution lacks features to allow organizations to engage in real-time interactive communication, which could be an opportunity to increase their attraction and recruitment rates. [Samaritan [2018]](#Samaritan2018) is a mobile application that notifies users whenever a homeless person enrolled in the program is nearby, then it proceeds to reveal the homeless person’s story in an attempt to humanize them, and allows the user to decide whether to help the homeless person or not. [Bedder [2013]](#Bedder2013) was a software application initiative born out of a Hackathon event with the objective of assisting homeless people with the task of securing temporary housing, by providing real time information about the status of the different shelters. [Copia [2018]](#Copia2018) is a new software solution that makes great impact in the community by matching excess food with the entities that need it the most.

## Statement of the Problem

Few solutions have tried increasing activity in the volunteering community by utilizing technology to bring volunteers and organizations together. However, these solutions may not be providing organizations with enough tools to improve their volunteering recruitment efforts. It has been found in research studies, that volunteers are attracted to an organization the most when it is clear that the organization in question supports their volunteers and makes them feel valued and appreciated. In addition, these solutions do not empower organizations with the opportunity to show the ways in which they support volunteers, nor do they provide good opportunities for establishing contact between the two parties. Recent research on chat services, has shown that the satisfaction and trust levels that online users feel when shopping online, is increased with the involvement of live chats. The same is suspected of the levels of initial satisfaction and trust when non-volunteers are evaluating whether to join an organization or not. VolunteerCloud therefore provides a cloud software application solution that can serve as a platform to study how the rates of volunteer recruitment and attraction are affected when organizations step out of their shelves, and sell a wider picture of the support they offer to volunteers through the use of real-time online interaction.

## Statement of Objectives

This project meets the following objectives:

1. Understand the difficulties of attracting new volunteers to an organization.
2. Identify the features and requirements needed to enhance volunteer recruitment.
3. Produce the architecture and design specifications to meet the requirements of VolunteerCloud.
4. Develop the VolunteerCloud cloud software application.
5. Test the solution exhaustively until achieving a stable state.
6. Deploy VolunteerCloud in an Amazon Web Services server.

## Overview of Report

This report presents the following content: Chapter 1 introduces the problem to be solved, and the objective of this project; Chapter 2 presents the use cases, and the functional and non-functional requirements of this cloud software solution; Chapter 3 discusses VolunteerCloud’s architecture specification; Chapter 4 describes VolunteerCloud’s design specification; Chapter 5 presents VolunteerCloud’s features and it provides a guide into how it should be operated; Finally, Chapter 6 closes the report with a conclusion and plans for future work.

# VolunteerCloud Requirements Specification

Requirements gathering is one of the most essential tasks when developing a software system, since they are used to describe what the system must or mustn’t do. However, truly understanding the requirements of a problem is among the most difficult tasks that face a software engineer. When you first think about it, developing a clear understanding of requirements doesn’t seem that hard. After all, doesn’t the customer know what is required? Shouldn’t the end users have a good understanding of the features and functions that will provide benefit? Surprisingly, in many instances the answer to these questions is “no.” And even if customers and end users are explicit in their needs, those needs will change throughout the project. [[Pressman 2014](#Pressman2014)]

## Use Case Diagrams

Use cases help us understand and clarify the users’ required interactions with the system and can reveal most, if not all, functional requirements of the new system [[Dennis 2012](#Dennis2012)]. This is why despite the complexity involved in requirements gathering, use case-based requirements engineering has proven to be the best practice:

* A Use Case represents a small amount of work the software system is required to perform. Thus, decomposing a complex software system functionality into Use Cases enables the modularization needed to overcome the complexity. [[Balci 2016](#Balci2016)]
* Identifying the “real” functional requirements is always challenging for complex software systems. A Use Case describes an interaction, and based on that description, “real” functional requirements can be more successfully identified and associated with that Use Case. [[Balci 2016](#Balci2016)]
* Listing requirements one after the other, even in different categories, does not provide any help for transitioning from requirements to software system design. On the other hand, Use Cases turn themselves into classes in an object-oriented design and significantly facilitate the transition. [[Balci 2016](#Balci2016)]

VolunteerCloud's use cases are summarized in *Figure x* below. The types of users are comprised of volunteers and organizations, each of them with their own use cases. The main use cases in VolunteerCloud are enumerated the following way: 1) Manage Account, 2) Search Volunteers, 3) Search Organizations, 4) Search Opportunities, 5) Manage Volunteering Opportunities, 6) Manage Volunteering Interests, 7) View History Report, 8) Link account to VM, and 9) Webchat.

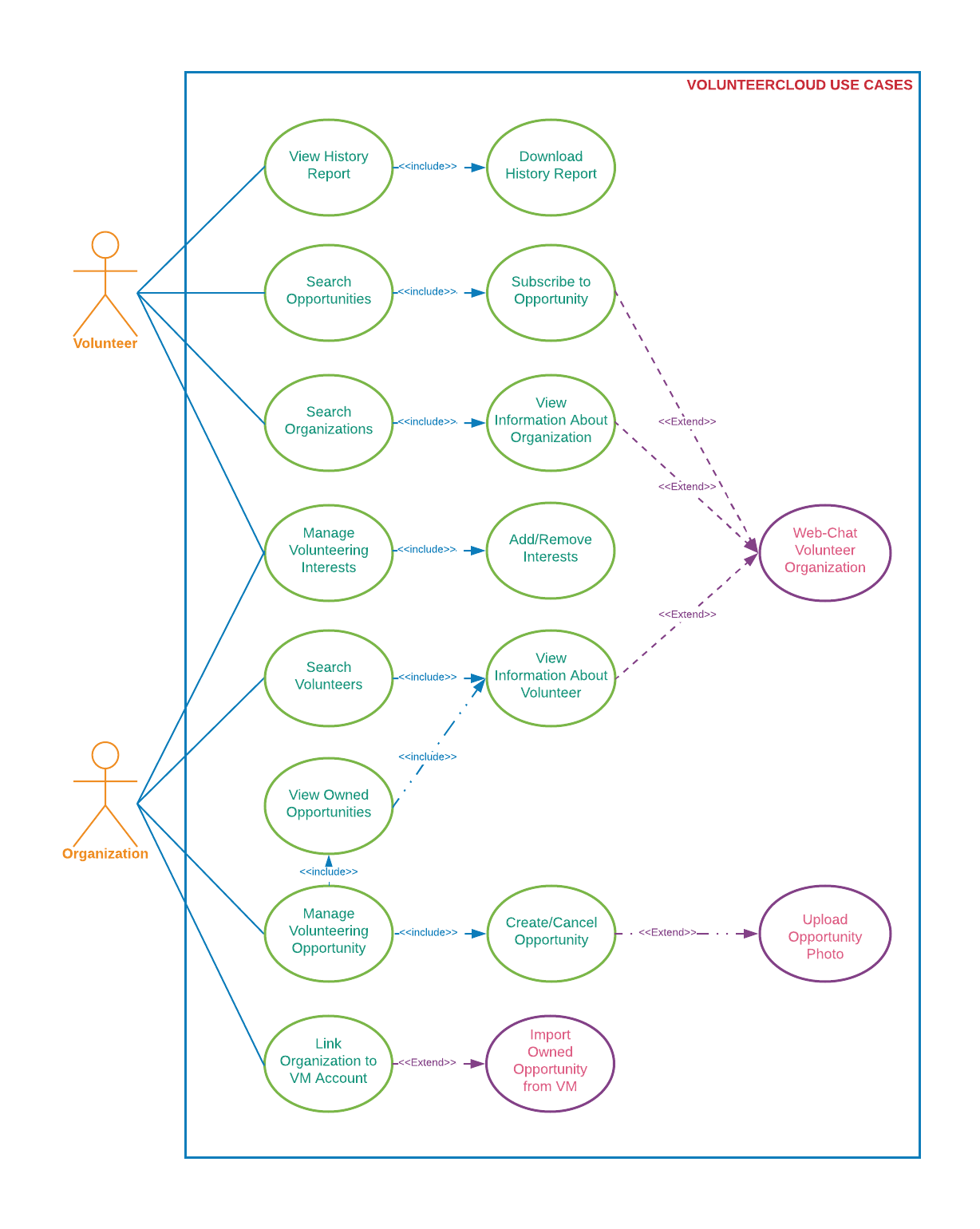


Figure x. VolunteerCloud Use Cases

The Manage Account Use Case is shown in *Figure 2* below. Both volunteers and organizations are able to create an account in VolunteerCloud. After enrollment, the users are able to view and update their account information, deactivate their account, reset their password, and upload a profile picture. After deactivating an account, users can reactive it by signing-in once again.

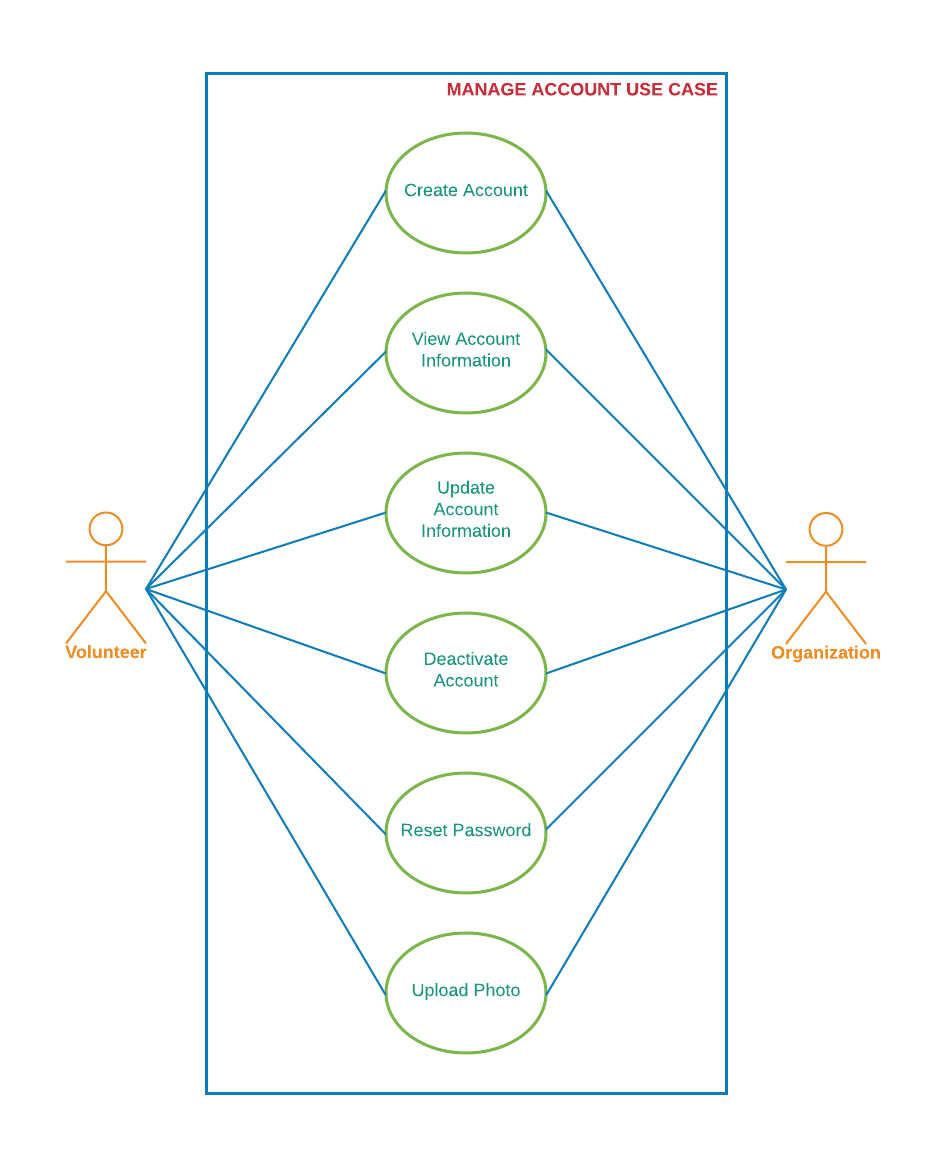


Figure x. Manage Account Use Case

The Search Volunteer Use Case is shown in *Figure 3*below. Organizations are able to search and recruit volunteers in VolunteerCloud. Organizations can search for volunteers by name, area of interest, and location. Then, a list of results matching the search criteria is displayed, and the organization can select a volunteer to access the profile. After selecting a record, the organization views detailed information about the volunteer, areas of interest, volunteering history, and the can start a chat with the volunteer to discuss about past or future volunteering opportunities.

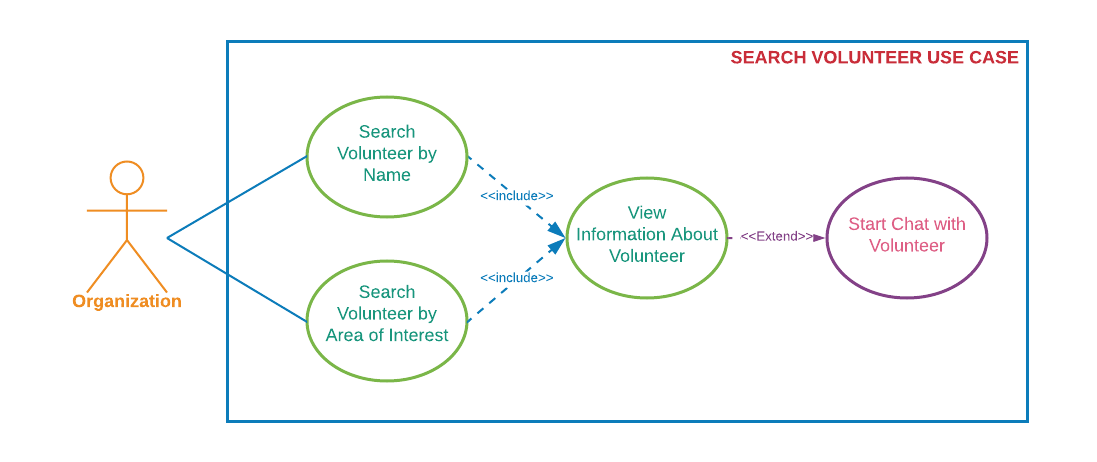
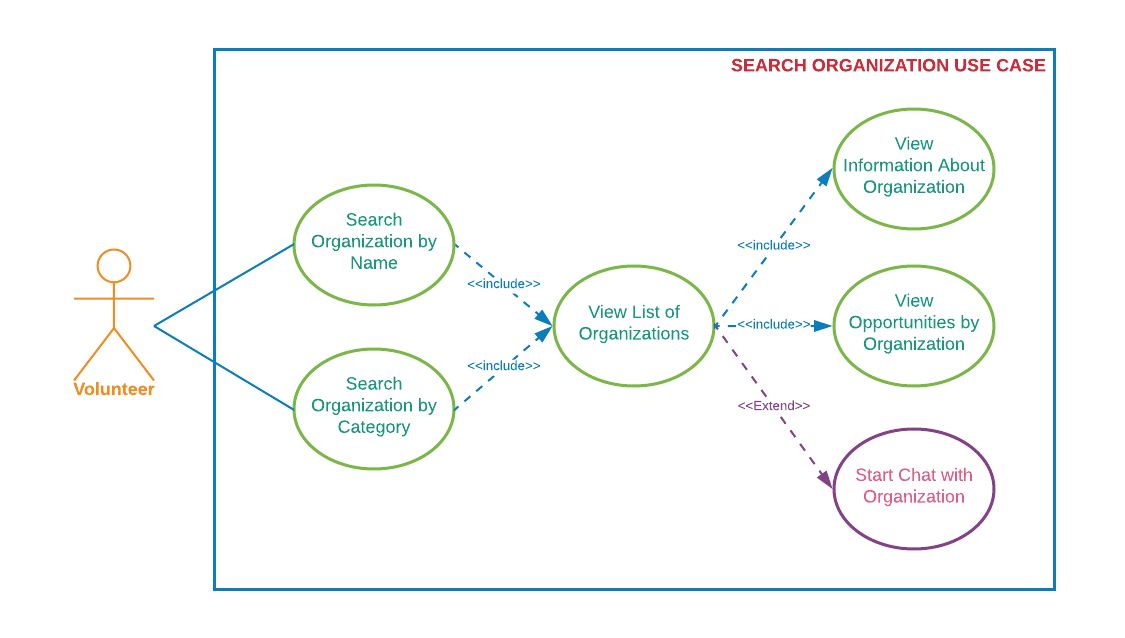


Figure x. Search Volunteer Use Case

The Search Organization Use Case is shown in *Figure x* below. Volunteers are able to search for organizations in VolunteerCloud. Volunteers can search for organizations by name, volunteering areas of interest, location, and keywords that match the organization’s mission statement. Then, a list of results matching the search criteria is displayed, and the volunteer can select an organization to access its profile. After selecting a record, the volunteer views detailed information about the organization, their volunteering areas of interest, the volunteering opportunities posted by them, and the volunteer can start a chat to discuss about past or future volunteering opportunities.

Figure x. Search Organization Use Case

The Search Opportunity Use Case is shown in *Figure x*below. Volunteers and Organizations are both able to search for volunteering opportunities in VolunteerCloud. Both types of users can search for opportunities by title, description keyword, organization name, location, date ranges, and volunteering area. Then, a list of results matching the search criteria is displayed, and the user can select an opportunity to access more information about it. Volunteers are able to subscribe and unsubscribe from the opportunity based on the opportunity’s date of occurrence, and organizations are able to manage the opportunity if owned by them. If interested, volunteers are able to start a chat with the organization who owns the opportunity.

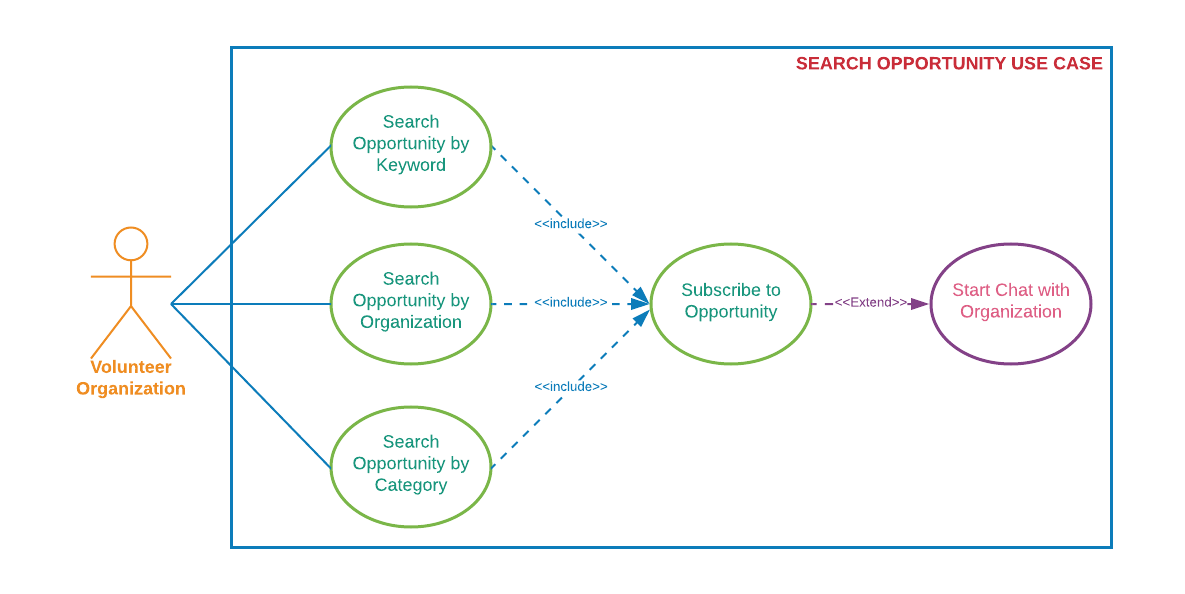


Figure x. Search Opportunity Use Case

The Manage Volunteering Opportunity Use Case is shown in *Figure x* below. Organizations are be able to view, create, update, and cancel volunteering opportunities at their disposal on VolunteerCloud. Organizations can create volunteering opportunities by specifying title, description, location, volunteering area, and date. In addition, organizations can upload a photo for each opportunity individually. Organizations are also able to access a list of all their volunteering opportunities by accessing their volunteering activity page. After selecting an opportunity, the organization is able to see detailed information about it, update it, or cancel it based on the date of occurrence. Furthermore, organizations are presented with a list of the registered participants for this opportunity, and they can confirm or decline the volunteer’s participation in the opportunity. Afterwards, organizations can follow up or thank the volunteer by starting a chat with them.

