April 2015

WPI Student Organization Management System

Alyssa Marie Graham
Worcester Polytechnic Institute

Project Number: MXC-406

WPI Student Organization Management System

A Major Qualifying Project Report

submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the Bachelor of Science degree

by

Alyssa Graham

Date: April 30, 2015

WPI

Professor Michael Ciaraldi, Advisor

Abstract

The goal of the project was to develop a system that would allow small organizations on campus to maintain necessary member and organizational records. The project resulted in an application which fulfilled the initial goal, and could be freely hosted by using WPI's web servers and MySQL servers and easily replicable by WPI organizations. The resulting application would have features that are not existent in freely available applications such as the ability to maintain information about tasks assigned to members of the organization.

Table of Contents

Table of Figures

Chapter 1 Introduction

Managing an organization often requires the organization and archiving of records and information pertaining to the members and events of the organization. This record keeping often is the responsibility of a senior or ranking member of the organization. While it may be a vital function in the organization, often record keeping is time consuming and error prone. This detracts from the total time the responsible party has to devote to other organizational tasks that would be impossible to automate. Automating the process of record keeping for organizations would solve the problem of unnecessary expenditures of time. The goal of this project was to create an application WPI student organizations could use to maintain and review member and organizational records. This project provides an option for organizations to maintain an efficient and low cost system for record keeping.

The record keeping system is designed to be flexible, allowing it to be compatible with a variety of different organizations, but college-based organizations were the inspiration for the project. Larger organizations with a large membership and sufficient funding most likely have their own proprietary and customized systems, so this system did not focus on addressing the needs of such organizations.

1.1 Fulfilled Need

The inspiration for this project came from my experience as Membership Vice President for WPI's chapter of Alpha Phi Omega. My responsibilities were varied, but included maintaining records which included static personal information as well as information that needed to be updated weekly. I found this to be needlessly time consuming and error prone. I believe the reliance on human effort and simple solutions such as maintaining multiple Excel spreadsheets for different records is significantly flawed; this warrants a new system that will improve the process. There are many systems which can keep track of events and event registration, but few will aggregate the information on a per-member basis, and allow members to see their progress toward a specific task or goal. Until these needs are meant, there will exist the need for someone in the organization to maintain records through some other means. The need addressed

will be the need for the maintenance of these records in a system that streamlines the process and helps reduce human error.

1.2 Existing Systems

There are few existing systems that can be utilized to fulfill the needs outlined above without significant drawbacks that make the system infeasible to use. This application eliminates many of these drawbacks. The most basic solution that addresses the same needs solved by this application would be maintaining records using paper and pen. Maintaining physical records typically isn't an ideal solution as it is time consuming and prone to human error. In the particular case of WPI Student run organizations, paper records have the additional drawback of requiring storage space that might not be available to the organization. Additionally because most student-based organizations will experience a complete rotation of membership in about four years; it can be easy for paper records to become lost.

The next solution would be to mimic paper records using a technique such as maintaining Excel spreadsheets. I experienced a system of maintaining several Excel spreadsheets and paper documents coupled with Microsoft's SharePoint tool for the archival and sharing of the Excel spreadsheets. This system was prone to human error, and inefficient. Often the paper sheets would not match with the spreadsheets, and it was easy to add entries to the wrong column or row. Additionally finding any one member's information often required a summation of values through one or more spreadsheets. While the system "worked", it was a chore, and off-putting.

Similar sites such as Facebook.com and orgsync.com are used by organizations to handle event registration, however these do not work well for organizations that need to keep track of statistics such as who has paid dues or what the specific requirements for membership are. Orgsync is similar to Facebook, but does have more organization-geared features, but OrgSync has flaws which make it entirely unfeasible for an organization like WPI's chapter of APO and other similar organizations. OrgSync requires that members of an organization have administrative privileges to create events [1]. Additionally organizations which do not have open membership might want

something that is designed with more internal use in mind as opposed to Facebook or OrgSync which are community oriented.

APOonline.org is an option that is available specifically for chapters of Alpha Phi Omega. This is a management system designed to maintain records which go beyond just events, however APOonline.org comes with subscription based fees, which range from $3 - $4 per member per semester [2]. This expense can be a burden on chapters. For example, a chapter with 50 members would need to pay $300 per year for this service, and this only covers the 50 active members. Additionally during the duration of this project, there was a time that spanned several weeks in which APOonline.org was continuously unavailable. For a subscription based service, this would be incredibly inconvenient, although I am uncertain if chapters were able to access their information during that time.

1.3 Target Audience

The initial target audience of this application was student organizations such as Alpha Phi Omega on WPI's campus. The need was inspired by a WPI organization and it was a natural transition to develop the application with this target audience in mind. Currently WPI has over 200 registered clubs and organizations [3]. This is a potentially large target audience, although many of the clubs and organizations may not maintain the level of records that would necessitate this application. While the application was designed for this target audience the potential for expansion exists. There is a future possibility for expanding the application's audience to other universities as well as any small organization or club that could benefit from the application.

1.4 Expected Features

The expected features of the application were outlined during the project's proposal. Listed below are the features that were proposed. Additionally the use cases created during the initial phase of the project are included at the end of the report in Appendix A.

1. Allow members to register in the application
2. Allow members to sign in and log out of the application

3. Allow for the creation of events
4. Maintain records on event attendance
5. Maintain other pertinent data related to the organization and membership

In order for the project to be successful, these goals needed to be met in a way that allows the application to be easily used, reduces human error, and remains cost effective for organizations to use and implement.

Chapter 2 Background

2.1 Alpha Phi Omega

My experience with WPI's chapter of Alpha Phi Omega served as the main inspiration for this project. Alpha Phi Omega is a co-ed service fraternity which has over 400,000 members on over 375 campuses [4]. Membership in Alpha Phi Omega requires a pledging process, and typically chapters maintain requirements to maintain "active" membership. The details vary from chapter to chapter, but all members are responsible for paying dues to the national office in Independence, Missouri [5]. WPI's chapter maintains records on service hours, events attended, member status, as well as static information such as names and email address. Information such as event dates and times is easily managed through applications such as Facebook or OrgSync. However, data on which members actually attended the event, what requirements the event might have counted toward, and what requirements each member is responsible for is not easily captured in either application. For this reason, this application was developed. As mentioned previously, there is no freely available application that specifically maintains this type of data; this data which is important for the chapter of Alpha Phi Omega at WPI to maintain.