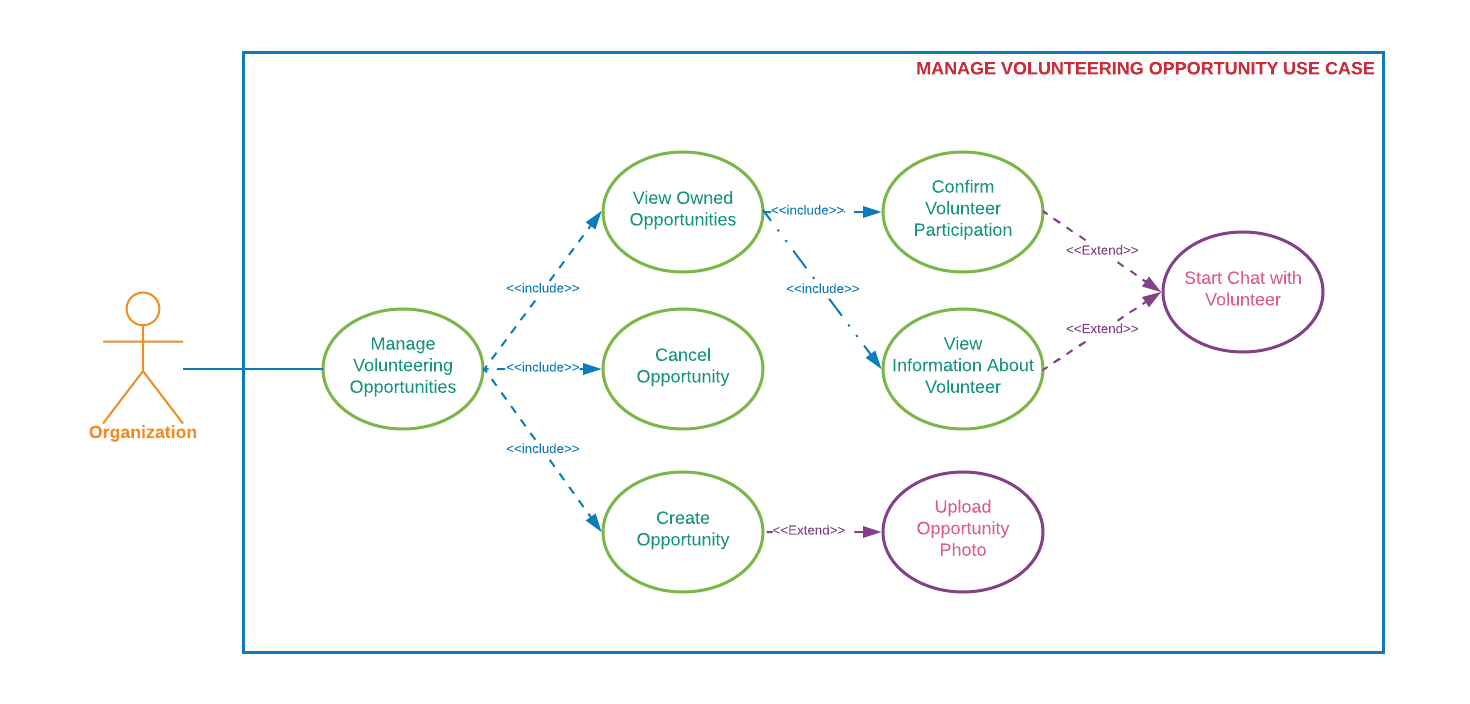
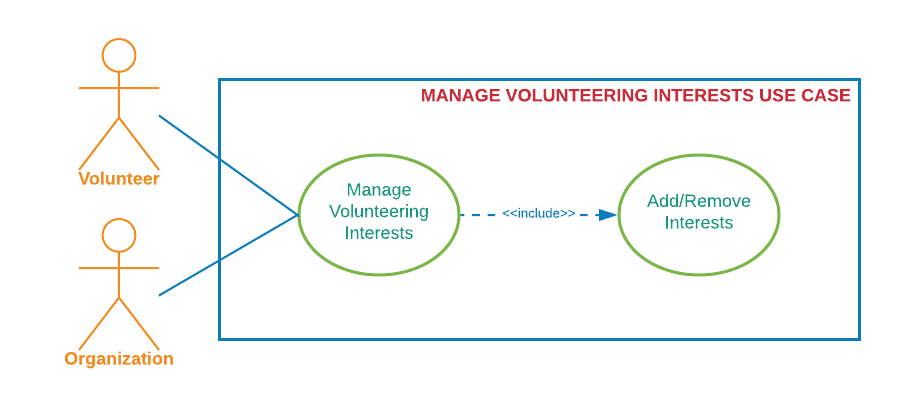
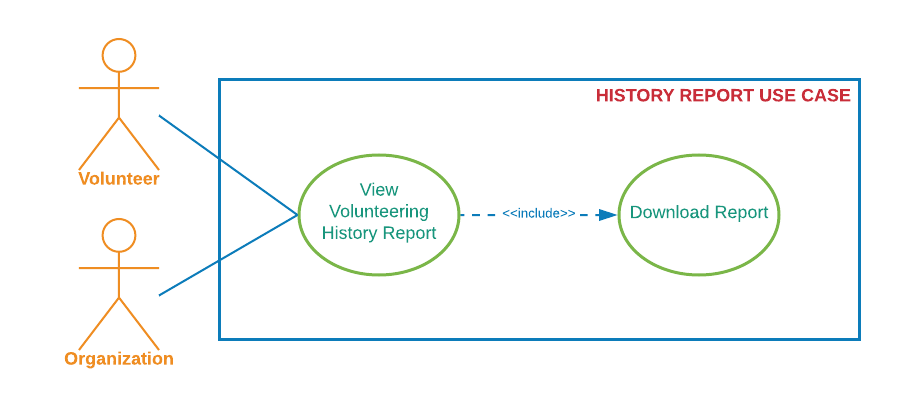


Figure x. Manage Volunteering Opportunity Use Case

The Manage Volunteering Interest Use Case is shown in *Figure 7* below. Volunteers and Organizations are able to update their volunteering areas of interests at any given moment in VolunteerCloud. Both types of users can access their volunteering activities page, and are able to edit their volunteering interests by selecting/unselecting the different areas hosted at VolunteerCloud. After updating their areas of interest, their searchability is updated as well.

Figure x. Manage Volunteering Interests Use Case

The History Report Use Case is shown in *Figure 8* below. Volunteers and Organizations are able to view, filter, and export their volunteering history at their disposal on VolunteerCloud. Users can access their volunteering activity page to see their volunteering history in a newest to oldest fashion. In addition, users can filter their volunteering history by activity title, keyword, location, volunteering area, and date ranges. Finally, users can download their whole or filtered volunteering history report which they’ll be able to store, print, or send to someone once downloaded.

Figure x. History Report Use Case

The Link To VM Use Case is shown in *Figure 9* below. Organizations are able to link their account to their Volunteer Match account (if any) on VolunteerCloud. Organizations can access their profile, select “Link To VM” as long as they haven’t linked their organization already, and search for their own account using the Volunteer Match API. After selecting their account, the accounts are linked, and organizations are able to import opportunities by selecting the “Import Opportunity” option. Once the organization accesses the “Import Opportunity” page, a list of volunteering activities from Volunteer Match is displayed, then organizations can select one and import it to VolunteerCloud for further volunteer recruitment.



Figure x. Link To VM Use Case

The Web-Chat Use Case is shown in *Figure x* below. As mentioned before, once an organization or volunteer access another user’s information page, they can start a chat with the other party. Both volunteers and organizations are able to start chats, send messages, receive messages, and access conversation history by using the Twilio API.

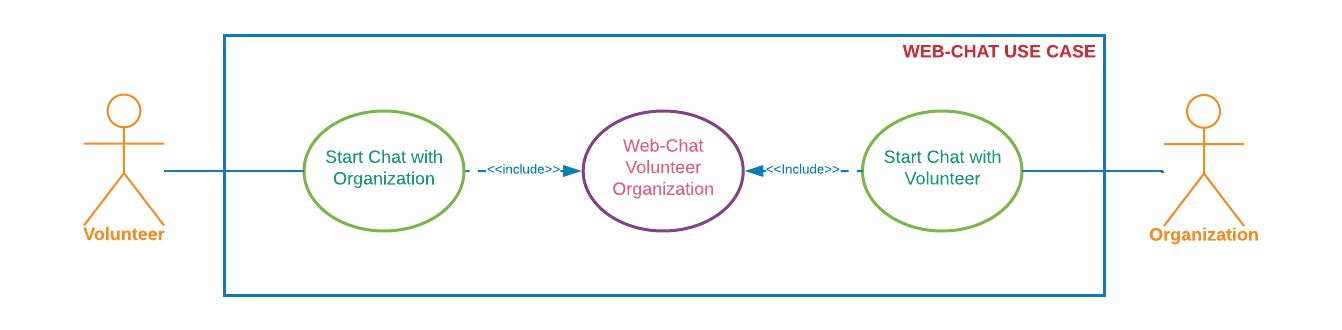


Figure x. Web-Chat Use Case

## Use Cases and Functional Requirements

Functional requirements: 1) Are statements of services that the software, software product or software-based system should provide, how the system should react to particular inputs, and how the system should behave in particular situations. 2) Describe functionality or system services. 3) Are requirements about the behavior and input-output transformations of the software, software product or software-based system. [[Balci 2016](#Balci2016)]

This subsection documents nine use cases offered in the VolunteerCloud cloud software application. Following the instructions for requirement specifications learned in [[Balci 2016](#Balci2016)], each use case is documented in terms of the following fields: description, actors, preconditions, flow of events, extension points, and post conditions. In addition, a set of functional requirements has been included for each use case.

### Use Case 1: Manage Account

Describes how volunteers and organizations can manage an account in VolunteerCloud.

#### Use Case 1.1 Documentation

Table 1. Use Case 1.1 Documentation

|  |  |
| --- | --- |
| Use Case 1.1 Documentation | |
| Use Case ID: | 1.1 |
| Use Case Name: | **Create Account**  This use case describes the process by which a Volunteer or an Organization can create an account in VolunteerCloud. |
| Actors: | |
| Volunteer or Organization | |
| Preconditions: | |
| User accesses VolunteerCloud. | |
| Flow of Events of the Primary Scenario: | |
| 1. Use case starts when user accesses the VolunteerCloud home page. 2. User selects “Register”. 3. User chooses whether to register as a Volunteer or an Organization. 4. User provides: username, password, email address, and security questions/answers. 5. User clicks ***Continue***. 6. The system verifies the input:   If the username, email, or password are not valid,  The system will give an alert message.  If the username or email already exists,  The system will give an alert message.  If the confirm password differs from the password,  The system will give an alert message.   1. User provides additional basic information: name, address, mission, website, phone number. 2. User clicks ***Submit***. 3. The system stores the information into the database. 4. User has access to the new account. | |
| Flow of Events of the Alternative Scenarios: | |
| None | |
| Flow of Events of the Exception Scenarios: | |
| * User cancels creation of the account. * User attempts registration with an existing email address or username. * Database exception occurs while account is being created. | |
| Extension Points: | |
| None | |
| “Used” Use Cases: | |
| None | |
| Postconditions: | |
| The user account is created and the ***Dashboard*** screen is displayed. | |

#### Functional Requirements Associated with Use Case 1.1

1. User shall be able to create at most 1 account per email address.
2. User shall be able to select a unique username that contains 4 to 14 alphanumeric characters.
3. Password shall contain 6 to 16 characters with at least; 1 uppercase, 1 lowercase, 1 number, and 1 special character.
4. User shall be able to select a user type: volunteer or organization.
5. User shall be able to select a security question and to store its answer.

### Use Case 1.2: Update an Account

Describes how volunteers and organizations can update their accounts in VolunteerCloud.

#### Use Case 1.2 Documentation

Table 2. Use Case 1.2 Documentation

|  |  |
| --- | --- |
| Use Case 1.2 Documentation | |
| Use Case ID: | 1.2 |
| Use Case Name: | **Update Account**  This use case describes the process by which a Volunteer or an Organization can update an account in VolunteerCloud. |
| Actors: | |
| Volunteer or Organization | |
| Preconditions: | |
| User must be signed-in  System displays ***Profile*** screen | |
| Flow of Events of the Primary Scenario: | |
| 1. Use case starts when user selects ***Edit*** on the ***Profile*** screen. 2. System enables editing. 3. User updates basic account information. 4. System validates changes to password 5. User clicks ***Submit***. 6. Changes are validated and stored. 7. VolunteerCloud displays ***Profile*** screen with new information. | |
| Flow of Events of the Alternative Scenarios: | |
| None | |
| Flow of Events of the Exception Scenarios: | |
| * User cancels update account editing. * Database exception occurs while account is being created. | |
| Extension Points: | |
| None | |
| “Used” Use Cases: | |
| None | |
| Postconditions: | |
| The user account is updated and an updated ***Profile*** screen is displayed. | |

#### Functional Requirements Associated with Use Case 1.2

1. User shall be able to update account information.
2. User shall be prevented from changing username or email address.
3. Password changes shall be verified; contain 6 to 16 characters, 1 uppercase, 1 lowercase, 1 number, and 1 special character.
4. User shall fill required fields.
5. Changes shall be visible after submission.
6. VolunteerCloud shall enable user to upload a photo to display in profile(jpg, jpeg, png).

### Use Case 1.3: Deactivate Account

Describes how volunteers and organizations can Deactivate their accounts in VolunteerCloud.

#### Use Case 1.3 Documentation

Table 3. Use Case 1.3 Documentation

|  |  |
| --- | --- |
| Use Case 1.3 Documentation | |
| Use Case ID: | 1.3 |
| Use Case Name: | **Deactivate Account**  This use case describes the process by which a Volunteer or an Organization can deactivate their VolunteerCloud account. |
| Actors: | |
| Volunteer or Organization | |
| Preconditions: | |
| User must be signed-in  System displays ***Profile*** screen | |
| Flow of Events of the Primary Scenario: | |
| 1. Use case starts when user selects ***Deactivate Account*** on the ***Profile*** screen. 2. System prompts user with action verification message. 3. User confirms action by selecting ***yes***. 4. Use case finishes when account is deactivated and user is redirected to ***Home Screen***. | |
| Flow of Events of the Alternative Scenarios: | |
| None | |
| Flow of Events of the Exception Scenarios: | |
| * User cancels deactivation of the account. * Database exception occurs while account is being deactivated. | |
| Extension Points: | |
| None | |
| “Used” Use Cases: | |
| None | |
| Postconditions: | |
| The user account is deactivated and the ***Home Screen*** is displayed. | |

#### Functional Requirements Associated with Use Case 1.3

1. Users shall be able to deactivate their accounts.
2. VolunteerCloud shall prompt user with action verification message prior to deactivation.
3. User’s information shall remain stored after deactivation of account.
4. User’s profile information shall not be accessible if account is deactivated.
5. Chat shall only be available with active users.

### Use Case 2: Search Volunteers

Describes how users can search volunteers in VolunteerCloud.

#### Use Case 2 Documentation

Table 10. Use Case 2 Documentation

|  |  |
| --- | --- |
| Use Case 2 Documentation | |
| Use Case ID: | 2 |
| Use Case Name: | **Search Volunteers**  Describes how users can search volunteers in VolunteerCloud. |
| Actors: | |
| Organization | |
| Preconditions: | |
| * User has active account. * User has logged-in to the system. | |
| Flow of Events of the Primary Scenario: | |
| 1. The use case starts after the user clicks ***Search Volunteers***. 2. User inputs a zip code, name, and/or area of interest. 3. User clicks ***Search***. 4. VolunteerCloud queries database and returns results:   If no matches were found,  The system will display a notification message.  If matches were found,  The system will return and display a list of volunteers   1. If the userwants to access more information about the volunteer,   The userclicks on the volunteer’s record   1. If the userwants to chat with the volunteer,   The userclicks ***Chat***, see “use case": Web Chat   1. The use case ends. | |
| Flow of Events of the Alternative Scenarios: | |
| None | |
| Flow of Events of the Exception Scenarios: | |
| * Database exception occurs and no results are returned. * Network connectivity is dropped. | |
| Extension Points: | |
| None | |
| “Used” Use Cases: | |
| 1. Web Chat | |
| Postconditions: | |
| Organization receives a list of volunteers that match the search criteria. | |

#### Functional Requirements Associated with Use Case 2

1. VolunteerCloud shall allow organizations to search for volunteers.
2. VolunteerCloud shall allow organizations to search volunteers by location, name, or categories.
3. VolunteerCloud shall display message to notify user if no results were found.
4. VolunteerCloud shall present results as a list of volunteers by displaying basic information such as name, contact information, mission, and photo of the volunteer.
5. VolunteerCloud shall enable the user to access more information about the volunteer by selection.
6. VolunteerCloud shall not return volunteers whose accounts have been deactivated.

### Use Case 3: Search Organizations

Describes how users can search for organizations in VolunteerCloud.

#### Use Case 3 Documentation

Table 5. Use Case 3 Documentation

|  |  |
| --- | --- |
| Use Case 3 Documentation | |
| Use Case ID: | 3 |
| Use Case Name: | **Search Organization**  Describes how users can search for organizations in VolunteerCloud. |
| Actors: | |
| Volunteer | |
| Preconditions: | |
| * Volunteer has active account. * Volunteer has logged-in to the system. | |
| Flow of Events of the Primary Scenario: | |
| 1. Use case starts when Volunteer is redirected to “Search Organization” screen 2. Volunteer inputs a name, zip code, keyword, and/or area of interest. 3. Volunteer clicks ***Search***. 4. VolunteerCloud queries database and returns results:   If no matches were found,  The system will display a notification message.  If matches were found,  The system will return and display a list of organizations   1. If the Volunteerwants to access more information about the organization,   The Volunteerclicks on the organization’s record   1. If the Volunteerwants to see the organization’s opportunities,   The Volunteerclicks on the organization’s record   1. If the Volunteerwants to chat with the organization,   The Volunteerclicks ***Chat***, view “use case": Web Chat   1. The use case ends. | |
| Flow of Events of the Alternative Scenarios: | |
| None | |
| Flow of Events of the Exception Scenarios: | |
| * Database exception occurs and no results are returned. * Chat disabled due to organization having deactivated their account. * Subscription to opportunity has been disabled due to organization having deactivated their account. * Network connectivity is dropped. | |
| Extension Points: | |
| None | |
| “Used” Use Cases: | |
| 1. Web Chat | |
| Postconditions: | |
| Volunteer receives a list of organizations that matches search criteria. | |

#### Functional Requirements Associated with Use 3

1. VolunteerCloud shall allow volunteers to search for organizations.
2. VolunteerCloud shall allow volunteers to search organizations by name, location, keyword, or areas of interest.
3. VolunteerCloud shall display message to notify user if no organizations that match the criteria were found.
4. VolunteerCloud shall present results as a list of organizations by displaying basic information such as name, contact information, mission, and a photo.
5. VolunteerCloud shall allow the user to access detailed information about the organization.
6. VolunteerCloud shall allow the user to access the opportunities posted by the organization.
7. VolunteerCloud shall only return organizations with active accounts in the search result.

### Use Case 4: Search Opportunities

Describes how users can search for volunteering opportunities in VolunteerCloud.

#### Use Case 4 Documentation

Table 4.Use Case 4 Documentation

|  |  |
| --- | --- |
| Use Case 4 Documentation | |
| Use Case ID: | 4 |
| Use Case Name: | **Search Opportunity**  Describes how users can search for volunteering opportunities in VolunteerCloud. |
| Actors: | |
| Volunteer or Organization | |
| Preconditions: | |
| * User has active account. * User has logged-in to the system. | |
| Flow of Events of the Primary Scenario: | |
| 1. Use case starts when User is redirected to *Search Opportunities* screen 2. User inputs a location, name of the organization, title, keyword, date, and/or area of interest. 3. User clicks ***Search***. 4. VolunteerCloud queries database and returns results:   If no matches were found,  The system will display a notification message.  If matches were found,  The system will return and display a list of opportunities   1. If the Userwants to access more information about the opportunity,   The Userclicks on the opportunity’s record   1. If the Userwants to access more information about the organization,   The Userclicks on the organization’s name   1. If the Volunteerwants to subscribe/unsubscribe to the opportunity,   The Volunteerclicks ***Subscribe/Unsubscribe***   1. If the Volunteerwants to chat about the opportunity,   The Volunteerclicks ***Chat***, use “use case": Web Chat   1. The use case ends. | |
| Flow of Events of the Alternative Scenarios: | |
| None | |
| Flow of Events of the Exception Scenarios: | |
| * Database exception occurs and no results are returned. * Chat disabled due to organization having deactivated their account. * Subscription to opportunity has been disabled due to organization having deactivated their account. * Network connectivity is dropped. | |
| Extension Points: | |
| None | |
| “Used” Use Cases: | |
| 1. Web Chat | |
| Postconditions: | |
| Volunteer receives a list of volunteering opportunities that matches search criteria. | |

#### Functional Requirements Associated with Use Case 4

1. VolunteerCloud shall allow users to search for opportunities.
2. VolunteerCloud shall allow users to search opportunities by date ranges, location, title, keyword, name of the organization, or areas of interest.
3. VolunteerCloud shall allow volunteers to subscribe to however many opportunities he/she wants.
4. VolunteerCloud shall allow users to unsubscribe from opportunity before it takes place.
5. VolunteerCloud shall notify user if no search results were found.
6. VolunteerCloud shall present results as a list of opportunities by displaying basic information such as title, description, date, name of the organization, and a photo.
7. VolunteerCloud shall enable the user to access more information about the organization.
8. VolunteerCloud shall enable the user to access more information about the opportunity.
9. VolunteerCloud shall not allow volunteers to subscribe to opportunity if the organization’s account is deactivated.
10. VolunteerCloud shall not allow volunteers to subscribe to opportunity if the date of occurrence has already passed.

### Use Case 5: Manage Volunteering Opportunities

Describes how users can manage the volunteering opportunities posted by them in VolunteerCloud.

#### Use Case 5.1 Documentation

Table 7. Use Case 5.1 Documentation

|  |  |
| --- | --- |
| Use Case 5.1 Documentation | |
| Use Case ID: | 5.1 |
| Use Case Name: | **Manage Volunteering Opportunity**  Describes how users can access and manage the volunteering opportunities posted by them in VolunteerCloud. |
| Actors: | |
| Organization | |
| Preconditions: | |
| * Organization has active account. * Organization has logged-in to the system. | |
| Flow of Events of the Primary Scenario: | |
| 1. The use case starts when the Organization enters the ***View Activity*** screen. 2. VolunteerCloud shows a list of the organization’s opportunities. 3. The user clicks on the opportunity’s record to access more information about the opportunity. 4. VolunteerCloud displays basic information about the opportunity, and lists the volunteers subscribed to it. 5. If the Organization wants to create a new opportunity,   View “use case”: Create Volunteering Opportunity.   1. If the Organization wants to edit an opportunity,   View clicks ***Edit***.   1. If the Organization wants to cancel an opportunity,   View “use case”: Cancel Volunteering Opportunity.   1. If the Organization wants to confirm or deny a volunteer’s participation in the event,   The Organization selects the appropriate option in the ***Participation*** dropdown.   1. If the Organization wants to see more details about one of the registered volunteers,   Organization clicks on the volunteer’s name.  Then VolunteerCloud displays detailed information about the volunteer.   1. If the Organizationwants to chat with the volunteer,   The Organizationclicks ***Chat***, use “use case": Web Chat   1. The use case ends. | |
| Flow of Events of the Alternative Scenarios: | |
| 1. If the user selects ***New Opportunity***, view use case “Create Volunteering Opportunity”. 2. If the user selects ***Cancel Opportunity***, view use case “Cancel Volunteering Opportunity”. | |
| Flow of Events of the Exception Scenarios: | |
| * Database exception occurs and data is not loaded properly. * Network connectivity is dropped. * Chat with volunteer disabled due to volunteer having deactivated his/her account. * Volunteer’s participation fails to be stored correctly | |
| Extension Points: | |
| None | |
| “Used” Use Cases: | |
| 1. Create Volunteering Opportunity 2. Cancel Volunteering Opportunity 3. Web Chat | |
| Postconditions: | |
| User has accessed opportunity, seen information about the registered volunteers, and confirm their participation in the event. | |

#### Functional Requirements Associated with Use Case 5.1

1. VolunteerCloud shall allow the user to access information about their owned volunteering opportunities such as: title, category, description, date, location, and registered volunteers.
2. VolunteerCloud shall enable the user to edit information about their owned volunteering opportunities such as: address, status, description, category, and title.
3. VolunteerCloud shall allow users to edit their volunteering opportunity prior to the date of its occurrence.
4. VolunteerCloud shall allow users to cancel their volunteering opportunity prior to the date of its occurrence.
5. VolunteerCloud shall not allow users to edit an opportunity after the date of its occurrence has passed.
6. VolunteerCloud shall not allow users to cancel an opportunity after the date of its occurrence has passed.
7. VolunteerCloud shall allow users access more information about its registered volunteers by clicking on their names.
8. VolunteerCloud shall enable users to confirm or deny a volunteer’s participation in an event after the date of occurrence has passed.

#### Use Case 5.2: Create Volunteering Opportunity

Describes how users can create a new volunteering opportunity in VolunteerCloud

#### Use Case 5.2 Documentation

Table 8. Use Case 5.2 Documentation

|  |  |
| --- | --- |
| Use Case 5.2 Documentation | |
| Use Case ID: | 5.2 |
| Use Case Name: | **Create Volunteering Opportunity**  Describes how users can create a new volunteering opportunity in VolunteerCloud. |
| Actors: | |
| Organization | |
| Preconditions: | |
| * Organization has active account. * Organization has logged-in to the system. | |
| Flow of Events of the Primary Scenario: | |
| 1. The use case starts after the user clicks ***Create Opportunity***. 2. User provides: title, category, description, date, and address. 3. User clicks ***Submit***. 4. The system verifies the input:   If the title, category, or address are empty,  The system will give an alert message.  If the chosen date is a date prior to the current date,  The system will give an alert message.   1. The system stores the information into the database. 2. The new opportunity is accessible for subscription | |
| Flow of Events of the Alternative Scenarios: | |
| None | |
| Flow of Events of the Exception Scenarios: | |
| * User cancels creation of the opportunity. * Database exception occurs while opportunity is being created. * User attempts to create opportunity using date from the past. | |
| Extension Points: | |
| None | |
| “Used” Use Cases: | |
| None | |
| Postconditions: | |
| The opportunity is created and an updated dashboard screen is displayed. | |

#### Functional Requirements Associated with Use Case 5.2

1. User shall be able to create as many volunteering opportunities as they desire.
2. User shall be able to select the volunteering area to which the opportunity belongs to.
3. User shall be able to indicate the address where the opportunity will occur.
4. User shall be able to include a title and a description for the volunteering opportunity.
5. User shall be able to indicate the future date when the opportunity will occur.
6. System shall verify input and provide an alert message if required fields are left blank.
7. System shall verify that the opportunity’s date is in the future.

#### Use Case 5.3: Cancel Volunteering Opportunity

Describes how users can cancel a volunteering opportunity in VolunteerCloud.

#### Use Case 5.3 Documentation

Table 9. Use Case 5.3 Documentation

|  |  |
| --- | --- |
| Use Case 5.3 Documentation | |
| Use Case ID: | 5.3 |
| Use Case Name: | **Cancel Volunteering Opportunity**  Describes how users can cancel a volunteering opportunity in VolunteerCloud |
| Actors: | |
| Organization | |
| Preconditions: | |
| * Organization has active account. * Organization has logged-in to the system. * There exists an opportunity to be canceled. * Opportunity to be canceled has not occurred yet. | |
| Flow of Events of the Primary Scenario: | |
| 1. The use case starts after the user clicks ***Cancel Opportunity***. 2. System prompts user with action verification message. 3. User confirms action by selection ***yes***. 4. Opportunity is canceled and “Dashboard” screen is updated accordingly. | |
| Flow of Events of the Alternative Scenarios: | |
| None | |
| Flow of Events of the Exception Scenarios: | |
| * User declines action verification prompt. * Database exception occurs while opportunity is being canceled. * User attempts to cancel opportunity that has already occurred. | |
| Extension Points: | |
| None | |
| “Used” Use Cases: | |
| None | |
| Postconditions: | |
| The opportunity is canceled and data its status is changed to inactive. | |

#### Functional Requirements Associated with Use Case 5.3

1. VolunteerCloud shall enable users to cancel opportunities prior to date of occurrence.
2. VolunteerCloud shall prompt user for cancelation confirmation.
3. VolunteerCloud shall disable opportunity subscription after being canceled.

### Use Case 6: Manage Volunteering Areas of Interests

Describes how users can manage their volunteering areas of interest in VolunteerCloud

#### Use Case 6 Documentation

Table 11. Use Case 6 Documentation

|  |  |
| --- | --- |
| Use Case 6 Documentation | |
| Use Case ID: | 6 |
| Use Case Name: | **Manage Volunteering Areas of Interest**  Describes how users can manage their volunteering areas of interest in VolunteerCloud. |
| Actors: | |
| Volunteer or Organization | |
| Preconditions: | |
| * User has active account. * User has logged-in to the system. * System is currently displaying ***View Activity***screen. | |
| Flow of Events of the Primary Scenario: | |
| 1. The use case starts after the user clicks ***Edit*** under ***Areas of Interest***. 2. System enables checkboxes listing volunteering areas of interest 3. If the useris interested in an area of interest,   User checks the volunteering category   1. If the user is not interested in an area of interest,   User unchecks the volunteering category   1. The userclicks ***Save*** 2. Screen is updated 3. The use case ends. | |
| Flow of Events of the Alternative Scenarios: | |
| None | |
| Flow of Events of the Exception Scenarios: | |
| * Database exception occurs and changes are not stored. * Network connectivity is dropped. | |
| Extension Points: | |
| None | |
| “Used” Use Cases: | |
| None | |
| Postconditions: | |
| The changes are saved and the user’s ***View Activity*** screen is updated. | |

#### Functional Requirements Associated with Use Case 6

1. VolunteerCloud shall display the users’ volunteering areas of interest.
2. VolunteerCloud shall enable users to update their areas of interest.
3. VolunteerCloud shall update the user’s searchability after updating areas of interest.

### Use Case 7: View History Report

Describes how users can access and download their volunteering activities history.

#### Use Case 7 Documentation

Table 6. Use Case 7 Documentation

|  |  |
| --- | --- |
| Use Case 7 Documentation | |
| Use Case ID: | 7 |
| Use Case Name: | **View History Report**  Describes how users can access and download their volunteering activities history. |
| Actors: | |
| Volunteer or Organization | |
| Preconditions: | |
| * User has active account. * User has logged-in to the system. | |
| Flow of Events of the Primary Scenario: | |
| 1. The use case starts when the user accesses the ***View Activity***screen. 2. VolunteerCloud displays list of volunteering opportunities where user has participated. 3. User filters history by location, name of the organization, title, keyword, and/or area of interest. 4. If the user wants to see his/her volunteering history within a range of time,   The user inputs the desired dates.   1. If the user wants to download the report,   The user clicks ***Download***.   1. The use case ends. | |
| Flow of Events of the Alternative Scenarios: | |
| None | |
| Flow of Events of the Exception Scenarios: | |
| * Database exception occurs and data is not loaded properly. * Downloaded report is corrupted. * Network connectivity is dropped. | |
| Extension Points: | |
| None | |
| “Used” Use Cases: | |
| None | |
| Postconditions: | |
| The user accesses his/her volunteering history report and has downloaded report. | |

#### Functional Requirements Associated with Use Case 7

1. VolunteerCloud shall allow users to access all of their volunteering history information.
2. VolunteerCloud shall allow users to export their volunteering history report as a pdf document.
3. VolunteerCloud shall allow users to filter their volunteering history by location, name of the organization, title, keyword, and/or area of interest.
4. VolunteerCloud shall allow users to access their volunteering history information within their desired date range.
5. VolunteerCloud shall display an alert message if the start date takes place after the end date in the desired date range.

### Use Case 8: Linking Organization to VolunteerMatch’s Account

Describes how users can link their organization to their VolunteerMatch’s account in VolunteerCloud.

#### Use Case 8 Documentation

Table 13. Use Case 8 Documentation

|  |  |
| --- | --- |
| Use Case 8 Documentation | |
| Use Case ID: | 8 |
| Use Case Name: | **Linking Organization to VolunteerMatch’s Account**  Describes how users can link their organization to their VolunteerMatch’s account in VolunteerCloud. |
| Actors: | |
| Organization | |
| Preconditions: | |
| * User has active account. * User has logged-in to the system. * User owns account in VolunteerMatch. * System is currently displaying ***Profile***screen. * User’s organization has not been previously linked to another VolunteerMatch’s account. | |
| Flow of Events of the Primary Scenario: | |
| 1. The use case starts after the user clicks ***Link to VolunteerMatch***. 2. The system allows the user to perform a VolunteerMatch’s account search. 3. The user inputs a zip code and name. 4. User clicks ***Search***. 5. The system makes a call to the VolunteerMatch’s API and returns results:   If no matches were found,  The system will display a notification message.  If matches were found,  The system will return and display a list of organizations   1. The user selects his/her VolunteerMatch’s account. 2. The system stores the selected organization’s VolunteerMatch ID in the database. 3. System displays a success message. 4. The user clicks ***Import Opportunities*** 5. The system displays opportunities owned by the organization in VolunteerMatch. 6. The user selects the opportunity to be imported. 7. The system imports and stores the opportunity from VolunteerMatch into the database. 8. The use case ends. | |
| Flow of Events of the Alternative Scenarios: | |
| None | |
| Flow of Events of the Exception Scenarios: | |
| * Database exception occurs and imported opportunities are not stored. * Organization from VolunteerMatch has already been linked to VolunteerCloud * Opportunity from VolunteerMatch has already been imported into VolunteerCloud. * Network connectivity is dropped. | |
| Extension Points: | |
| None | |
| “Used” Use Cases: | |
| None | |
| Postconditions: | |
| Organization’s account has been linked to his/her organization in VolunteerMatch.  Organization is able to import owned opportunities in VolunteerMatch. | |

#### Functional Requirements Associated with Use Case 8

1. VolunteerCloud shall allow users to link their organizations to their accounts in VolunteerMatch.
2. VolunteerCloud shall allow users to import opportunities from VolunteerMatch after accounts have been linked.
3. VolunteerCloud shall not allow users to link their accounts to more than 1 organization.
4. VolunteerCloud shall allow changes before importing volunteering opportunity.

### Use Case 9: Web Chat

Describes how users can chat with other users in VolunteerCloud through the web-chat interface.

#### Use Case 9 Documentation

Table 12. Use Case 9 Documentation

|  |  |
| --- | --- |
| Use Case 9 Documentation | |
| Use Case ID: | 9 |
| Use Case Name: | **Web Chat**  Describes how users can chat with other users in VolunteerCloud through the web-chat interface. |
| Actors: | |
| Volunteer or Organization | |
| Preconditions: | |
| * User has active account. * User has logged-in to the system. * User has performed any of the use cases ***“2, 3, 4, or 5”*** beforehand. | |
| Flow of Events of the Primary Scenario: | |
| 1. The use case starts after the user clicks ***Chat*** with another user. 2. System displays web chat interface screen. 3. System lists previous messages held between the two users. 4. User types in message into input box. 5. The userclicks ***Send***. 6. User sends more messages or waits for response. 7. The use case ends. | |
| Flow of Events of the Alternative Scenarios: | |
| None | |
| Flow of Events of the Exception Scenarios: | |
| * Database exception occurs and no results are returned. * Receiver user’s account has been deactivated. * Message fails to be sent. * Network connectivity is dropped. | |
| Extension Points: | |
| None | |
| “Used” Use Cases: | |
| None | |
| Postconditions: | |
| Users can successfully hold an ongoing chat. | |

#### Functional Requirements Associated with Use Case 9

1. VolunteerCloud shall allow Volunteers and Organizations to engage in communication through a web chat interface.
2. VolunteerCloud shall display previous messages held between both parties of the conversation. display their volunteering areas of interest.
3. VolunteerCloud shall allow web chat communications between active accounts only.
4. VolunteerCloud shall allow users to send messages when the receiving user is offline.
5. VolunteerCloud shall enable users to access messages received while being offline.
6. VolunteerCloud shall enable users to access list of active conversations.
7. VolunteerCloud shall notify users when unread messages exist.
8. VolunteerCloud’s web chat interface shall display both of the users’ names, and their profile picture.

## Non-Functional Requirements

Non-functional requirements are requirements that are unrelated to functionality of the software, software product or software-based system. For example: Requirements for Interoperability, Performance, Usability, Standards, Delivery, Portability, Privacy, Safety. [[Balci 2016](#Balci2016)]

This section displays the list of non-functional requirements that VolunteerCloud shall meet:

1. VolunteerCloud shall be accessible over HTTP request-response based connections.
2. VolunteerCloud shall be freely available for no cost.
3. VolunteerCloud shall be accessible by computers and mobile devices.
4. VolunteerCloud shall be accessible through the following web browsers:
   1. Google Chrome
   2. Mozilla Firefox
5. VolunteerCloud shall be developed in an environment using the following software:
   1. NetBeans IDE,
   2. Primefaces Framework,
   3. Glassfish Java EE Application Server,
   4. MySQL Database.
6. VolunteerCloud shall be deployed and delivered on a virtual private server hosted by Amazon Web Services.
7. The virtual private server shall contain the following software specifications:
   1. CentOS operating system,
   2. Glassfish Java EE Application Server,
   3. MySQL Relational Database and Management System
   4. Server URL: http://www.degallosolutions.com/
8. VolunteerCloud shall be capable of communicating with other APIs through HTTP responses.
9. VolunteerCloud shall be stable and maintainable.
10. VolunteerCloud shall be flexible and extendable to adapt richer features in the future.

## 

# VolunteerCloud Architecture Specification

Software application architecture is the process of defining a structured solution that meets all of the technical and operational requirements, while optimizing common quality attributes such as performance, security, and manageability. It involves a series of decisions based on a wide range of factors, and each of these decisions can have considerable impact on the quality, performance, maintainability, and overall success of the application. [[Microsoft 2009](#Microsoft2009)]

The architecture is not the operational software. Rather, it is a representation that enables you to (1) analyze the effectiveness of the design in meeting its stated requirements, (2) consider architectural alternatives at a stage when making design changes is still relatively easy, and (3) reduce the risks associated with the construction of the software. [[Pressman 2014](#Pressman2014)]

Choosing the architecture of a system is highly important due to the impact it can have or not in the end product. It is important to remember that the architecture chosen for a software product will likely affect certain of its quality attributes such as modifiability, performance, security, and even usability. Choosing the right architecture is also important due to the trade-offs and constraints that a style imposes on a system. Binding it all together, George Fairbanks [[Fairbanks 2010](#Fairbanks2010)] explained how architecture affects the system in a more concrete manner by providing the following points:

* Architecture influences quality attributes.
* Architecture acts as the skeleton of a system.
* Architecture is (mostly) orthogonal to functionality.
* Architecture constrains systems.

This section presents the architecture models used in this project, and explains why these models are a good fit for the implementation of VolunteerCloud.

## Client-Server Architecture

In the client-server architectural style, clients request services from servers. The request is usually synchronous and across a request-reply connector, but can vary. There is an asymmetry between client and server in that the client can request that the server do work, but not the reverse [[Fairbanks 2010](#Fairbanks2010)]. The client-server architecture is famous in the industry of providing software as a service because it does not require the server to know the existence of all its clients, while the clients are able to perform as many requests as possible.

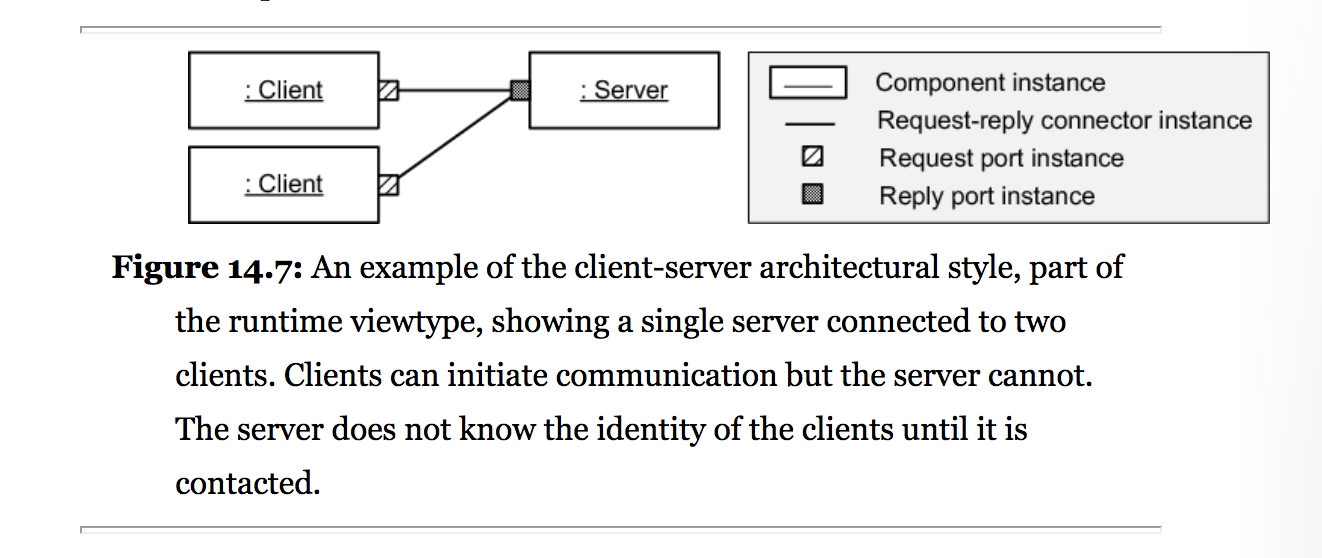


Figure X. Client-Server Architecture [[Fairbanks 2010](#Fairbanks2010)]

One common variant of this architecture style is known as the N-tier client-server architecture style. This style is regularly used because of the horizontal scalability it offers to the system as new or richer version of the requirements appear. For example, figure [N-Tier] can be easily mapped to a software solution that processes the clients’ requests through a load balancer before sending them to their respective application servers.