2.2 Available Environment

In keeping with the goals of the project, the development environment needed to support certain functionalities while being cost effective. Purchasing a domain name, and a hosting service, or running private servers is typically the choice larger, funded organizations go with when developing applications and web pages, however this is not feasible for small on-campus organizations.

2.2.1 WPI Server

WPI's web space is free for use by WPI organizations, students, faculty, and staff. Information about the web space can be found on a help wiki maintained by WPI's Helpdesk [4]. The server allows supports Perl, Python, Ruby, Rails, CGI, and several other frameworks, but does not support or allow PHP. PHP is a common language to

utilize when building dynamic websites, but was not an option for this application because it is not allowed on WPI's web servers.

2.3 Utilized Technology

2.3.1 Common Gateway Interface

Common Gateway Interface, or CGI, is the main building block of the application. CGI scripts can be written in different languages including, but not limited to, C/C++, Perl, and Python. For this application Perl was used in conjunction with CGI. The figure below from the O'Reilly book on CGI shows the relay of data between the server and client when a CGI program is run. CGI is executed on the server side instead of the client side, which is necessary for some of the security features of the application. CGI is supported on WPI's web space, which was the motivator for utilizing CGI.

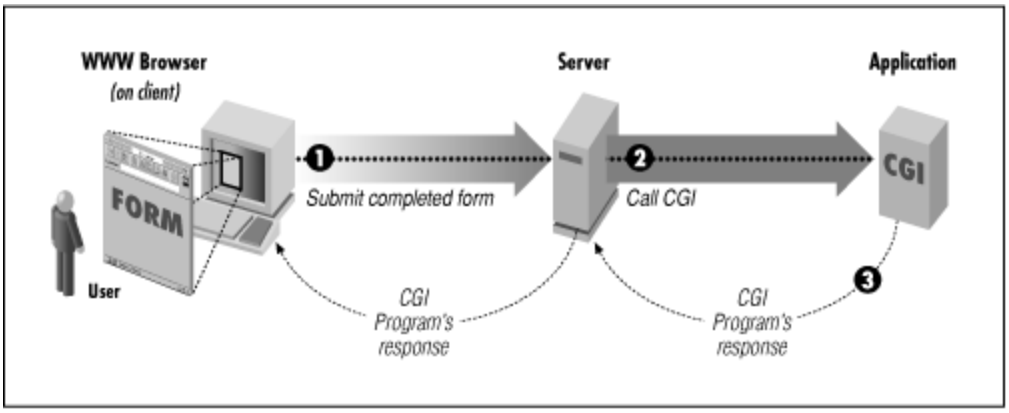

Figure 1: "Simple diagram of CGI" [5]

2.3.2 Perl

Perl is a programming language which has been in development for twenty-seven years. Perl can be used with CGI to create dynamic web applications [6]. Perl is similar to many other programming languages in that it supports importing modules, which aids in development. Modules can be created to modularize code, and apply the same methods across an application. At the time of writing this report, I wrote a CGI/Perl script, which displays, on a web page, a list of all the Perl modules currently installed on WPI's web servers. The resulting list will be available at the end of the report as Appendix B. The modules used by the application will be in a bold typeface. One of the

benefits of using Perl is the relative maturity and popularity of the language, which comes with no shortage of documentation.

2.3.3 SQL

SQL or Structured Query Language is a programming language used for managing relational database. SQL was used when manipulating the MySQL database in the application.

2.3.4 HTML/CSS and HTML Templates

The web pages are written as .html and .tmpl files. The two file types both serve different purposes, and are handled differently on the server side of the application. The only .html files that exist are the outward facing public pages of the application. There was no efficient way to obscure access to .html files as any scripts executed from the .html page would only execute after the file was given to the requester. The .tmpl or template html files solved this issue. Their access can be set to 'read-only by the owner' since the actual file is not what is sent to the requester. Instead the file is read by a CGI script, which interprets and prints the html code when the CGI is executed. This allows the script to run other user verification scripts before any information is sent to the requester. Additionally because CGI is executed by the server, there is no way to access the CGI file itself, only the results of running the CGI file. An additional benefit to using template html instead of regular html is that the template files allow for the addition of tags which can be populated with information by using variables during the execution of the CGI. The template files can also have "if" tags which only output the contents between the tags if during execution the CGI script passes the variable set in the "if" tag with a value of "true". This is utilized in the application to only display certain links when the user should have access to those pages. (The pages themselves also perform authentication, so this is for convenience and aesthetics).

2.3.5 FullCalendar

FullCalendar is a jQuery plugin calendar available under an MIT license. You can view the license information at the FullCalendar website at fullcalendar.io/license. The FullCalendar source code uses PHP to populate the calendar with events from a JSON

file or array. This allows the plugin to get event data from a database. However, because PHP is not allowed on WPI's web servers, it was necessary to modify the existing FullCalendar code to work with CGI/Perl.

2.3.6 Apache HTTP Web Server

WPI's web server runs using the version 2.2.3 Apache HTTP server software. Directions for using WPI's web space can be found at http://wiki.wpi.edu/ITS/WPI_Web_Space. Users of the web space must follow WPI's acceptable use policy. Additionally the web server's configuration is not changeable, so web applications have to be built to work with WPI's configuration.

2.3.7 Sendmail

Sendmail is utilized for sending email through the application. Currently email is not tightly integrated in the application, however, email is utilized during the registration process. The application is set to email an administrator when there is a new registration in the system. I decided that it was necessary to include this so that registrations would be processed in a timely manner. More automatic emails might improve the application, but I do not believe they are necessary for the application to have a full level of functionality.

2.4 Security Precautions

2.4.1 Password Encryption

Password encryption is used to obscure user passwords. There are several options for password encryption built into MySQL databases. For this project I selected SHA1 encryption as other forms of encryption were not possible or easily implemented with the configuration of the MySQL database. SHA1 or the Secure Hash Algorithm was developed by the National Institute of Standards and Technology and the National Security Agency [9]. There are more secure options available such as SHA3, but for the purposes of this application, I feel SHA1 is an acceptable additional step to secure password data.

2.4.2 HTTPS

HTTPS is the HTTP layered on the SSL protocol. The SSL protocol or Secure Sockets Layer was designed to add an extra layer of security to web browsing. A secure connection is formed between the client and server and data that is transmitted is encrypted [10]. This application utilizes HTTPS across the whole application as an additional security measure.

2.4.3 SQL Injection

SQL injection is a commonly exploited vulnerability that can exist in web application which do not take any preventative measures against it. There are several different ways an SQL injection can successfully occur, but the vulnerability exists at any point where user input or input that can be manipulated by the client are used in SQL methods. In this application I took measures against SQL injection, which are outlined in Section 3.2.2.

Chapter 3 Methodology

3.1 User Interface Design

The application's user interface was designed to follow the common pattern of web applications. The application has two main sections which appear on the main, non-admin pages. These sections are navigation and content. The navigation is kept constant across the application. All links are kept to the left side while the content is to the right. The navigation buttons were designed with a slight rounded corner, which sets them apart from the squared content box. Additionally the button's color changes as it is hovered over, which further reinforces the idea that they are links. Most of the aesthetics of the application was accomplished by utilizing a CSS stylesheet. One stylesheet is used across the application with the exception of the calendar page. To be compatible with the already existing FullCalendar code, a second stylesheet was added with some small edits. Lastly, there is some inline styling on some of the administrative pages which are simpler in design. I wanted these pages to have a different appearance to further emphasize their separate importance from the main application.

3.2 Security

For the application, I tried to consider security as I implemented the components of the application. I do not intend for the application to be used for sensitive information or any financial information. There are no components to the application that would lend itself to being utilized in such a manner, however, I do understand that future users may not use the application for its intended purpose, so precautions are built into the application. This application is not intended to be an outward facing website for organizations, so I mainly focused on protecting against common exploitations and security concerns.

3.2.1 Password Encryption

Password encryption for user's passwords uses SHA1 encryption. This encryption method is available for mySQL databases. As I mentioned in the background section, SHA1 encryption is no longer the best method available for password encryption, however due to the other precautions taken to secure the transmission of data combined

with the low level of sensitive information maintained by any instance of this application, it should be sufficient.

3.2.2 Protecting against SQL injection

In order to protect against SQL injection, I parameterized all SQL methods that use user input. At points in the application, such as user registration, there are forms in which information input by the user is used in SQL methods. This could cause an issue because it allows users to manipulate what text they input in forms to mimic SQL commands, which are then unintentionally executed against the MySQL database. The way parameterization in the CGI script protects against SQL injection is by designating variables using a '?' in SQL code that is to be executed. Then the form data is validated and passed as a variable when the SQL is about to be executed. Because of this separation, the variables are never executed as code.

3.2.3 User Authentication

User authentication and authorization is handled by a Perl module which I wrote to be part of the application. My method of user authentication utilizes the database with a randomly generated session key to protect against cookie editing. Cookies can easily be edited by end users, and cannot be solely relied on for user authentication, but requiring users to enter their username and password in for every page they want to view is not a feasible method of user authentication. This is why the session key is necessary. Every session key is a 40 to 70 character length randomly generated series of alphanumeric characters. This key is saved as a cookie and saved to the MySQL database. Each session key is unique for every login session and user, and each session key is cleared from the database when the user logs out of the application. Additionally the user's unique ID is saved as a cookie when they login. This allows each page to check in the database that the unique ID and session key match. This would make it difficult for the combination to be guessed in attempt to gain malicious entry into the application.

3.2.5 Obscuring Database Credentials

Database credentials were obscured to any end user. This was accomplished by maintaining a 'read-only by owner' txt file with the database credentials. Each CGI script which requires database credentials reads from the file and can then access the credentials for connecting to the MySQL database, however this information is not accessible to the user as the CGI scripts do not ever print that information as part of the resulting html that ultimately gets displayed to the end user.

3.2.6 Transmitting Data Securely

HTTPS is forced across the entire application. This was accomplished by added one line to the application's .htaccess file. This line causes any request to a page in the application to use HTTPS. This is handled by the server and occurs before the page is sent to the client.

3.3 Database Design

The MySQL database utilized by the application was designed so that the information would be maintained in several different tables, which help support the

ability to quickly and efficiently search for the data contained in the database. See