Client Load Balancer Server

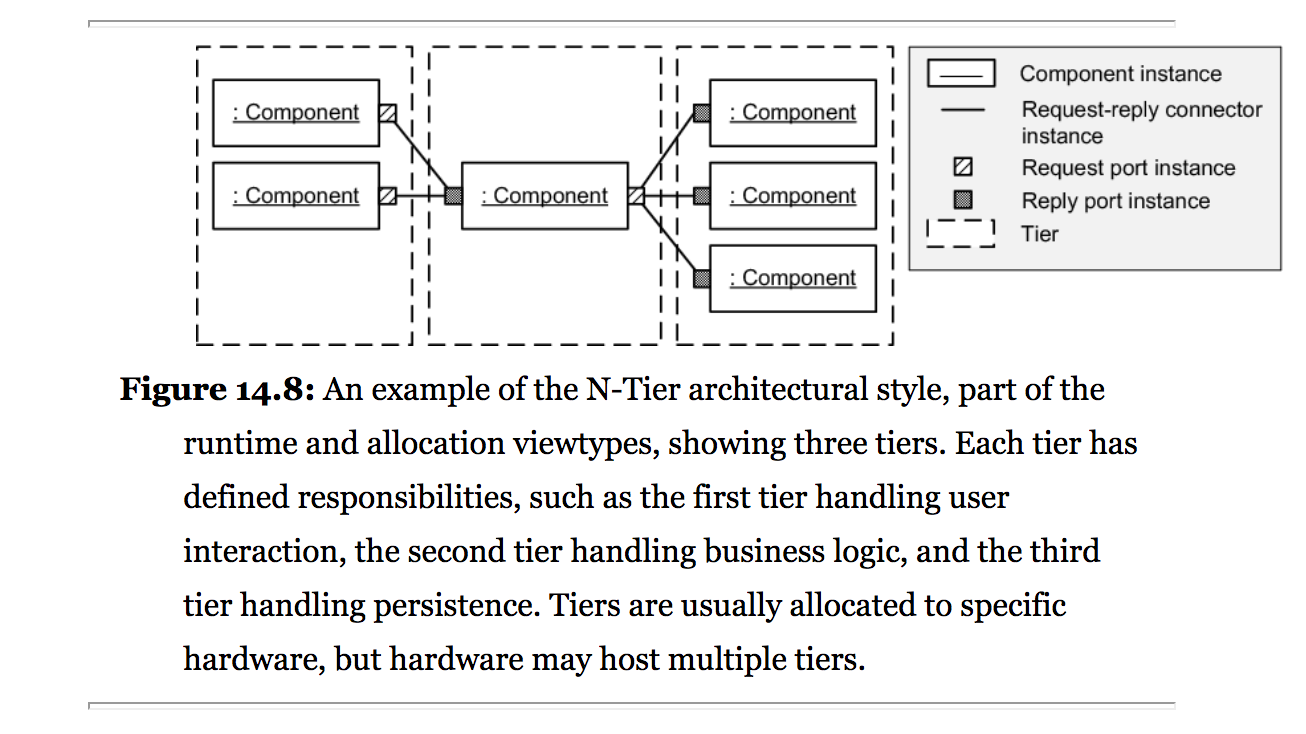


Figure X. N-Tier Client-Server Architecture [[Fairbanks 2010](#Fairbanks2010)]

Because this architectural style separates client code and server code into different components, it is possible to improve system performance by shuffling the components among computer processes [[Pfleeger 2010](#Pfleeger2010)]. This is one of the main reasons why technologies able to execute more logic on the client side such as JavaScript, have gained momentum in the past years.

## Model-View-Controller Architecture, MVC

One of the most popular architecture styles is the one known as Model-View-Controller or MVC architecture. This architecture has gained its reputation due to the benefits it provides, such as good scalability and modifiability. Since the model, the view, and the controller are all independent of each other, it offers developers the means to quickly adapt the system to new features and changes as needed.

The model contains all application-specific content and processing logic, including all content objects, access to external data/information sources, and all processing functionality that is application specific. The view contains all interface-specific functions and enables the presentation of content and processing logic, including all content objects, access to external data/ information sources, and all processing functionality required by the end user. The controller manages access to the model and the view and coordinates the flow of data between them. [[Pressman 2014](#Pressman2014)]

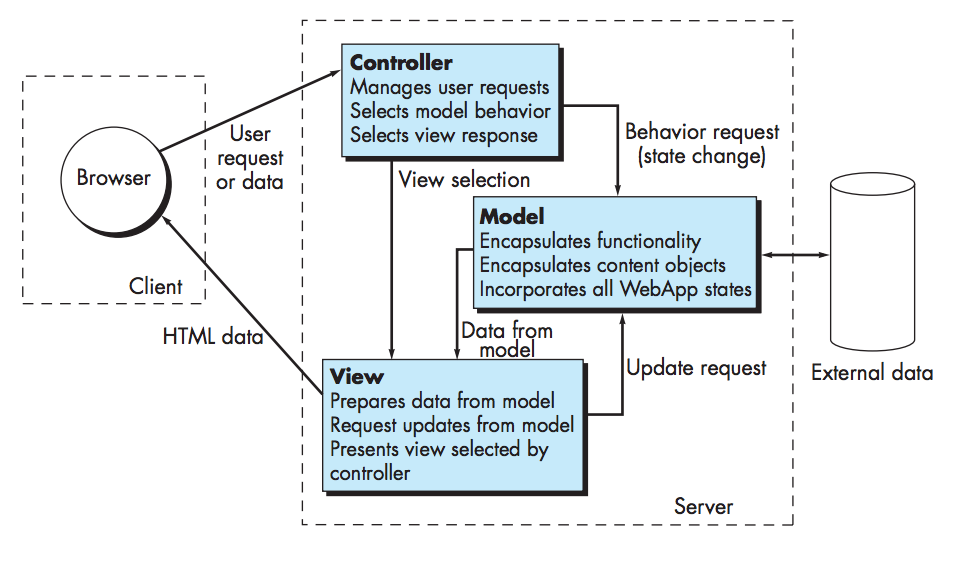


Figure X. Model View Controller (MVC) Architecture [[Pressman 2014](#Pressman2014)]

## VolunteerCloud’s Architecture

Given that VolunteerCloud is offered as a cloud software application, it falls under the paradigm of providing Software as a Service. Hence, VolunteerCloud’s software architecture consists of a mixture of the Client-Server architecture, and the Model-View-Controller architecture to benefit from the advantages these offer. As a consequence, VolunteerCloud shall display good maintainability, modifiability, and scalability.

Figure x shows VolunteerCloud’s software architecture; ***Label 1*** is comprised of the clients accessing the service through a web browser, and the user interface that they interact with when using VolunteerCloud. ***Label 2*** is comprised of the controllers that handle the communication between the client and business tiers. ***Label 3*** contains all of VolunteerCloud’s business logic. ***Label 4’s*** primary purpose consists of handling the data mapping between the business and data source tiers by the use of APIs such as JDBC and JPA. ***Label 5*** is comprised of VolunteerCloud’s MySQL relational database.



Figure X. MVC Mapping to Java EE Client-Server Architecture [[Balci 2016](#Balci2016)]

*Tier 1 - Client*: Java EE applications are unique in that not only do they support the typical web protocols like HTML, but they can also work in a dedicated client container to run Java code. Java EE can also be used as a RESTful API by returning data in XML or JSON, or any format for that matter, to the native client application to be displayed. Many of the libraries required to render JSF pages are loaded remotely as JavaScript components when the page is displayed on a browser. These libraries can either be hosted by the Java EE server or on a content delivery network (CDN) to minimize code reuse and load on a developer's infrastructure. A browser is not the only method of ingesting a Java EE web application as it can server as the backend implementation for any types of clients from smartphones to other web servers. [[Schutt and Balci 2016](#Schutt_And_Balci2016)]

*Tier 2 - Web*: The web tier for a Java EE web-application consists mainly of JSF pages (as defined by the expression language) and Context & Dependency Injection (CDI). CDI provided developers the ability to dynamically request modules as needed based on the incoming request. While JSF is not required by Java EE to serve content over the web, it is highly recommended as the standard method for generating HTML and JavaScript based on the incoming request since it abstracts data bindings and application logic by with expression language. The design of the JSF framework also values component reuse and extensibility which greatly reduces overall development work for the software engineer. [[Schutt and Balci 2016](#Schutt_And_Balci2016)]

*Tier 3 - Business*: The business tier contains all logic pertaining to a specific business need by implementing various Enterprise Java Beans (EJBs). These EJBs can then respond to specific requests to JAX-RS RESTful calls that perform some predefine function on the models. EJBs can also respond to specific events that originate from a JSF page or component. The Java Persistence API is also listed here because it is used to define the models that are represented in the database. The business tier in Java EE applications is responsible for the security and the state of the current application. These sessions are managed by the application on a client-by-client basis to improve the security and continuity of the business logic components. Finally, while Java EE is mostly a request-driven framework, it also contains modules that allow for event-driven processing and messaging between the server and the client called the Java Messaging Service. [[Schutt and Balci 2016](#Schutt_And_Balci2016)]

*Tier 4 – Data Mapping*: Java EE provides several different options for data mapping technologies including Java Database Connectivity (JDBC), Java Persistence API, Connector Architecture, and Transaction API. While each of these options relies on the base Java EE APIs for connecting to a data source, each of them play a unique role in the ecosystem. JDBC and the Java Persistence API are mainly used in the management of relational data objects by defining entities and generating SQL queries based on the class definition. The Connector Architecture is primarily geared toward enterprise customers looking to integrate legacy Java applications into a Java EE web application. Transaction API is a more fine-grained approach to JDBC by managing individual methods within an entity. [[Schutt and Balci 2016](#Schutt_And_Balci2016)]

*Tier 5 – Data Sources*: Java EE provides support for all major database management systems such as Oracle, DB2, SQLServer, MySQL, and PostgreSQL. Additionally, Java EE supports NoSQL data sources like mongodb. Like other web-application frameworks, Java EE is fairly agnostic to the systems implemented as the data source. Modern data mapping technologies, like Java Persistence API, are generic enough in definition to be extended to work with any kind of data source, object- or document-based. The type of data source utilized by a web-application is entirely dependent on the requirements and expertise of the developers involved. The choice of relational or NoSQL is left to the software engineer since there has yet to be a unified consensus on the appropriate usage of data sources for a given application. [[Schutt and Balci 2016](#Schutt_And_Balci2016)]

## The Department of Defense and Architecture Framework (DoDAF)

The Department of Defense Architecture Framework (DoDAF) is the overarching, comprehensive framework and conceptual model for architectural descriptions developed within the DoD. This framework helps Department of Defense (DoD) managers at all levels make effective decisions by ensuring the sharing of consistent and common information across the Department, Joint Capability Areas (JCAs), missions, components, and programs. The DoDAF helps the DoD Chief Information Officer (CIO) develop and maintain architectures required by the Clinger-Cohen Act. It also fulfills guidance from the Office of Management and Budget (OMB) and other Departmental directives and instructions. [[DoDAF](#DoDAF)]

### OV-2: Operational Resource Flow Description

The following OV-2 model below (***Figure x)*** illustrates the resources flow in VolunteerCloud. This model is characterized as being a description of the resource flows exchanged between operational activities. [[DoDAF](#DoDAF)]

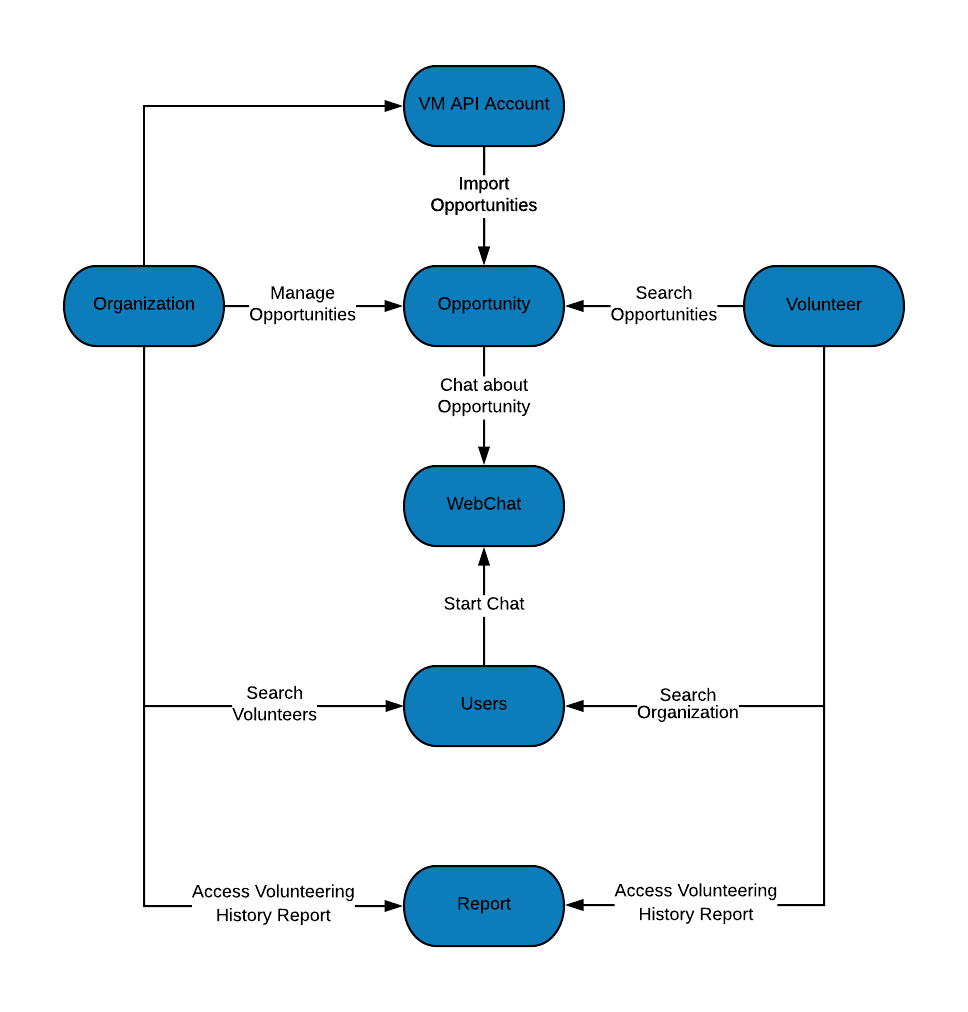
After creating an account in VolunteerCloud; 1) Organizations can link their VolunteerCloud account to their VM account if desired. 2) Organizations can manage and import their owned opportunities from VolunteerMatch. 3) Volunteers can search for opportunities. 4) Volunteers can search for organizations. 5) Organizations can search for volunteers. 6) Volunteers and organizations can start a webchat. 6) Organizations and volunteers can access their volunteering activity history.

Figure X. OV-2: Operational Resource Flow Description

### SV-1: Systems Interface Description

The following SV-1 model below (***Figure x)*** illustrates VolunteerCloud’s system interface. This model is characterized for identifying an applications’ systems, and their interconnections. [[DoDAF](#DoDAF)]

VolunteerCloud has a “Control System” that consists of the application’s controllers which hold data and execute methods during the life of the session. The control system communicates with the different classes and their resources in the database, which the user views and interacts with through the user interface. In addition, VolunteerCloud communicates externally with the VolunteerMatch’s, Twilio’s, and ZipCode’s APIs in order to provide the *webchat* and *opportunity import* features. Finally, users can interact with VolunteerCloud through its user interface.

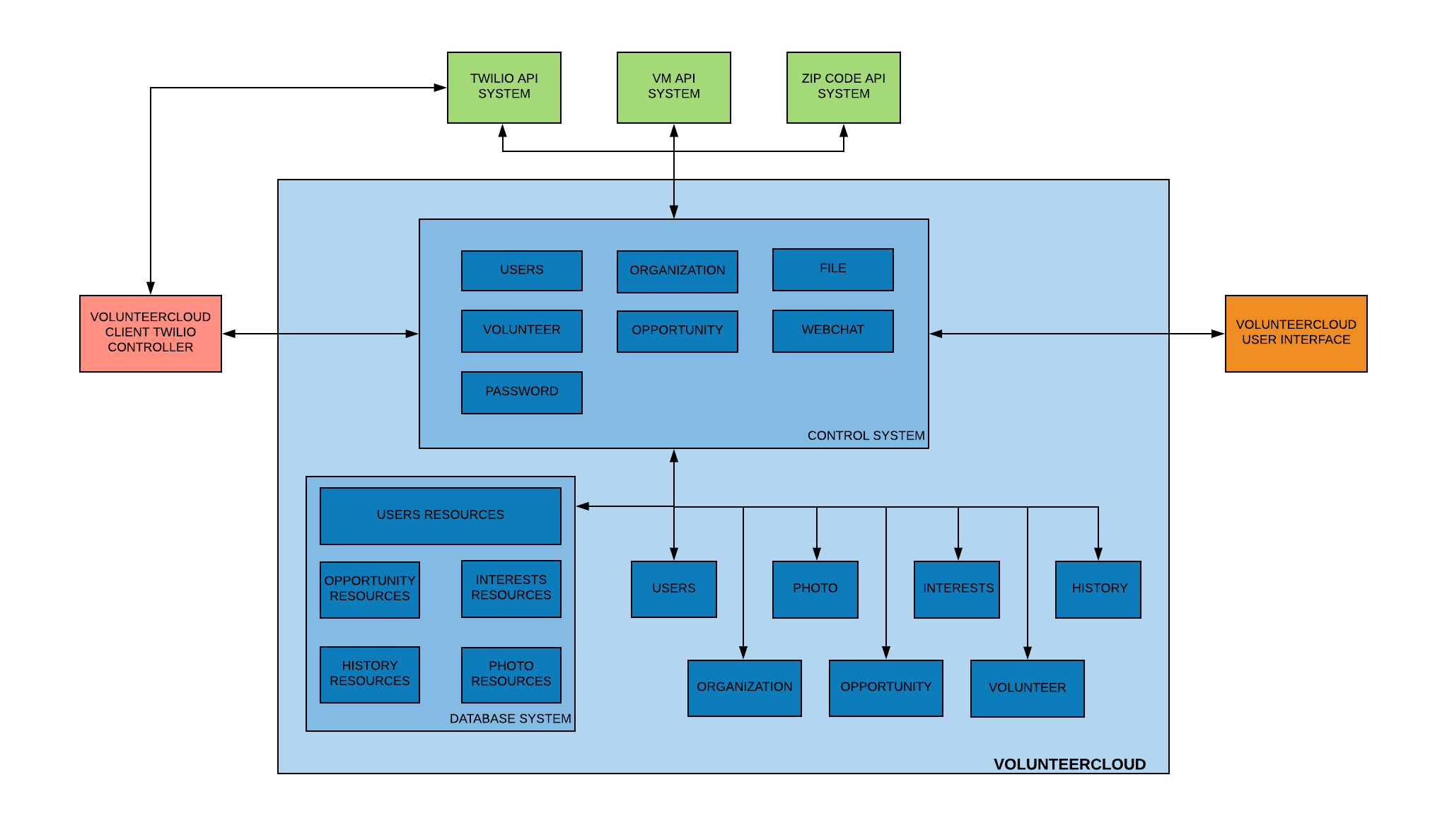


Figure X. SV-1: Systems Interface Description

### SV-4: Systems Functionality Description

The following SV-4 model below (***Figure x)*** depicts VolunteerCloud’s functions. This model is characterized for describing the functions performed by systems and the system data flows among system functions. [[DoDAF](#DoDAF)]