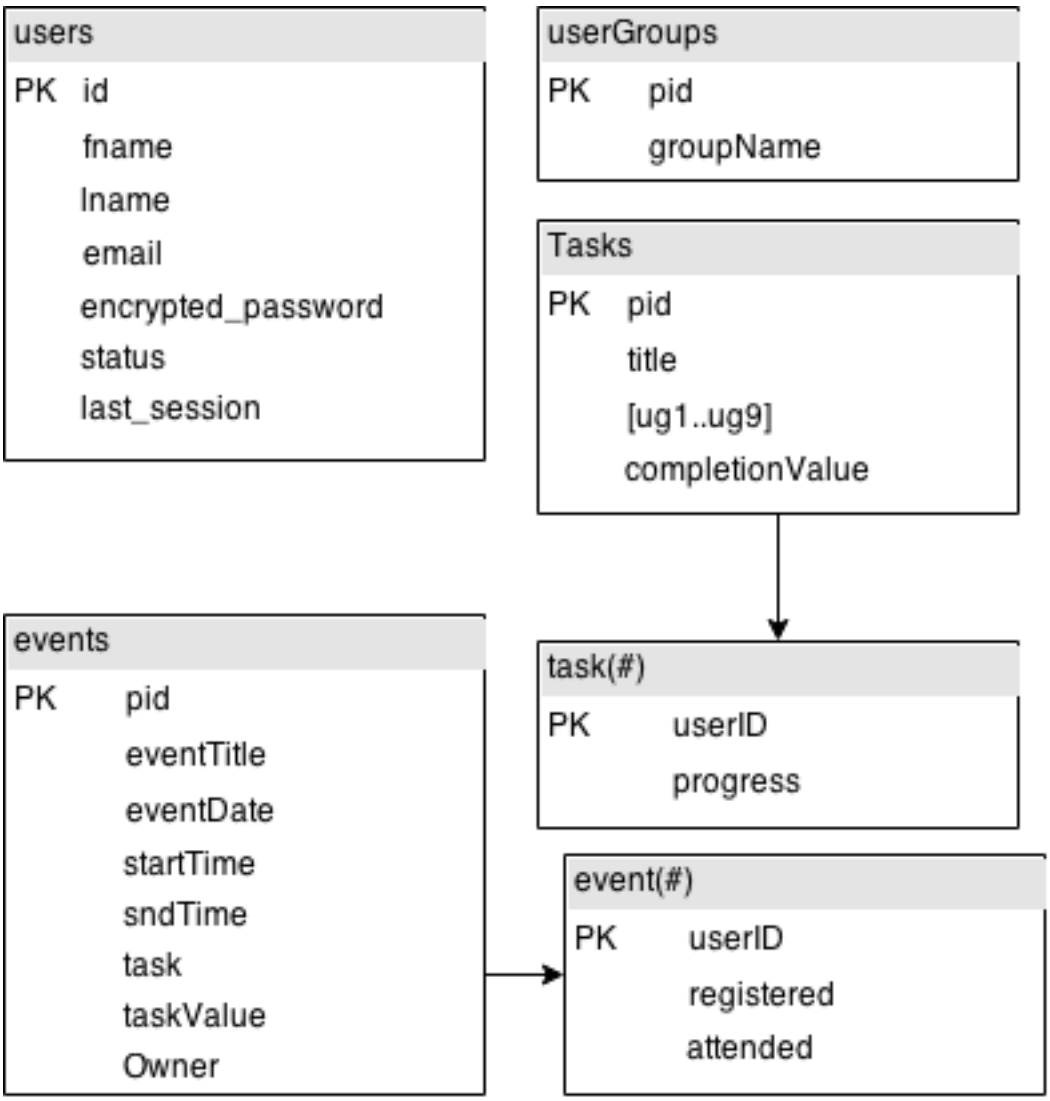

Figure 2 located at the end of the report to view the database diagram.

3.3.1 User Information Table

Static user information is maintained in the "users" table. This table maintains a user's unique ID, first name, last name, email, status, password, and session ID. The unique ID is the table's primary key, and is automatically generated when a user is added to the table.

3.3.2 Event Information Table

The "events" table maintains information on every event currently in the application. The information maintained is the "pid" which is the primary key for the table, the "eventTitle", "eventDate", "startTime", "endTime", "location", "task", and "taskValue". The "task" column contains the primary key of a task if the event has one associated with it. The "taskValue" column contains the value users will receive toward that task if they attend the event.

3.3.3 Task Information Table

The "tasks" table contains information relevant to all tasks in the application. Each task has a "pid" which is the primary key, a "title", "completeValue", and a collection of Boolean values representing which user group(s) are assigned the task.

3.3.4 User Group Table

The "userGroups" table has a "pid" and "groupName" column. The table only keeps track of how many userGroups exist in the table and maintains the "pid" or primary key for unique reference to each user group.

3.3.5 Event Attendance Table

There is an "event#" table for each event. The table's name would be the word "event" followed by the "pid" of the event the table belongs to. These tables serve to keep track of which users are registered for an event and which users have attended an event.

3.3.6 Task Assignment Table

There is a "task#" table for each task. Similarly to the tables that exist for each event, this table maintains information on what progress each user had made toward the task.

Chapter 4 Results

The final results yielded an application which contains basic features and functionalities to allow organization on campus to maintain member, event, and task records. The figures section at the end of this report includes screenshots of the application, its various functionalities, and the error screens that exist in the system.

4.1 Application Capabilities

The application at the time of this report has the ability to create a new instance of the application, create and maintain user profiles, approve or deny user registration requests, create events, take attendance for events, display a calendar of events, display a list of events, change user statuses, register/unregister for events, and delete events. The application displays error messages to the user when requested actions cannot be performed (e.g. when a user attempts to register with a password that doesn't meet the six character minimum password requirement). See Figure 6 to view this example.

Chapter 5 Conclusion

5.1 Measure of Success

I consider the conclusion of the project to be successful. The initial goals of the project were met. The application is at a state where it can be improved, tested, or utilized for other similar projects.

5.1.1 Cost Effectiveness

Currently the application does not cost anything if a WPI organization would like to create their own instance of the application. WPI offers free web hosting services as well as free MySQL database hosting [7]. It was important that the application would be cost effective for organizations as often organizations do not receive any or much funding. In the example of Alpha Phi Omega, all funds come directly from members, or from fundraising.

5.1.2 Utilization of WPI Resources

Related to cost effectiveness, allowing the application to require only the resources that WPI supplies freely was a goal of the project. The application was majorly successful in only requiring WPI resources. The parts that were not supplied such as full calendar and the HTML::Template module are included in the code package, so that users do not need to configure anything aside from their WPI supplied MySQL database and public web space.

5.1.3 Application Capabilities

The current capabilities of the application, I believe are successful in fulfilling the initial project goal. The application is able to maintain, display, and create information related to events, tasks, and members. Every member is able to check their status on their own profile, which will allow them to immediately determine their progress without needing to shift through Excel spreadsheets or wait for a report to be emailed/made available to them.

5.1.4 Adaptability of Project

There is a potential for components of the project to be utilized for other projects on campus. The user authentication module can be used for projects with a similar level of need for user authentication. Additionally my project can serve as an example of a CGI/Perl based project which works according to WPI specifications. One of the most difficult parts of the project was attempting to determine what would work, and what was available to be utilized.

5.1.5 Issues and Concerns

The HTML::Template module shows that it should be accessible according to the script I wrote to print out a list of the installed Perl modules, however it does not work when a CGI script attempts to use the HTML::Template. This forced me to install the module under the directory containing the application. This is not ideal as I wanted to avoid situations that could cause user difficulty during the initialization of the application.

5.2 Future Work

5.2.1 Possible Additional Features

There are many possible features that can be added to the application to improve usability or make the application more appealing to organizations. As the project progressed I maintained a list of features that could be added given more time to work on the project. Potential additional features include, but are not limited to, automatic email rules, text alerts, customizable color schemes, event templates, multiple ownership of events, information pages, and support for individual tasks.

5.2.2 Future Target Audiences

As mentioned in section 1.3, the target audience can be expanded in further iterations of this project. The current target audience is limited, and serves as a good point of testing as WPI student organizations are very reachable for anyone who would continue working on the project on campus. While the application works for WPI organizations, expanding the target audience could be used to eventually make this

project a commercial project, or simply to improve the organization of other groups by providing a free, open source software.

5.2.3 User Testing and Case Studies

User testing and case studies with the application should be part of the future work for this project. Unfortunately I was unable to get to user testing with this application as developing the initial version required the complete year's effort. It would be beneficial for the next group or individual working on the project to focus a significant portion of the efforts on bring the application to a WPI student organization and working with them to test the application and add features that they request be added to the application. User testing is an important part of development projects, but unfortunately it was not feasible for this project.

5.2.4 Revision of Code

Lastly any continuation of this project should also attempt to revise the current code to increase efficiency and security. The code is documented in the application, but because the time constraints faced and the limited amount of work I could accomplish as one person, there is plenty of room for the code to be made more modular and less repetitive.

Figures

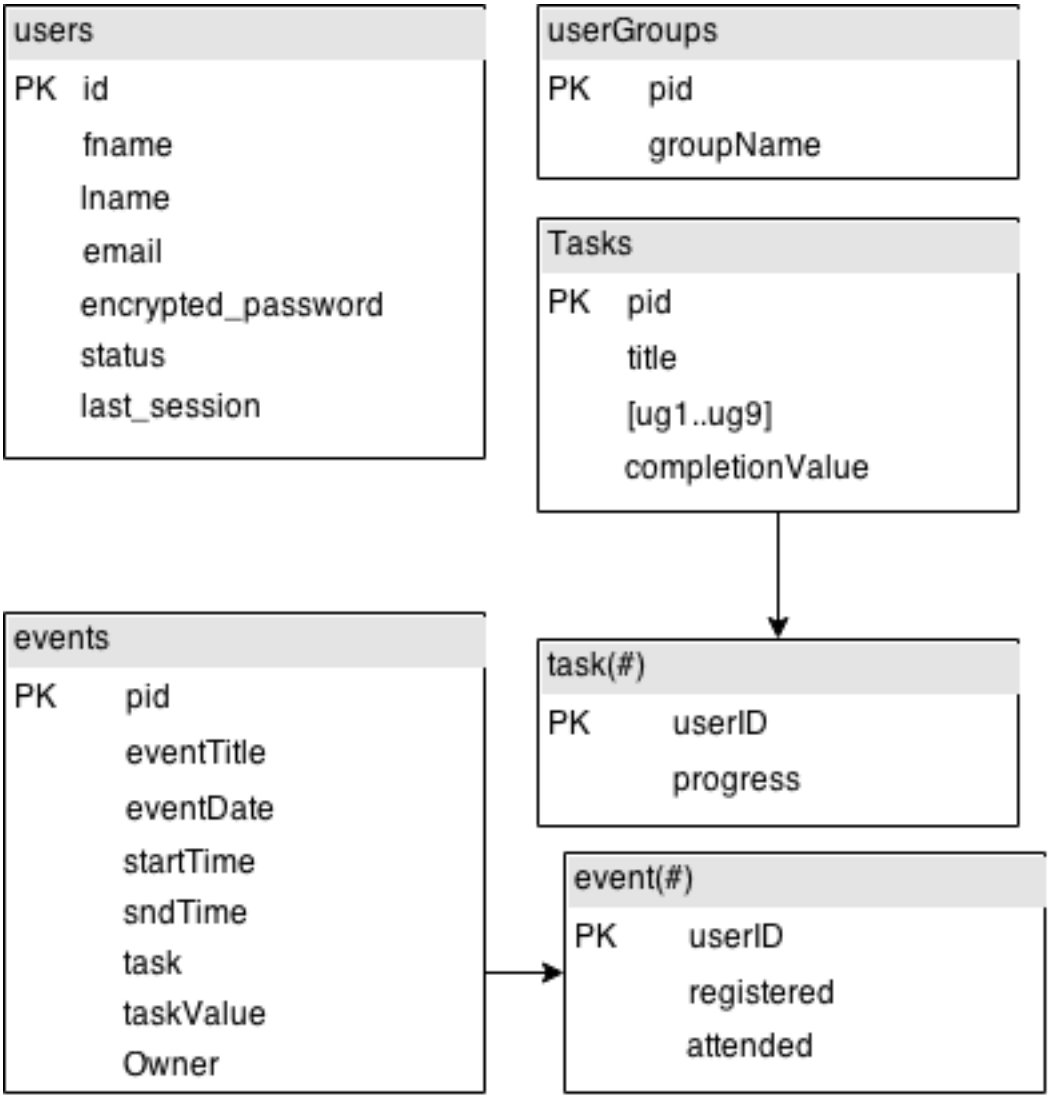

Figure 2: Database Schema for the application.

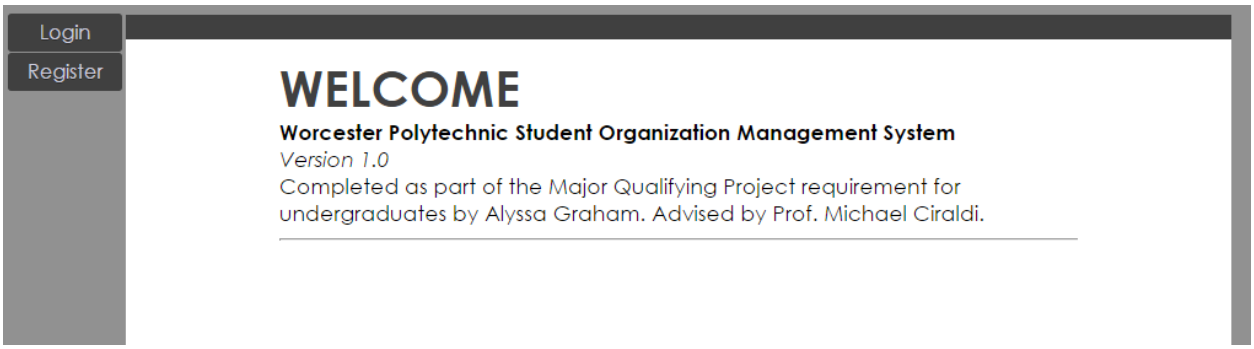

Figure 3: Public page for application. Links to the login and registration screens.