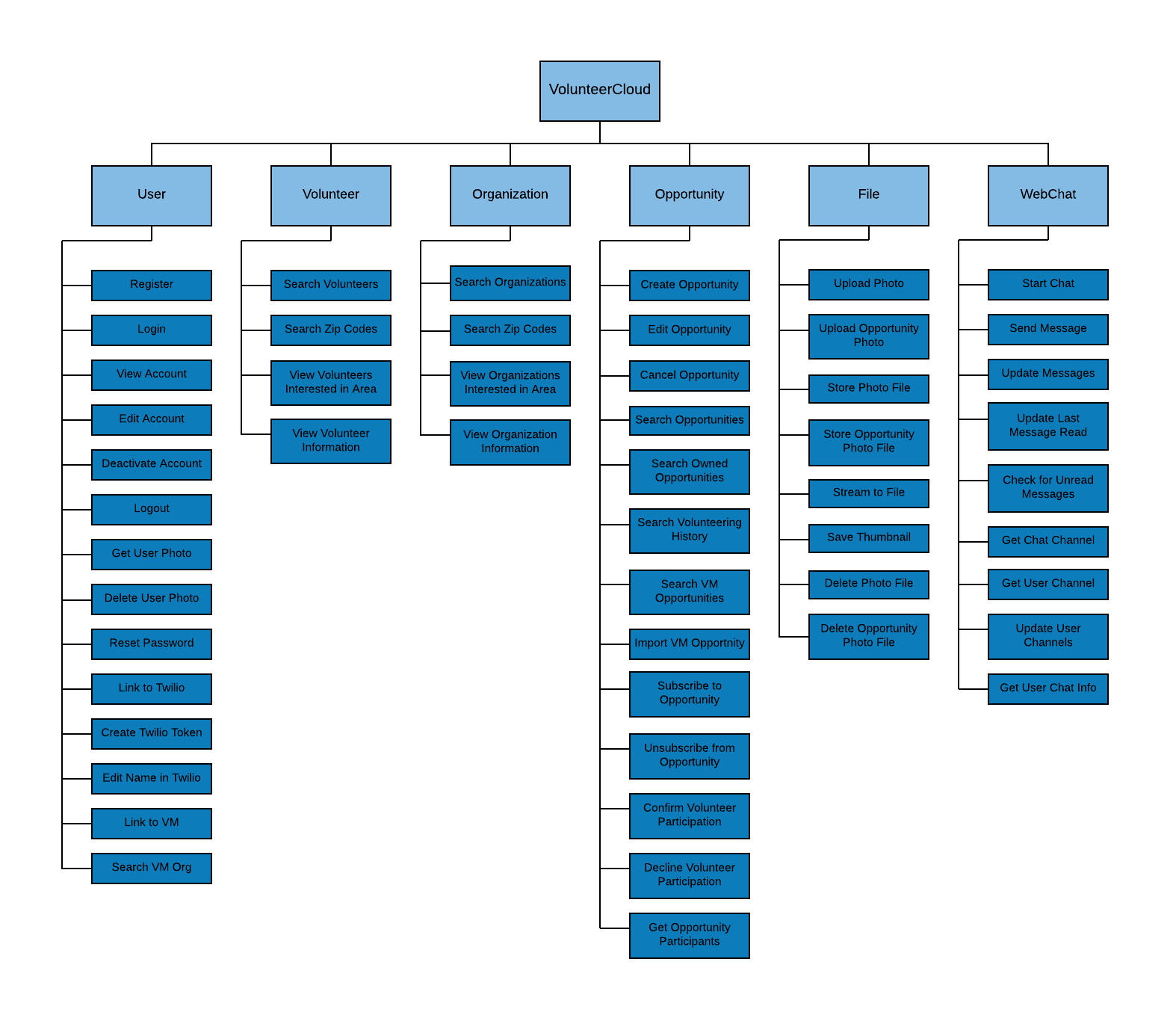
VolunteerCloud has a total of 6 modules where each of them has their own functions; 1) User module: register, login, view account, edit account, deactivate account, get photo, delete photo, logout, reset password, link to Twilio, create Twilio token, edit name in Twilio, link to VM, search VM organization account, and manage interests. 2) Volunteer module: search volunteers, search zip codes, view volunteers interested by area, and view volunteer information. 3) Organization module: search organizations, search zip codes, view organizations interested by area, and view organization information. 4) Opportunity module: create opportunity, edit opportunity, cancel opportunity, search opportunities, search owned opportunities, search volunteering history, search VM opportunities, Import VM opportunity, subscribe/unsubscribe from opportunity, confirm/decline volunteer participation, and get opportunity participants. 5) File module: upload photo, upload opportunity photo, store photo file, store opportunity photo file, stream to file, save thumbnail, delete photo file, and delete opportunity photo file. 6) WebChat module: start chat, send message, update messages, update last message read, check for unread messages, get chat channel, get user channel, update user channels, and get user’s chat information.

Figure X. SV-4: Systems Functionality Description

# VolunteerCloud Design Specification

The objective of Unified Modelling Language is to provide system architects, software engineers, and software developers with tools for analysis, design, and implementation of software based systems as well as for modeling business and similar processes. [[OMG 2011](#OMG2011)]

## Package Diagrams

Package diagrams are a subset of class diagrams. They organize elements of a system into related groups to minimize the amount of dependencies. One can think of package diagrams as a higher-level view of a system when compared to class diagrams [[Balci 2009](#Balci2009)].

### VolunteerCloud Package Diagram

The package diagram (***Figure 1***) below illustrates the packages that compose the VolunteerCloud system, and it illustrates the dependencies among them. In VolunteerCloud, the *entityclasses* package is dependent on the *sessionbeans* and *controllers* packages which may require adaptations if any changes to the entity classes occur.

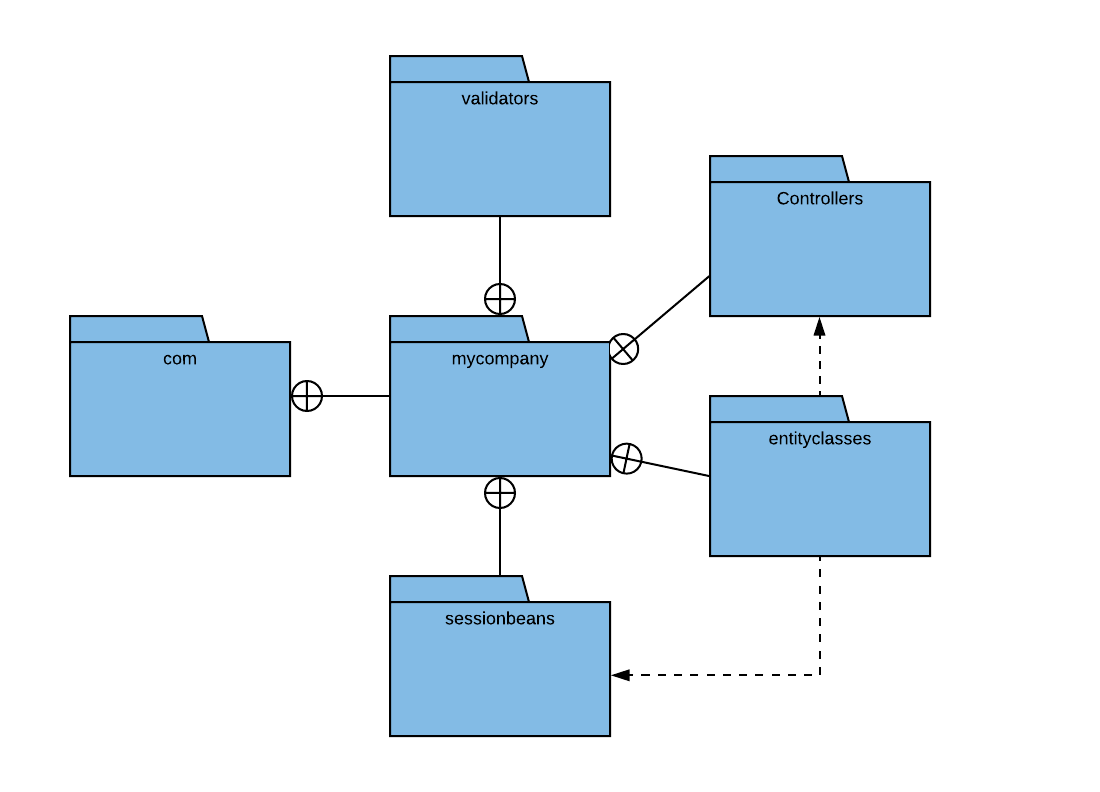


Figure . VolunteerCloud Package Diagram

## Component Diagrams

A component diagramhelps you visualize the high-level structure of the system, and the service behavior that those pieces provide and consume through interfaces [[Microsoft 2009](#Microsoft2009)].

### VolunteerCloud Component Diagram

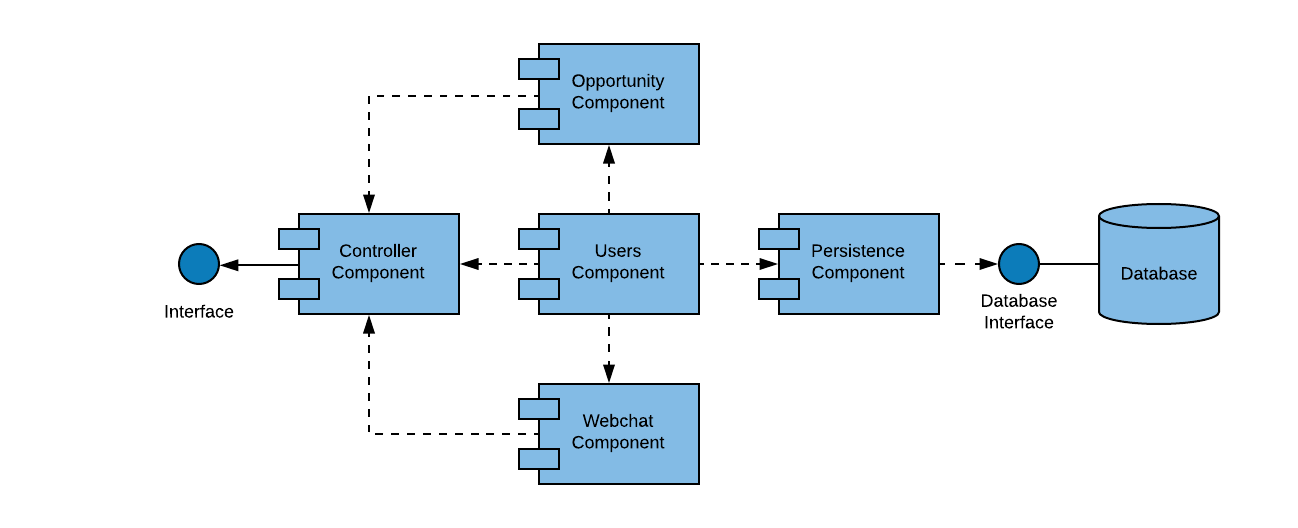
The component diagram (***Figure 2***) below illustrates the components that compose VolunteerCloud. In VolunteerCloud, the *Users Component* is dependent on all the other components as this is a user-based system. In addition, both the *Opportunity Component* and the *Webchat Component* are dependent on the *Controller Component* as any changes to these will affect the way in which the controller conducts its logic.

Figure 2. VolunteerCloud Component Diagram

## Class Diagrams

The class diagram is a static model that supports the static view of the evolving system. It shows the classes and the relationships among the classes that remain constant in the system over time [[Dennis 2012](#Dennis2012)].

### VolunteerCloud Class Diagram

The class diagram (***Figure 17*** ) below shows the entity classes that compose the VolunteerCloud system with their attributes, methods, and relationships. The *Organization* and *Volunteer* classes inherit from the *Users* class. The *VolunteeringInterest* class has a composite relationship with both the *Organization* and *Volunteer* classes, as it would not be able to exist on its own without them. In a similar fashion, the *VolunteeringHistory* class also holds a composite relationship with the *Volunteer* and *VolunteeringOpportunities* classes, as it depends on them to exist. Finally, the *VolunteeringOpportunities* class depends on the *Organization* class’ existence too.

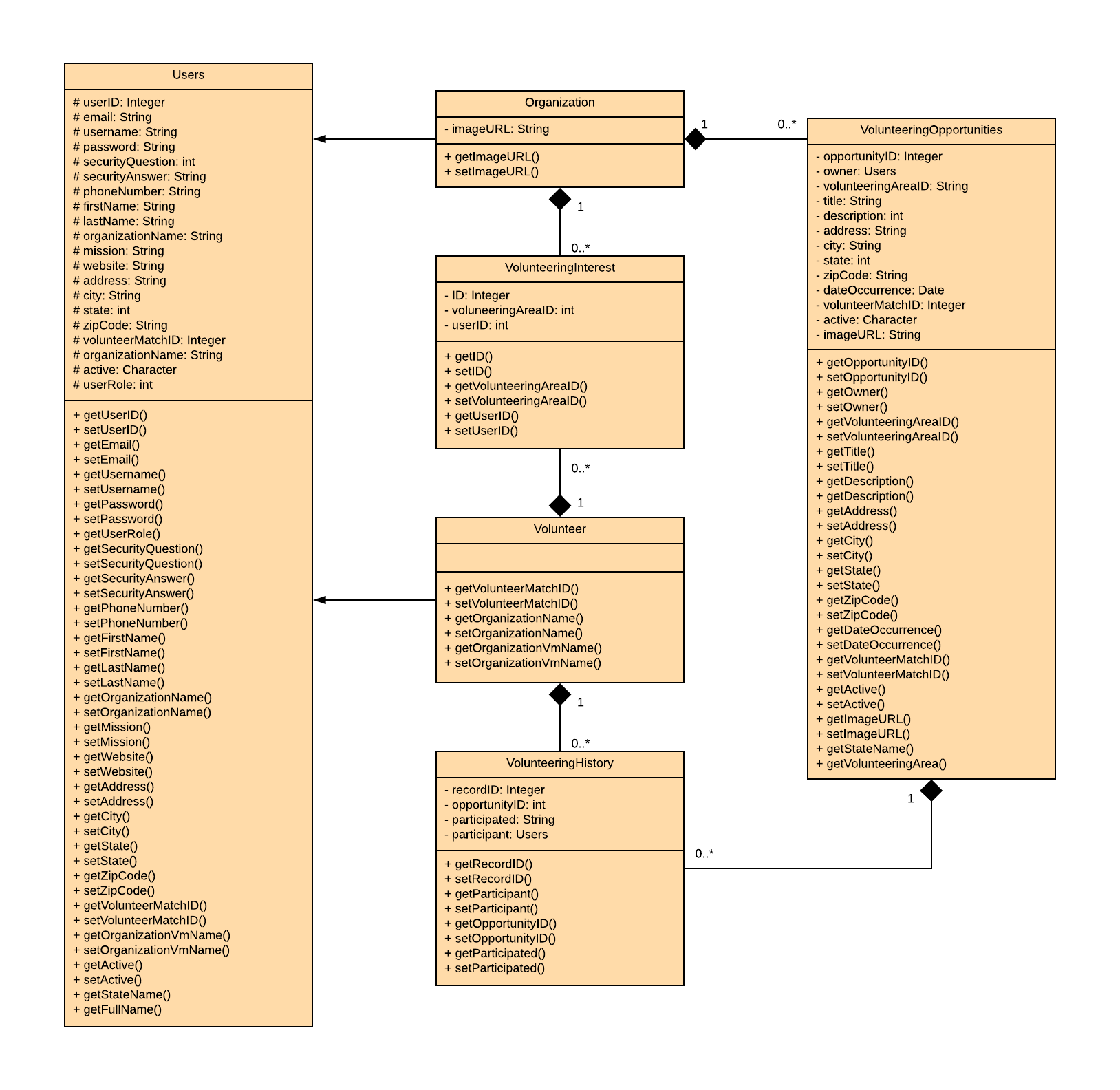


Figure 17. VolunteerCloud Class Diagram

## Statechart Diagrams

State diagrams or statechart diagrams are used to describe the behavior of a system by showing all the possible states of an object as events occur. Each diagram usually represents objects of a single class and tracks the different states of its objects through the system [[Balci 2009](#Balci2009)].

### Volunteer Statechart Diagram

The statechart diagram (***Figure 20***) below depicts the behavior a volunteer can engage in when using VolunteerCloud. After a successful sign-in, volunteers can either search for opportunities, organizations, or manage their activity. Once the volunteers have found an organization, they can start a chat with them. Similarly, if a volunteer finds and subscribes to an opportunity, the volunteer can inquire more information about the event by chatting with the corresponding organization. Finally, the volunteers can manage their activity by viewing it, updating their volunteering interests, and exporting their history as a PDF document.

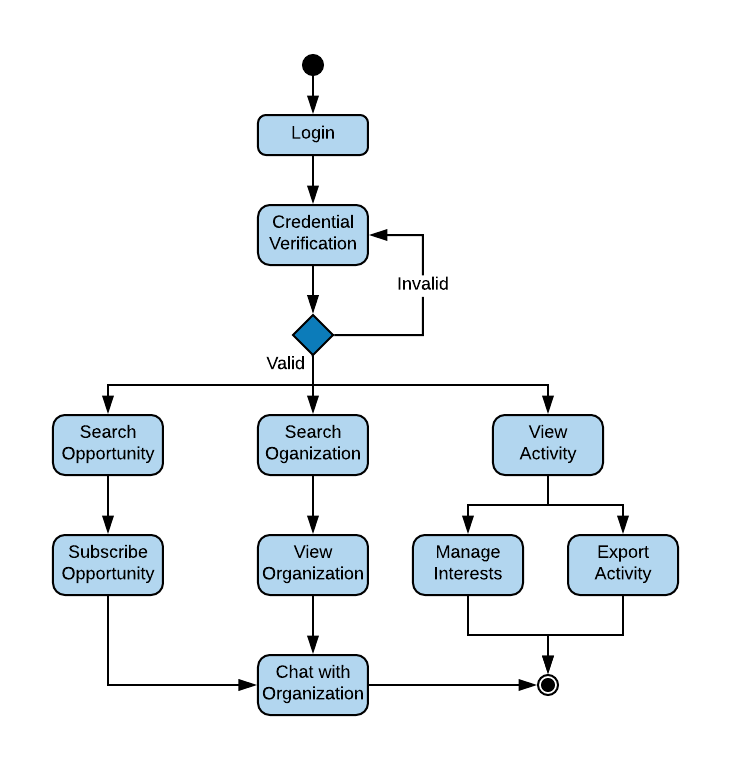


Figure 20. Volunteer Statchart Diagram

### Organization Statechart Diagram

The statechart diagram (***Figure 21***) below depicts the behavior an organization can engage in when using VolunteerCloud. After a successful sign-in, organizations can either search for volunteers, manage their opportunities, or manage their activity. Once the organizations have found potential volunteers, they can start a chat with them. Organizations can also manage volunteering opportunities by creating and canceling them, modifying them, and confirming or denying a volunteer’s participation in it. Finally, organizations can manage their activity by viewing it, updating their volunteering areas of interest, and exporting their history as a PDF document.

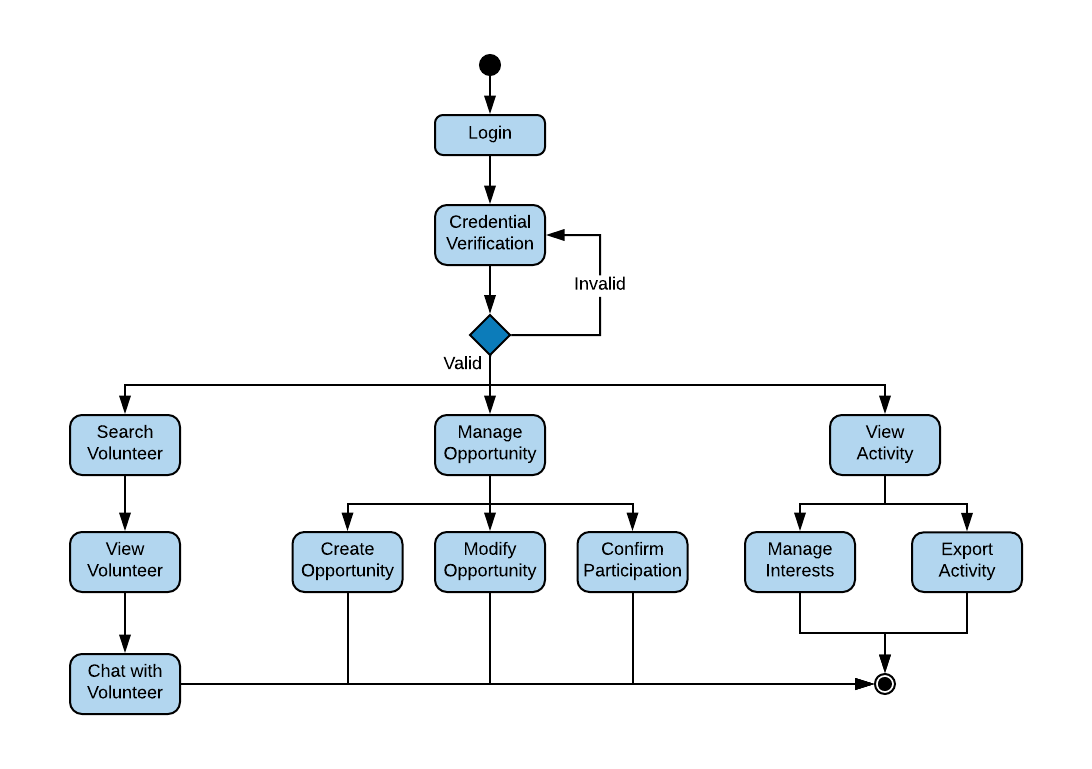


Figure 21. Organization Statchart Diagram

## Activity Diagrams

Activity diagrams describe the workflow behavior of a system. The diagrams describe the state of activities by showing the sequence of activities performed. Activity diagrams can show activities that are conditional or parallel, providing a way to model the workflow of a process similar to a flowchart. An activity diagram can also be used to model code-specific information such as a class operation [[Balci 2009](#Balci2009)].