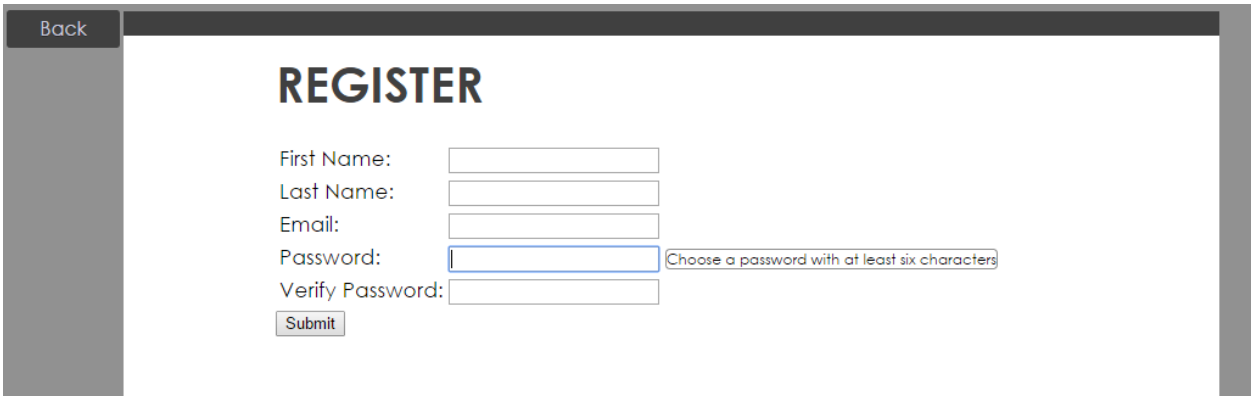

Figure 4: Registration screen with visible password tool tip.

There were empty fields in your registration.

Please go back and make sure you enter values for all the fields on the registration form.

Figure 5: Registration error - empty fields in form.

Your requested password does not meet minimum requirements.

Please go back and make sure you're following the password minimum requirements.

Figure 6: Registration error - password does not meet minimum requirements.

Your passwords did not match

Your passwords did not match. Please go back and make sure you're entering the same password in both password fields.

Figure 7: Registration error - password fields did not match.

Your email is already in the system

If you've already registered you may be pending approval. Talk to your admin if your account is not approved in a timely manner.

Figure 8: Registration error - already registered in the system.

Figure 9: Login screen.

You're currently not an approved user

You've already registered and are currently pending approval. Talk to your admin if your account is not approved in a timely manner.

Figure 10: Login error - not approved user.

Your password and/or email combination was incorrect

Please go back and try again.

Figure 11: Login error - incorrect password/email combination.

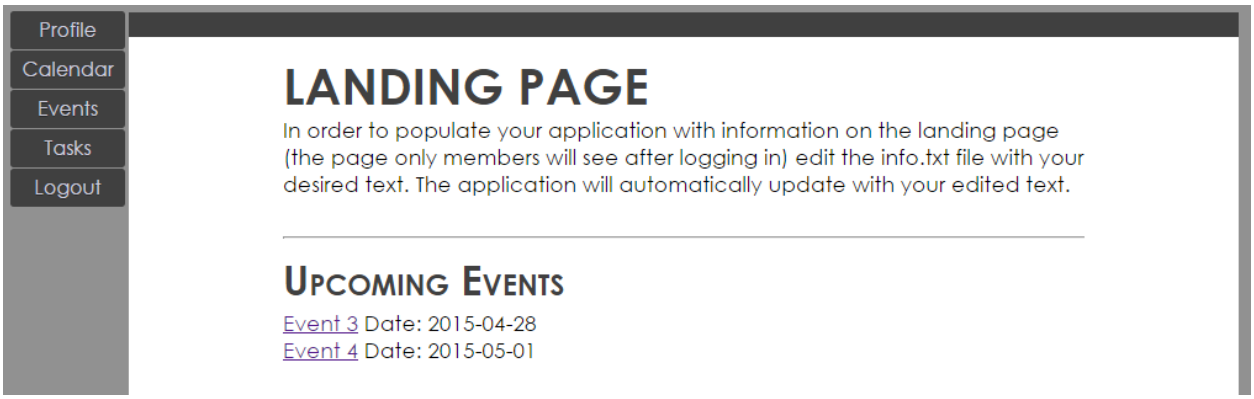

Figure 12: Landing Page. Displays customizable message and upcoming events.

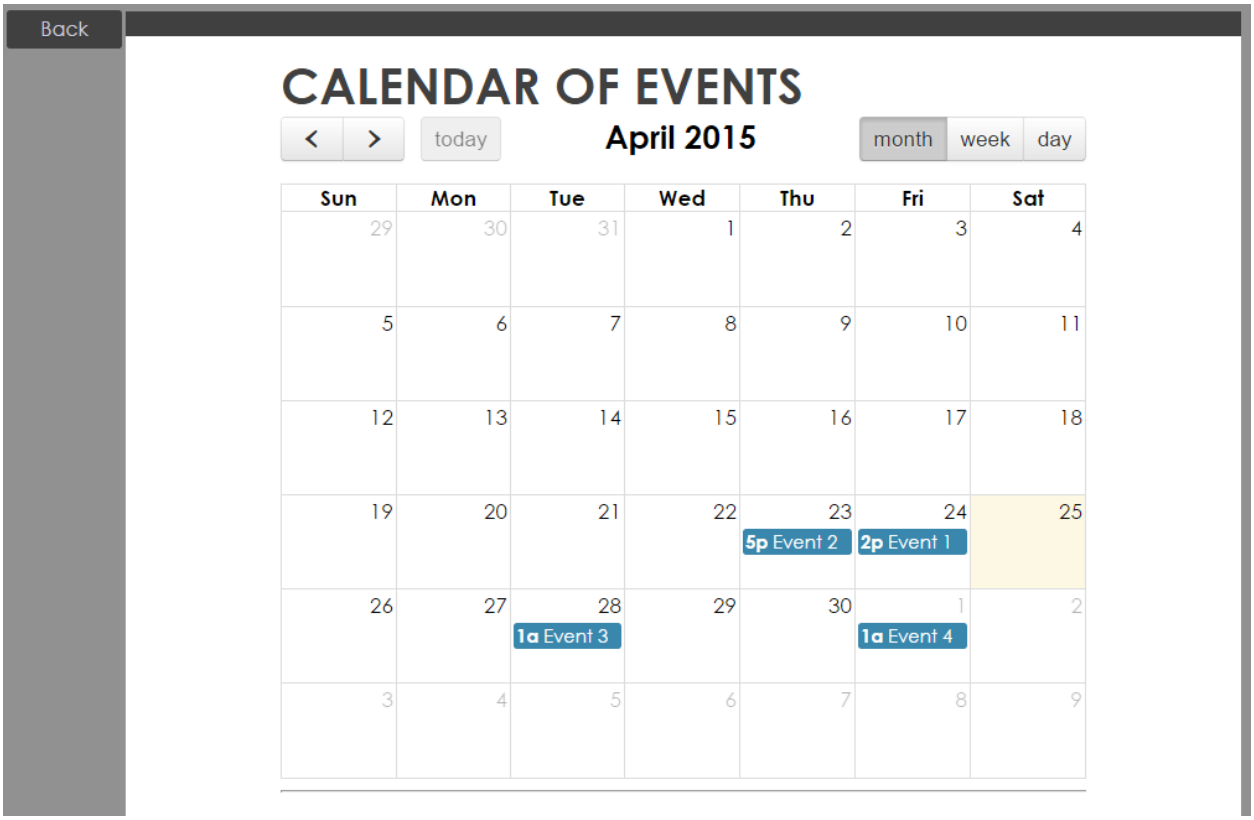

Figure 13: Calendar Page. Displays calendar populated with system events. Available in month, week, and day views.

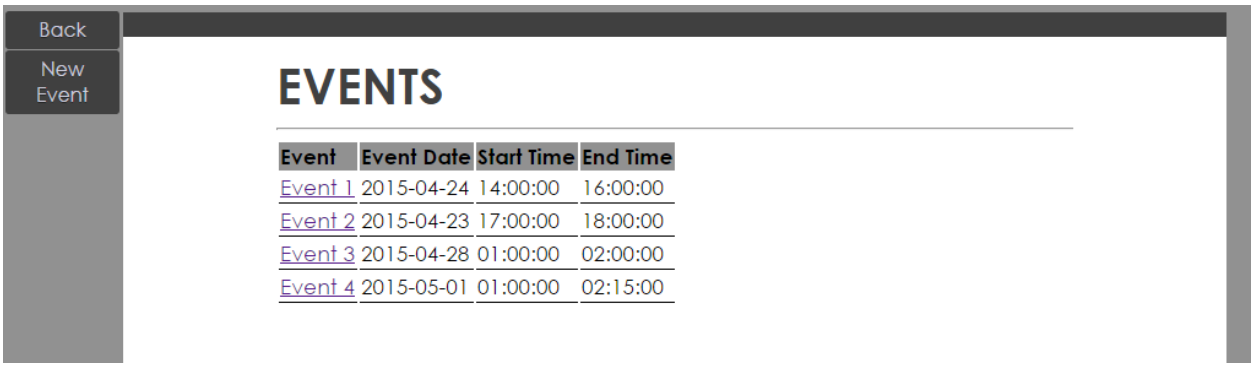

Figure 14: Events Page. Displays list view of events in system.

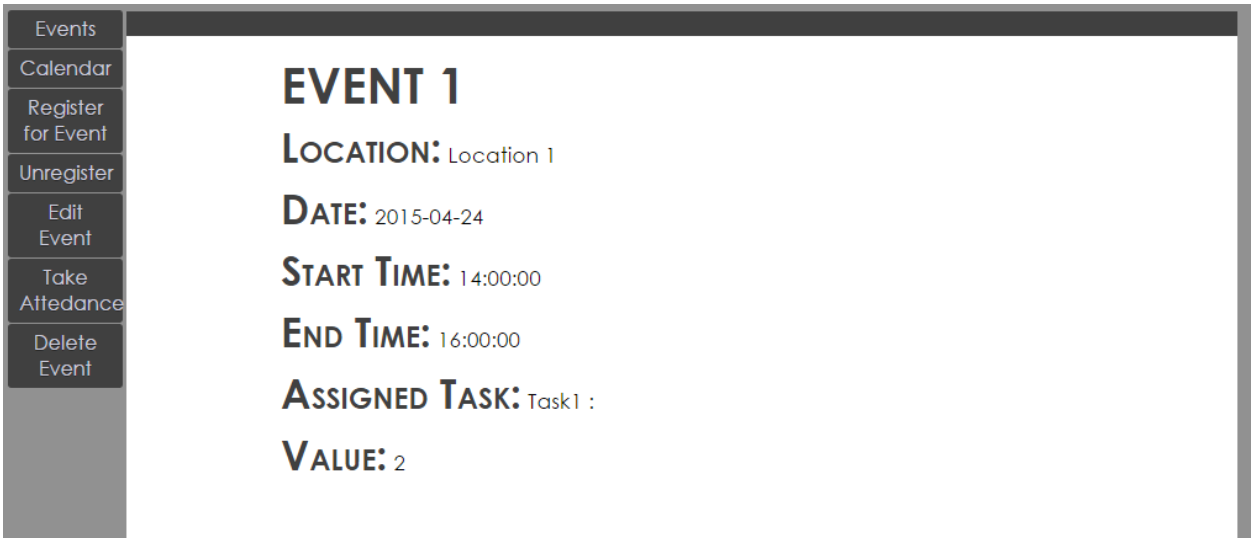

Figure 15: Individual Event Page in Admin/Owner View. Displays event's information.

Figure 16: Registered for Event Confirmation.

Figure 17: Unregistered for Event Confirmation.

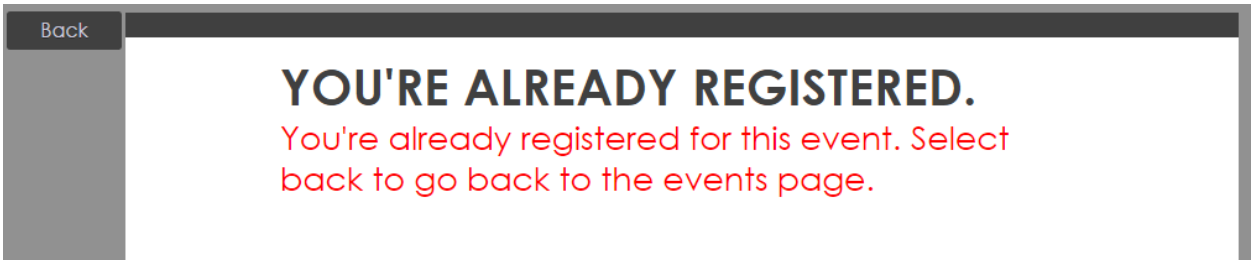

Figure 18: Already Registered Error.