### Organization-Volunteer Activity Diagram

The activity diagram (***Figure 22*)** describes the sequence of activities held between an organization and a volunteer in VolunteerCloud. First, an organization creates an opportunity, a volunteer views it, subscribes to it, and reaches out to the organization to inquire more information through chat. After the volunteering opportunity has taken place, the organization can confirm or decline the volunteer’s participation in the event.

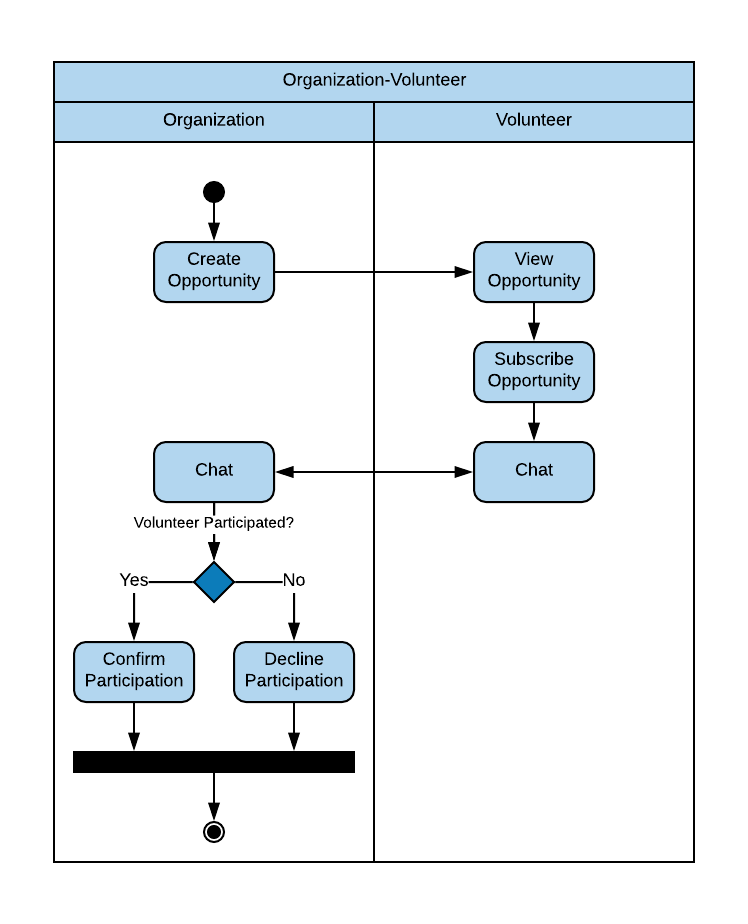


Figure 22. Organization-Volunteer Activity Diagram

## Sequence Diagrams

A sequence diagram is a dynamic model that supports a dynamic view of the evolving systems. It shows the explicit sequence of messages that are passed between objects in a defined interaction. Since sequence diagrams emphasize the time-based ordering of the activity that takes place among a set of objects, they are very helpful for understanding real-time specifications and complex use cases [[Dennis 2012](#Dennis2012)].

### Search Volunteer Sequence Diagram

The sequence diagram (***Figure 23*)** below depicts the time-based interaction that an organization engages in when searching for volunteers. First, the organization specifies a volunteer search criteria, and a list of results matching the description is returned. Then, the organization selects a volunteer, and detailed information about the volunteer is displayed. Finally, the organization starts a chat with the volunteer where they can discuss more about interests and opportunities.

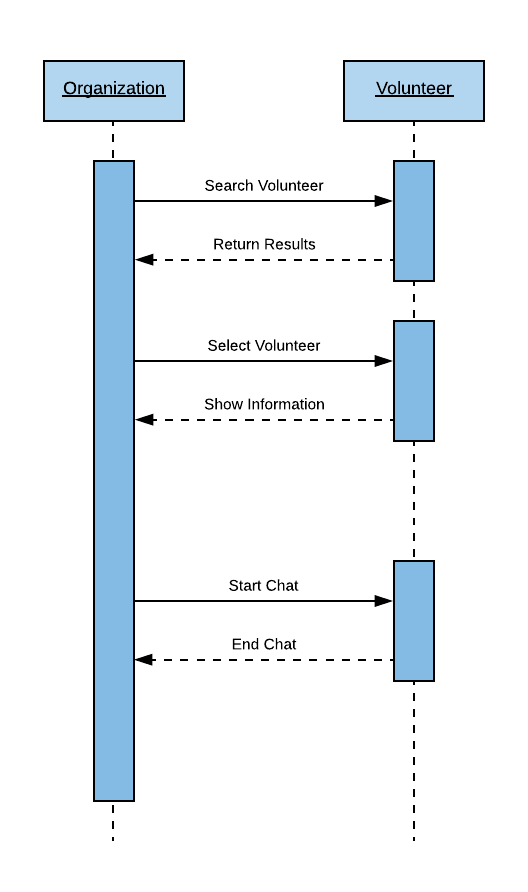


Figure 23. Search Volunteer Sequence Diagram

### Subscribe to Opportunity Sequence Diagram

The sequence diagram (***Figure 24*)** below depicts the time-based interaction that a volunteer engages in when subscribing to an opportunity. First, the volunteer specifies a volunteering opportunity search criteria, and a list of results matching the description is returned. Then, the volunteer selects an opportunity, and detailed information about the opportunity is displayed. After identifying an opportunity of interest, the volunteer starts a chat with the organization to inquire more information about the event. Finally, the volunteer subscribes to the opportunity and his/her volunteering activity is updated.

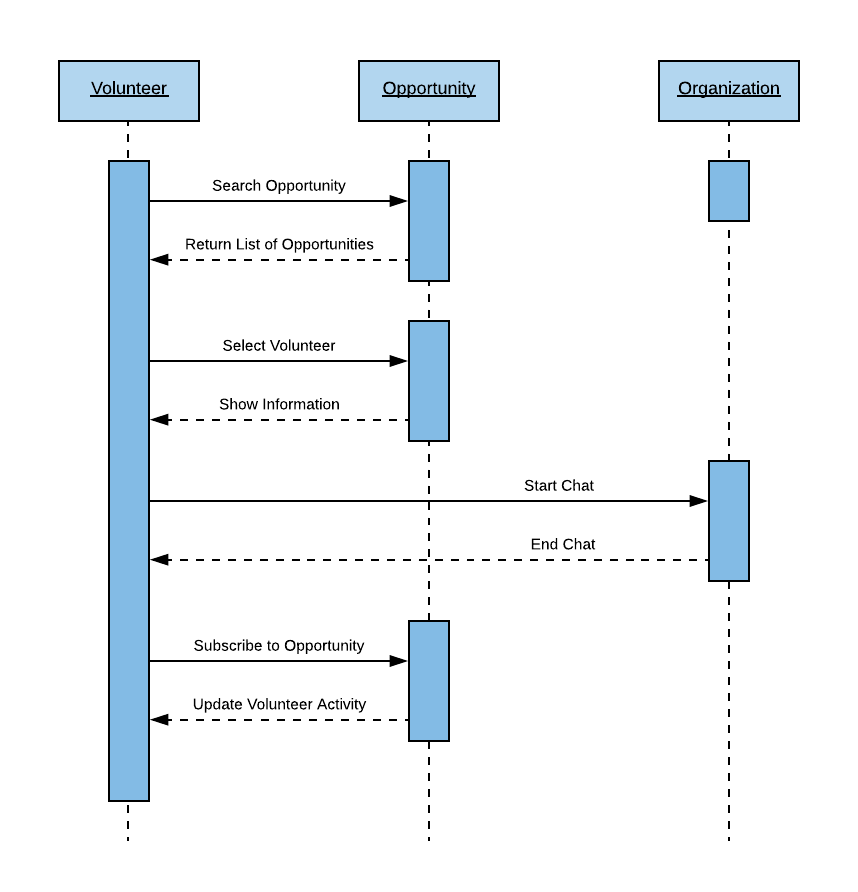


Figure 24. Subscribe to Opportunity Sequence Diagram

## Collaboration Diagrams

A collaboration diagram is an interaction diagram that shows the order of messages that implement an operation or a transaction. Collaboration diagrams show objects, their links, and their messages [[Balci 2009](#Balci2009)].

### Search Volunteer Collaboration Diagram

The collaboration diagram (***Figure 25***)below depicts the order of messages that implement the *Search Volunteer* operation; 1) The organization searches volunteers, 2) The organization selects a volunteer, 3) The organization starts a chat with the volunteer of interest, 4) And the organization finishes the chat after the conversation comes to a conclusion.

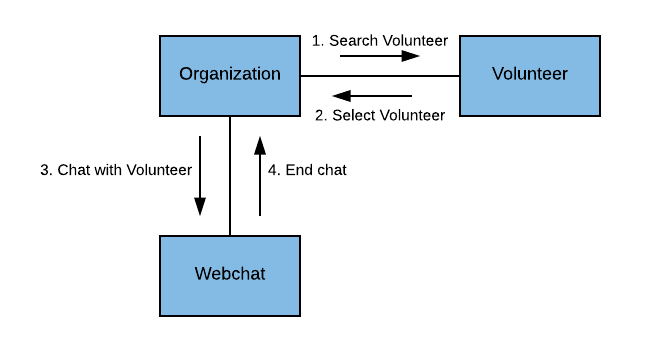


Figure 25. Search Volunteer Collaboration Diagram

### Subscribe to Opportunity Collaboration Diagram

The collaboration diagram (***Figure 26*)** below depicts the order of messages that implement the *Subscribe to Opportunity* operation; 1) The volunteer searches opportunities, 2) The volunteer selects an opportunity, 3) The volunteer starts a chat with the organization of interest, 4) The volunteer finishes the chat after the conversation comes to a conclusion. 5) And the volunteer subscribes to the opportunity.

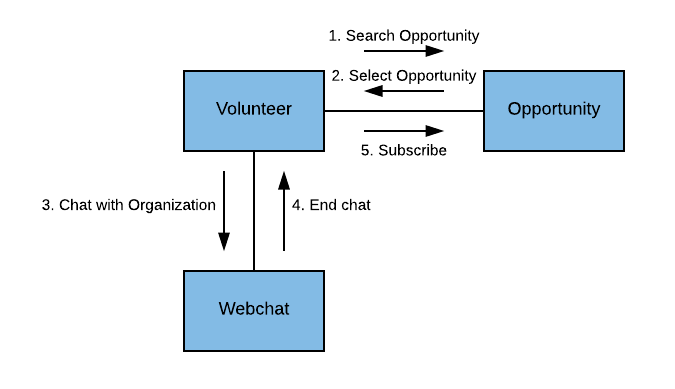


Figure 26. Subscribe to Opportunity Collaboration Diagram

## Deployment Diagrams

A deployment diagram shows processors, devices, and connections [[Balci 2009](#Balci2009)].

### VolunteerCloud Deployment Diagram

The deployment diagram (***Figure 27***) below indicates how the VolunteerCloud system is to be deployed. The application server hosts the VolunteerCloud CSA with all its modules. In addition, the application server also hosts the RDBMS database that holds all of VolunteerCloud’s data. Finally, the application server uses HTTP and HTTPS to communicate with the client’s computers and mobile devices, as well as the APIs used to provide the service.

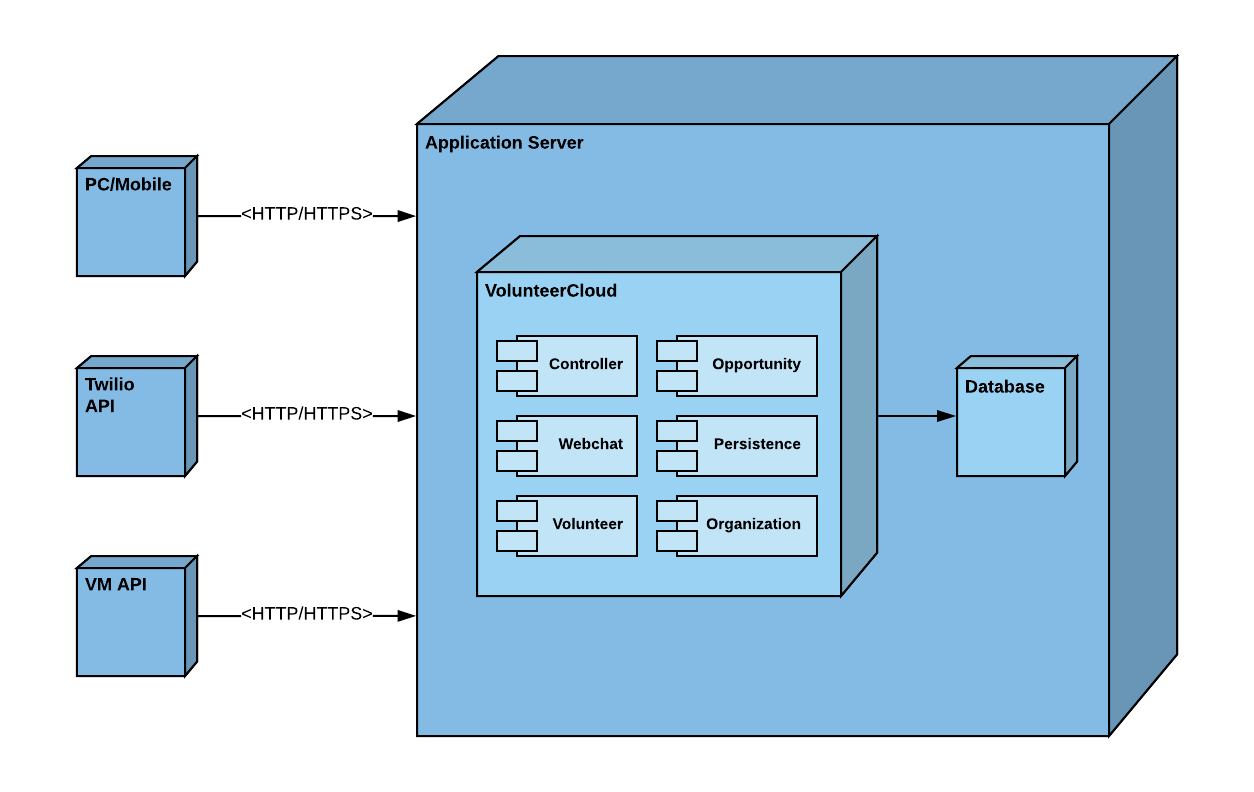


Figure 27. VolunteerCloud Deployment Diagram

# VolunteerCloud Functionality

This chapter describes the developed functionality of ARENAS. There are three user roles in this system: instructor, student and administrator. Each user role performs different functionalities. The following section will present the developed functionality for these use roles.

## User Registration

A regular user can register to be an instructor or a student. A user needs to provide the following required information: user name, user type, email and major field. The admin account is set by default in the system (***Figure 28.*** User Registeration Interface).

## 

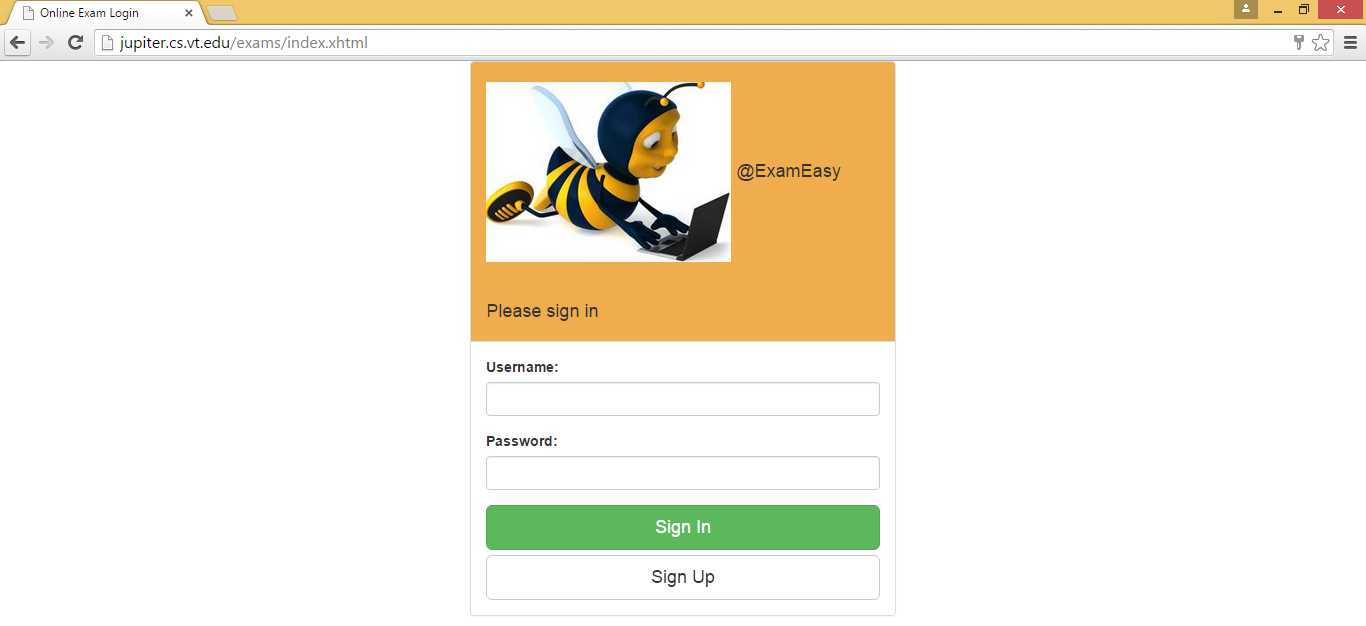


Figure 28. User Registeration Interface

## Question Repository

An instructor can create, delete, edit or view a question repository. When creating a question repository, the following information is required: repository name, description, and published or not (***Figure 29.*** Question Repository Interface.

## 

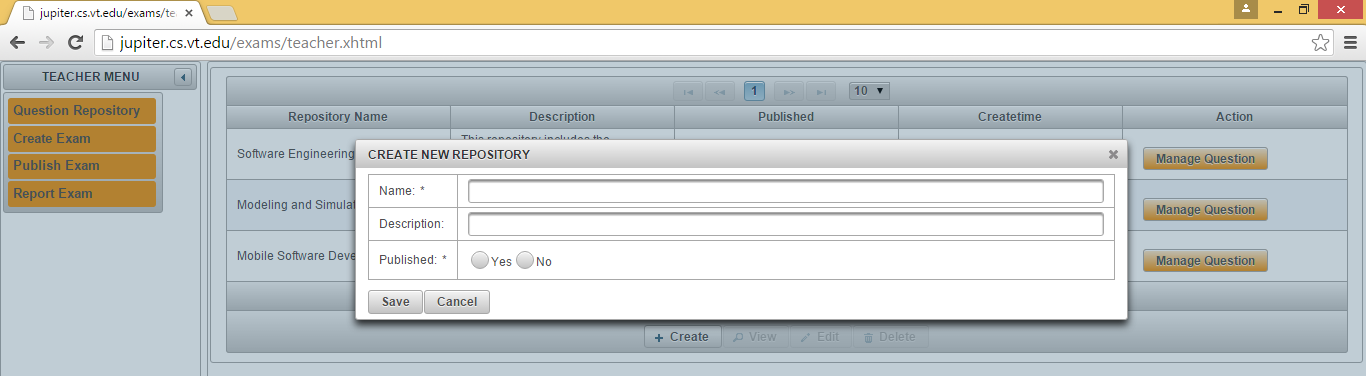


Figure 29. Question Repository Interface

## Question Management

## After a question repository is created, an instructor can create a question for the repository, delete a question from the repository, edit or view a question in the repository. Currently, three types of questions: single-choice question, multiple-choice question and essay/short answer question can be added into a question repository. When creating a question, the following information is required: question type, question title, question content, question options (only for single-choice question or multiple-choice question), points and published or not.

## When a question is created, an instructor can choose to add multimedia elements such as image, graphic, video or audio; these multimedia elements can be uploaded into the Server in which ARENAS is running if the instructor selects the “upload” choice. If the instructor selects “embedded” choice, he or she can copy a URL which directs a video in Youtube website and clicks “Submit URL” button, the video will be automatically embed in the question content.

## When creating a single-choice question or multiple-choice question, an instructor also needs to mark the correct answers, which will be stored into the databased for auto-grading and generating an exam report

## *(Figure 30. Question* Management Interface



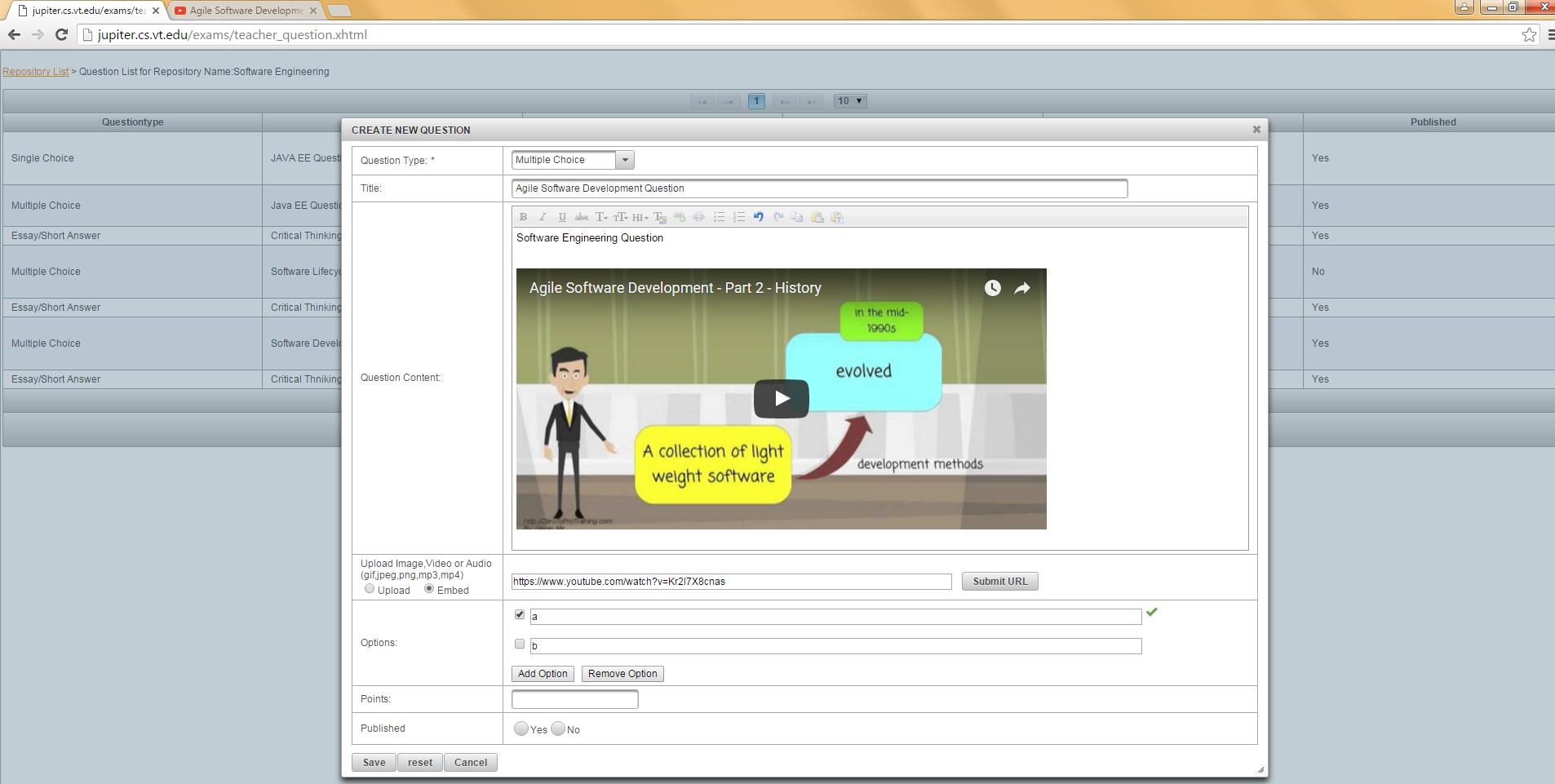
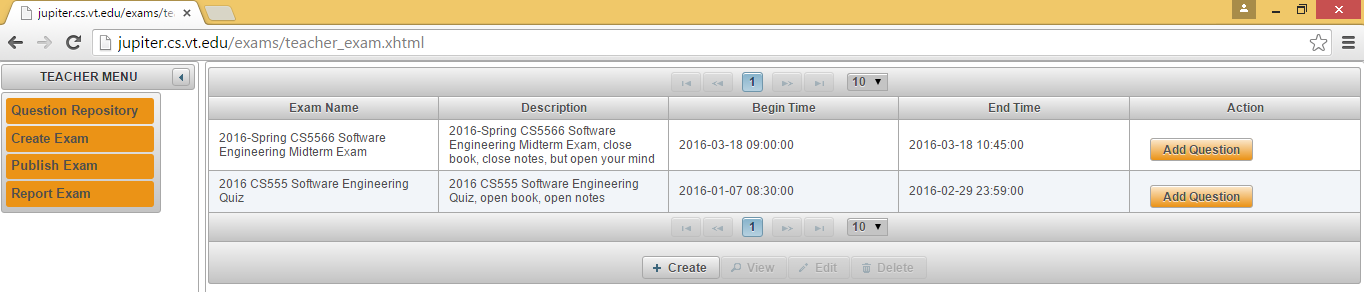


Figure 30. Question Management Interface

## Exam Creation

An instructor can create, edit, delete or view an exam anytime. The following information is required when creating an exam: exam name, exam description, beginning time and ending time. When setting the exam time, the exam time information will be formatted into YYYY-MM-DD HH:MM:SS at the backend ***(Figure 31.*** Exam Creation Interface).



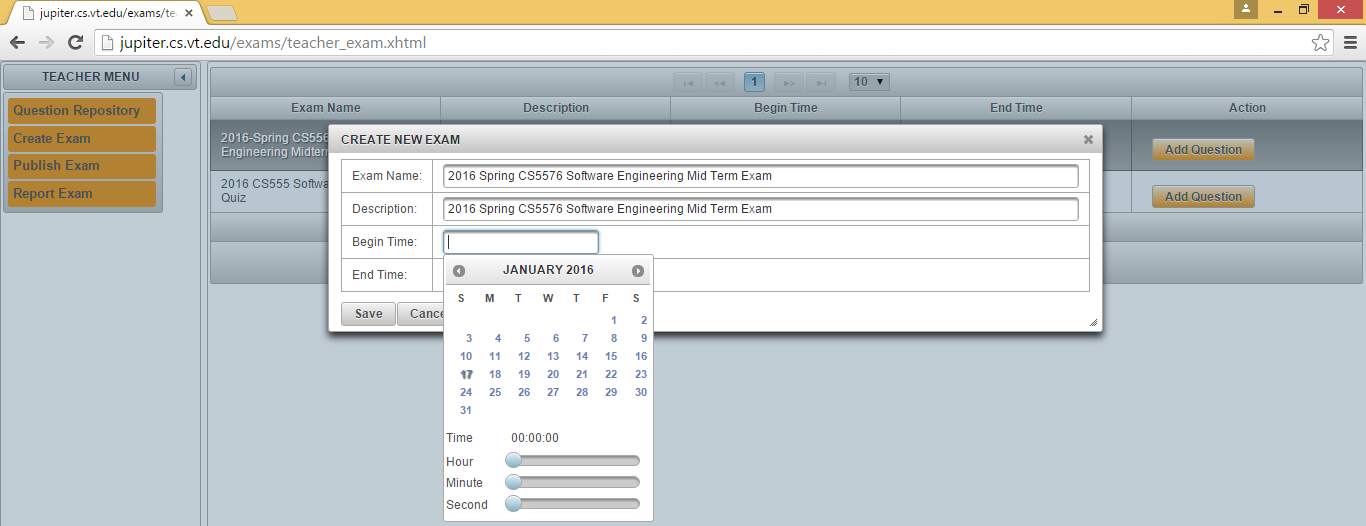
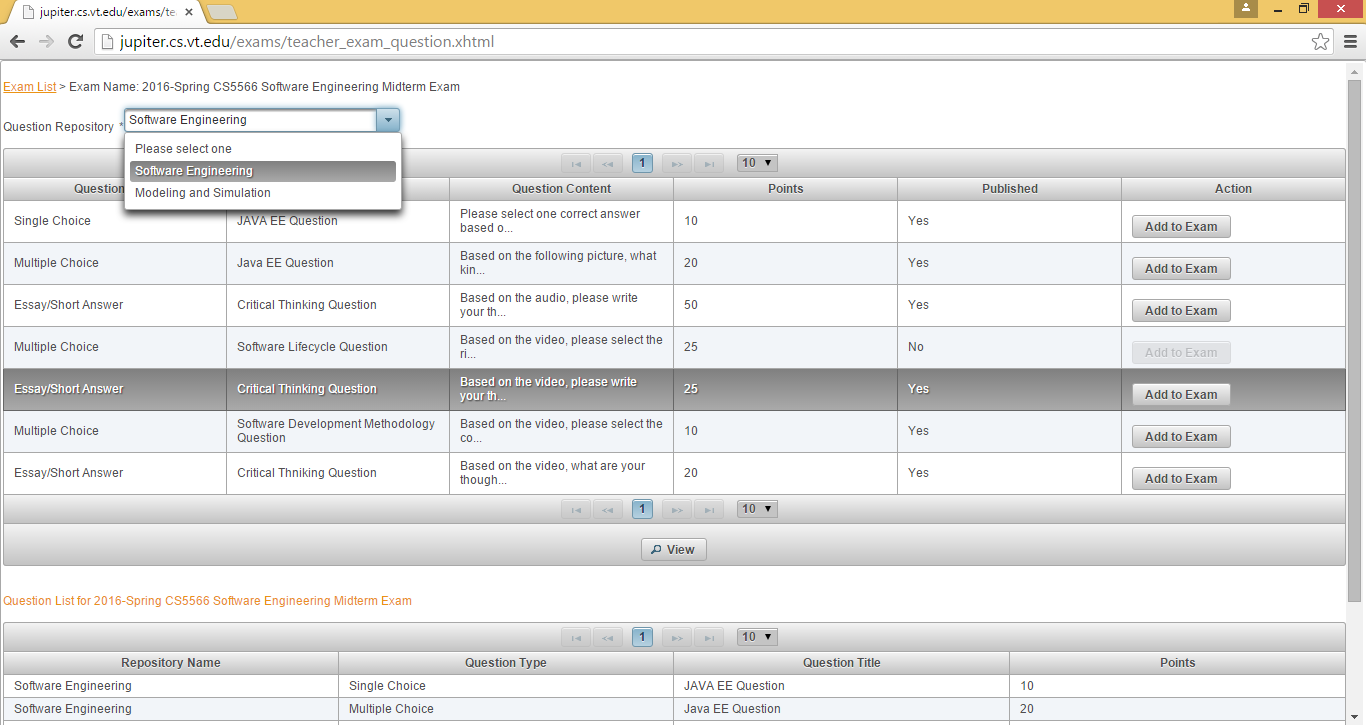


Figure 31. Exam Creation Interface

## Exam Question Management

After an exam is created, an instructor needs to select questions from the question repositories. Only the published question repositories will be opened for the instructor to choose from. If a question repository is published but one question in the question repository is not published, the unpublished question cannot be selected. After a question is added into an exam, the instructor can edit, delete or view that question. In addition, the instructor can choose to edit the points for that question which is added into the exam; otherwise, the default points for that question will be used ***(Figure 32.*** Exam Question Management Interface).



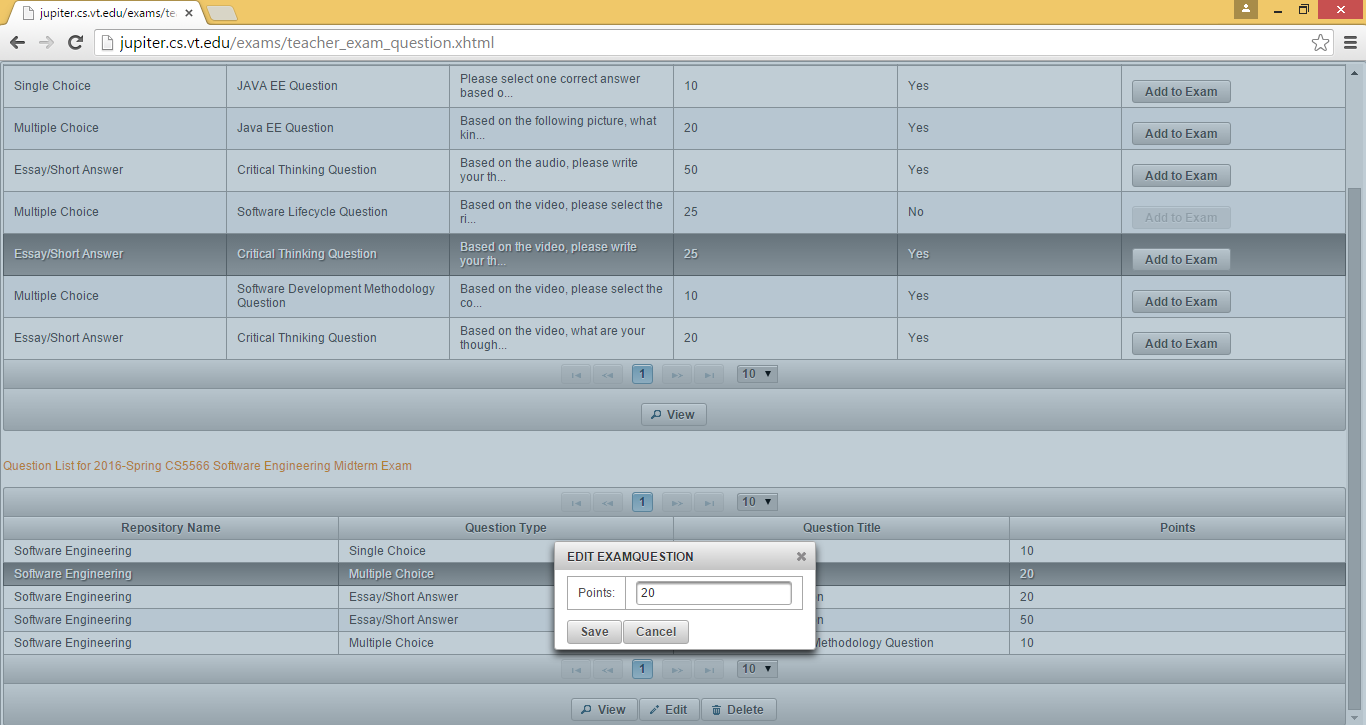
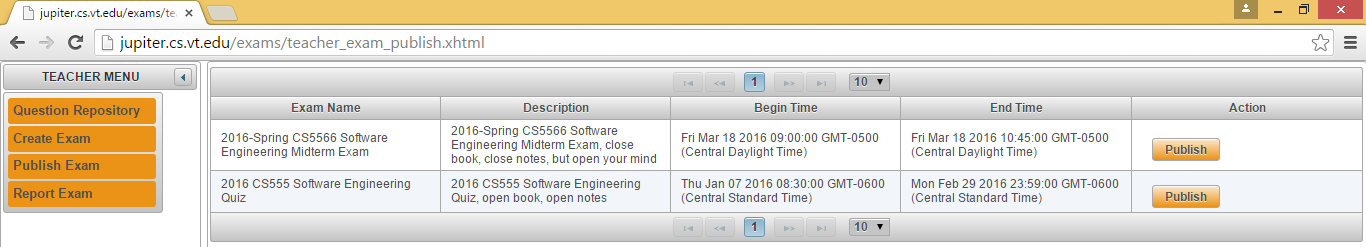
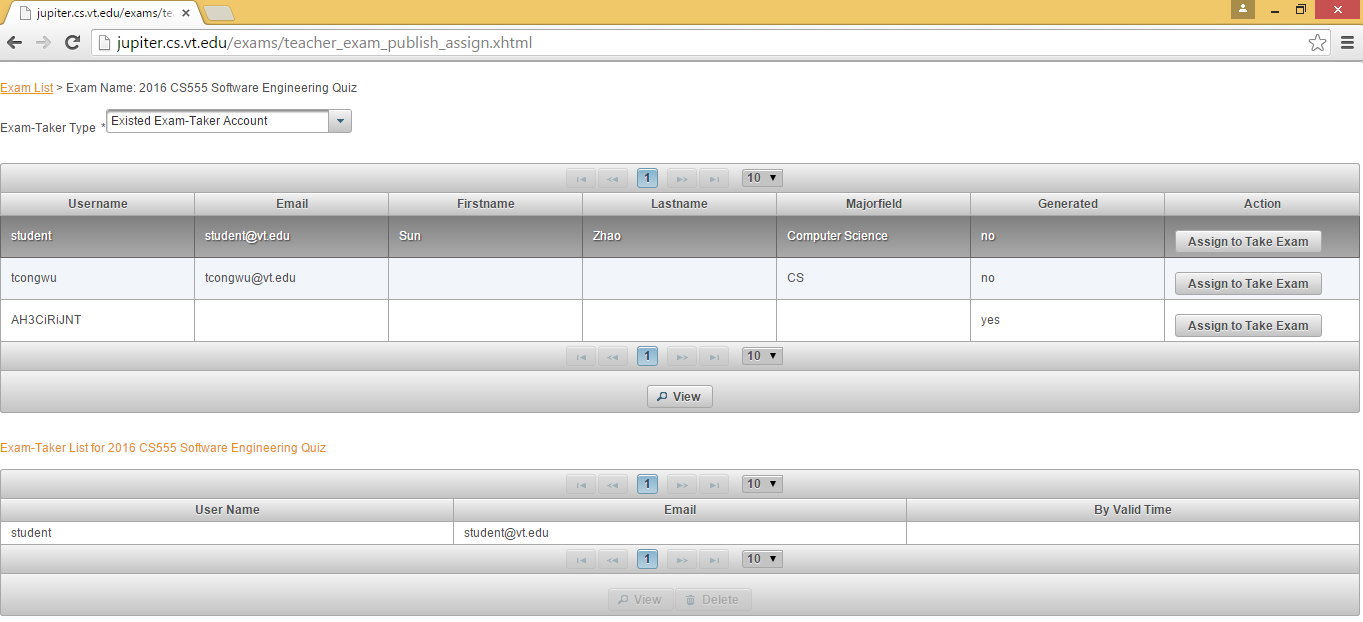


Figure 32. Exam Question Management Interface

## Publish Exam

After an exam is created by an instructor, the instructor can choose to publish the exam to existing student accounts or randomly generated student accounts. When the instructor publishes an exam to randomly generated student accounts, he or she can set the number and valid time for the generated student accounts. A student can use a generated account to log into the system to take an exam directly during the valid period without registering the system. A generated account is locked beyond the valid period ***(Figure 33***).





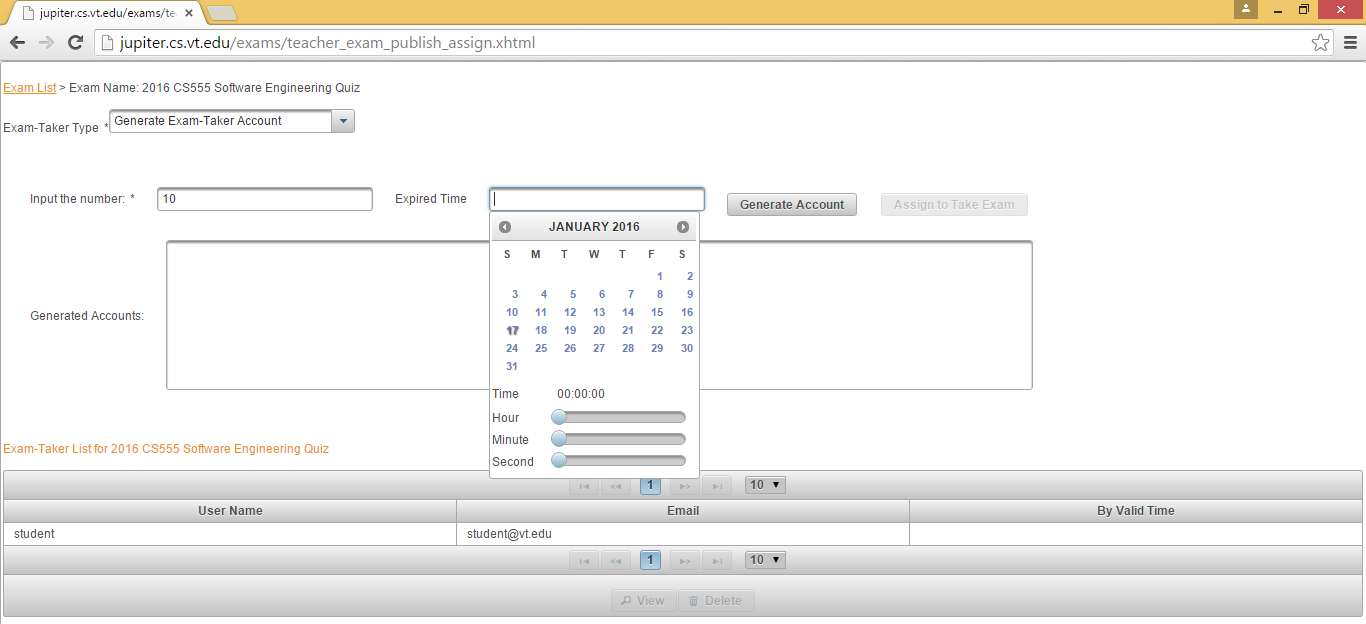
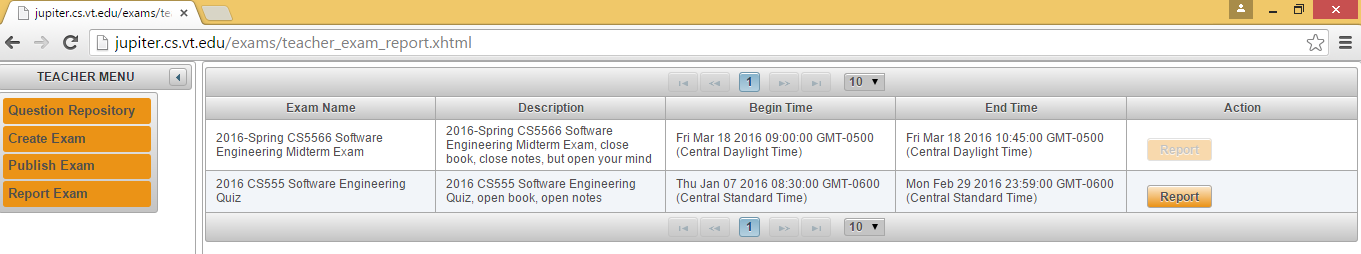
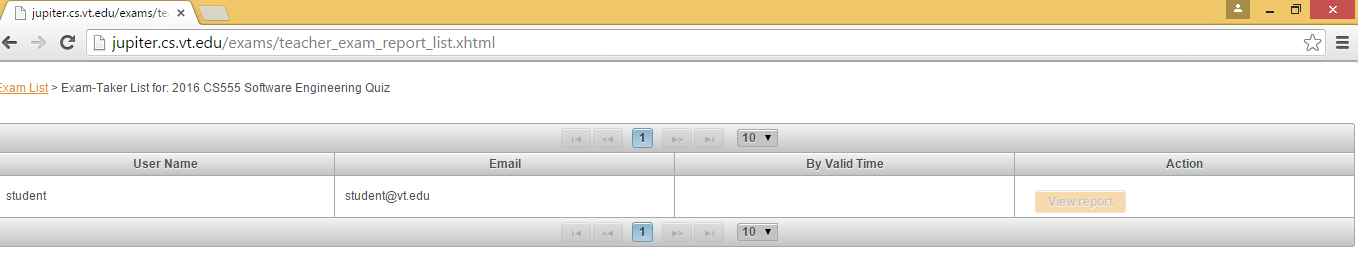


Figure 33. Publish Exam Interface

## Report Exam

After a student user completes an exam, an exam report will be automatically generated in the system. An instructor can tweak the report and write a summary feedback in the report and publish it to the student. If the exam includes essay or short answer questions, it requires the instructor to manually grade such type of questions (***Figure 34***).





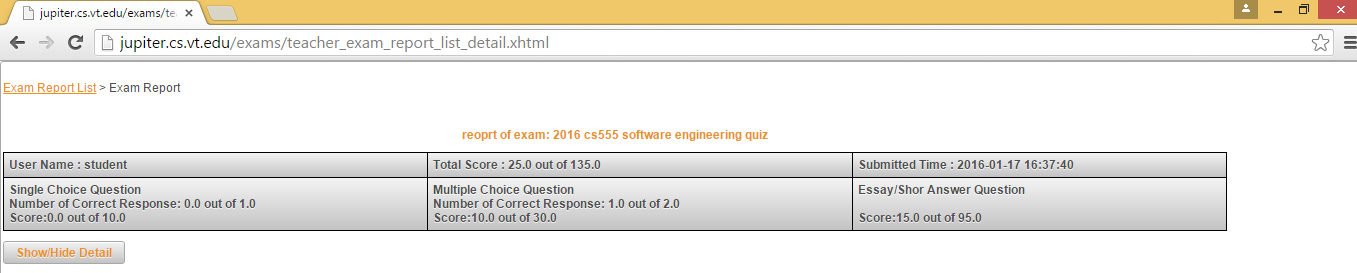
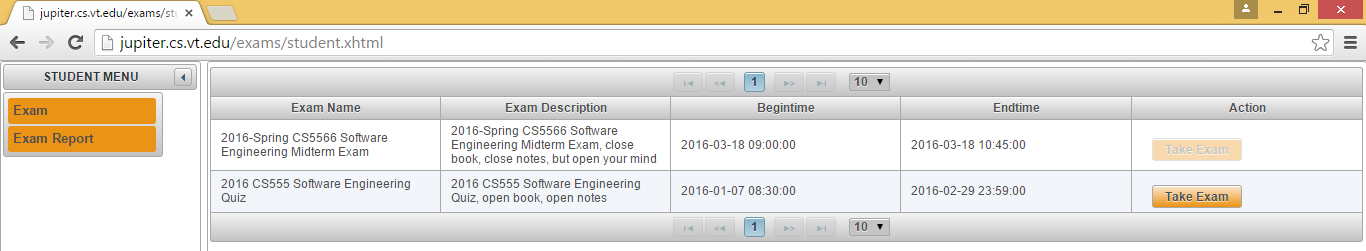




Figure 34. Report Exam Interface

## Take Exam

A student with a valid account can take an exam when the exam is open to him or her. When a student begins to take an exam, he or she can choose to hide or show the countdown timer. For answering any question in the exam, the student needs to click the “Confirm” button to submit his or her answer to that question. After the student completes all of questions in the exam, he or she needs to check a checkbox at the bottom of the webpage to indicate that he or she consents to submit the answers and the exam will be closed to him or her ***(Figure 35***).



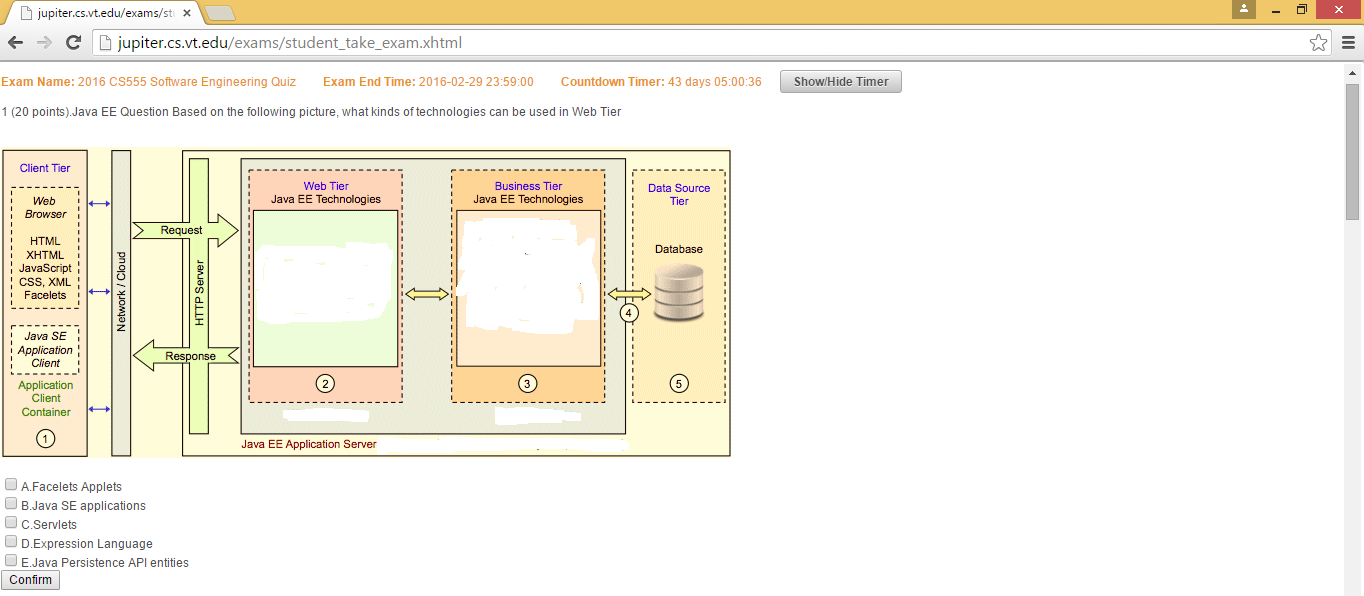


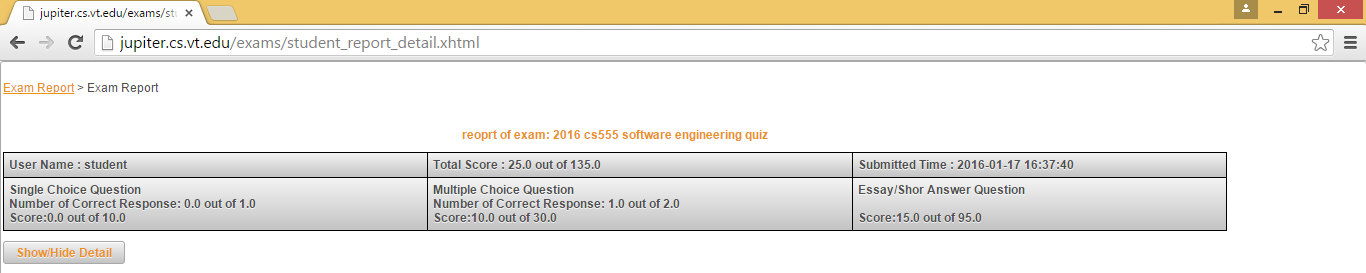


Figure 35. Take Exam Interface

## View Exam Report

After an exam report is published to a student, a student with a valid account can log into the system and view the exam report. The student can view the summary report at the top of the web page; he or she also can click the “Show/Hide Detail” button to see the detail feedback of each question in the report (***Figure*** **36**).





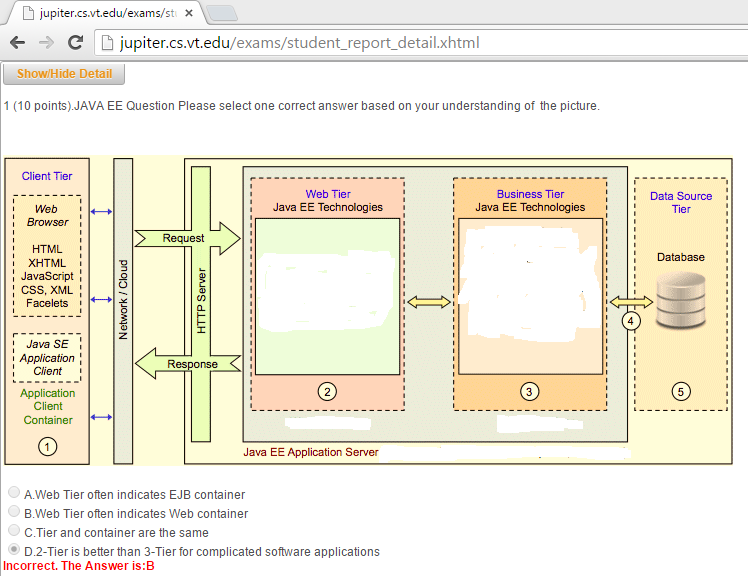


Figure 36. View Exam Report Interface

## Administrator Dashboard

An administrator can log into the system with a valid account and manage all of users, exams and exam reports in the system. The administrator can also configure the system at the dashboard (***Figure 37***).

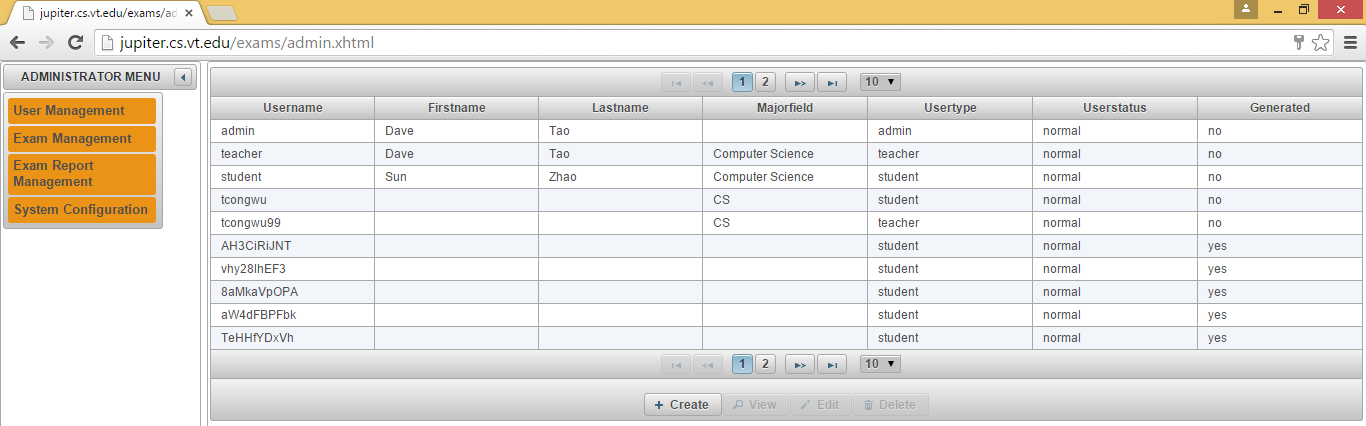


Figure 37. Administrator DashBoard Interface

# Conclusions and Future Development

## Conclusions

This report presents VolunteerCloud, a cloud-based software application for volunteer recruitment. Few successful solutions exist that try to bring volunteers and organizations together using technology, and these may not be equipping organizations with all the appropriate tools needed to improve volunteer attraction and recruitment efforts. In VolunteerCloud, volunteers are able to connect with organizations by searching for opportunities and subscribing to them, managing their volunteering history, and their volunteering interests. Organizations can create and manage their volunteering opportunities, manage their volunteering activity, search for volunteers, and establish communication with volunteers though the use of a webchat feature. VolunteerCloud therefore provides a cloud-based software solution that can serve as a platform to study how volunteer recruitment and attraction rates are affected when organizations are allowed to step out of their shelves, and sell a wider picture of the support they offer to volunteers through the use of real-time online interaction.

## Future Work

Although the current efforts of this project have been focused towards developing a cloud-based software application to serve as a platform where volunteer recruitment and attraction rates could be further studied, VolunteerCloud could be extended to meet the following objectives in the future:

1. Study how volunteer recruitment and attraction rates are affected by providing real-time online interaction.
2. Add a video-chat feature to enhance communications between organizations and volunteers.
3. Improve the graphical user interface.
4. Extend the VolunteerCloud service to be offered in the form of a mobile application as well.

REFERENCES

Balci, O. (2016), “CS5704 Software Engineering Course Website,” Department of Computer Science, Virginia Tech, Blacksburg, VA, <http://manta.cs.vt.edu/cs5704/>

Balci, O (2009), “A Tutorial on How to Create UML Diagrams,” Department of Computer Science, Virginia Tech, Blacksburg, VA, <https://manta.cs.vt.edu/cs5704/Handouts/UML%20Diagrams%20Tutorial.pdf>

Edwards, S. (2017), “CS5744 Software Design and Quality,” Department of Computer Science, Virginia Tech, Blacksburg, VA, <https://canvas.vt.edu/courses/53605>

Hartson, R. (2017), “CS5714 Usability Engineering,” Department of Computer Science, Virginia Tech, Blacksburg, VA, <https://canvas.vt.edu/courses/53605>

Pressman, R. (2014), Software Engineering, A Practitioner's Approach, 8th edition, McGraw- Hill Higher Education, New York, NY

Fairbanks, G. (2010), Just Enough Software Architecture, A Risk Driven-Approach, 1st edition, Marshall & Brainerd

Pfleeger, S. (2010), Software Engineering, Theory and Practice, 4th edition, Pearson, Hoboken, NJ

Hartson, R. (2012), The UX Book: Process and Guidelines for Ensuring a Quality User Experience*, 1st edition*, Morgan Kaufmann, Burlington, MA

Schutt K. and Balci O. (2016), “Cloud Software Development Platforms: A Comparative Overview”, Software Engineering, IEEE

DoDAF (2010a), “DoD Architecture Framework Version 2.02 Volume 1: Introduction, Overview, and Concepts,” Architecture Framework Working Group, Department of Defense, Washington, D.C.

DoDAF (2010b), “DoD Architecture Framework Version 2.02 Volume 2: Architectural Data and Models,” Architecture Framework Working Group, Department of Defense, Washington, D.C.

DoDAF (2010c), “DoD Architecture Framework Version 2.02 Volume III: DoDAF Meta-model Physical Exchange Specification - Developer’s Guide,” Architecture Framework Working Group, Washington, DC.

Chigani, A. & Balci, O. (2011), “Campus Situational Awareness and Emergency Response Management System,” Virginia Tech, Blacksburg, VA, <https://manta.cs.vt.edu/sinergy/index.html>

Microsoft (2009), “Microsoft Application Architecture Guide”, Microsoft Corporation, Redmond, WA <https://msdn.microsoft.com/en-us/library/ff650706.aspx>

Dennis A., Roth R., and Wixom, B. (2012). *Systems Analysis and Design with UML*. John Wiley & Sons, Inc.

Microsoft (2018), “What is SaaS? Software as a Service”, Microsoft Corporation, Redmond, WA, <https://azure.microsoft.com/en-us/overview/what-is-saas/>

Wilson, J. (2012), “Volunteerism Research: A Review Essay”, Nonprofit and Voluntary Sector Quarterly, 41, 176-212

Snyder, M. & Omoto, A. (2008), “Volunteerism: Social Issues, Perspectives and Social Policy Implications”, Social Issues and Policy Review, 2(1), 1-36

Bekkers, R. (2010), “Who Gives What and When? A Scenario Study of Intentions to Give Time and Money”, Social Science Research, 39, 369-381

Cuskelly, G. & Taylor. T & Hoye, R. & Darcy, S. (2016), “Volunteer Management Practices and Volunteer Retention: A Human Resource Management Approach”, Journal of Sport Management, 9, 141-163

Boezeman, E. & Ellemers, N. (2008), “Volunteer Recruitment: The role of organizational support and anticipated respect in non-volunteers’ attraction to charitable volunteer organizations”, Journal of Applied Psychology, 93(5), 1013-1026. Nonprofit and Voluntary Sector Quarterly, 41, 176-212, <http://dx.doi.org/10.1037/0021-9010.93.5.1013>

Yu, K. & Cable, D. (2013), “The Oxford Handbook of Recruitment”, Oxford University Press, Chapter 6, 73-87

Hager, M. & Brudney, J. (2004), “Volunteer Manegement and Retention of Volunteers”, The Urban Institute, Washington, D.C.

Fritz, J. (2017), “What Your Volunteers Need to Be Happy”, The Balance, New York, NY, <https://www.thebalance.com/making-volunteers-happy-2502612>

Mero, J. (2018), “The effects of two-way communication and chat service usage on consumer attitudes in the e-commerce retailing sector”, Electronic Markets, 1-13, <https://www.doi.org/10.1007/s12525-017-0281-2>

VolunteerMatch. (2018), “Volunteer Match Website”, VolunteerMatch Organization, <https://www.volunteermatch.org/>

Samaritan. (2018), “Samaritan Website”, Samaritan Project, <https://www.samaritan.city/>

Bedder. (2013), “Seattle Devs hack app in 48 hours to help homeless find shelter”, Bedder Application, <https://venturebeat.com/2013/10/07/seattle-devs-hack-app-in-48-hours-to-help-homeless-find-shelter/>

Copia (2018), “Copia Website”, Go Copia, PBC, <https://www.gocopia.com/>