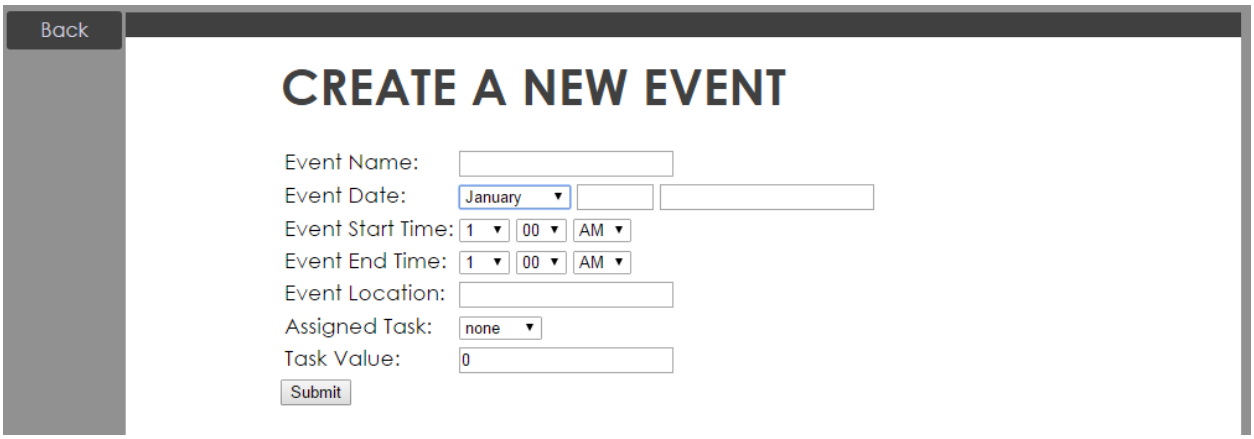

Figure 19: Create New Event Page.

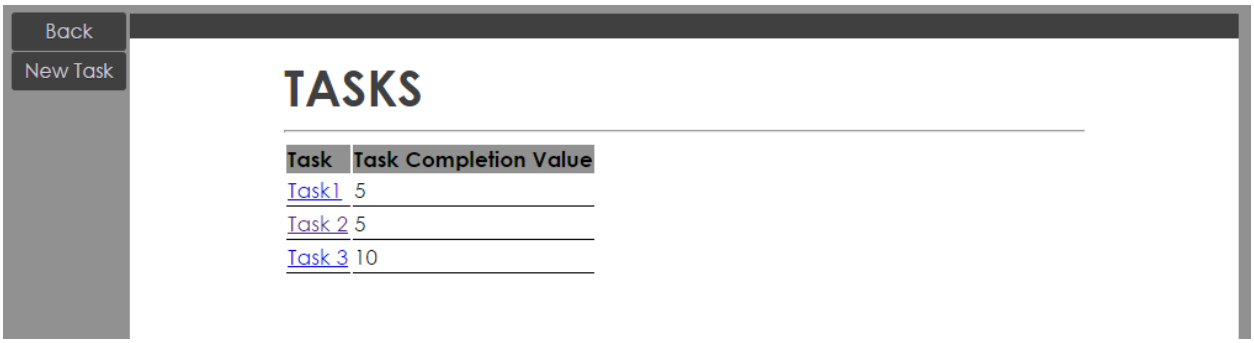

Figure 20: Tasks Page.

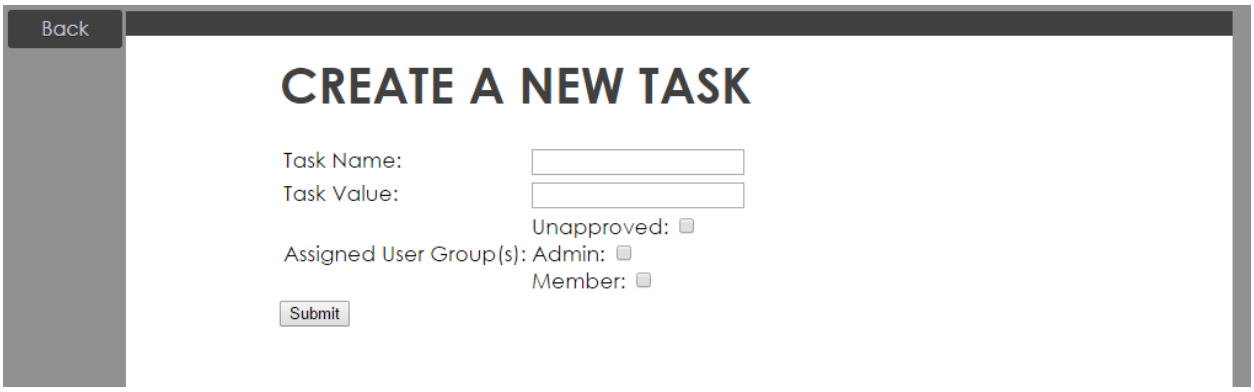

Figure 21: Create a New Task Page.

References

[1] OrgSync, "Create an Event," OrgSync, [Online]. Available: https://help.orgsync.com/hc/en-us/articles/203702346-Create-an-Event. [Accessed 25 04 2015].

[2] APOonline, "Pricing," APOonline, [Online]. Available: http://www.apoonline.org/pricing.php. [Accessed 25 04 2015].

[3] Worcester Polytechnic Institute, "WPI Student Organizations," [Online]. Available: http://wpi.orgsync.com/Student_Orgs.

[4] Alpha Phi Omega, "Alpha Phi Omega - Who We Are," 2012. [Online]. Available: http://www.apo.org/aboutus. [Accessed 29 4 2015].

[5] Alpha Phi Omega, "Alpha Phi Omega - History & Tour," 2012. [Online]. Available: http://www.apo.org/aboutus/ourorganization/nationaloffice. [Accessed 29 4 2015].

[6] WPI Helpdesk, "WPI Web Space," 8 10 2014. [Online]. Available: http://wiki.wpi.edu/ITS/WPI_Web_Space.

[7] S. Gundavaram, "O'Reilly Open Books Project," 3 1996. [Online]. Available: http://www.oreilly.com/openbook/cgi/ch01_01.html.

[8] Perl.org, "About Perl," 2015. [Online]. Available: https://www.perl.org/about.html.

[9] P. A. DesAutels, "SHA1 Secure Hash Algorithm - Version 1.0," W3C, 1 10 1997. [Online]. Available: http://www.w3.org/PICS/DSig/SHA1_1_0.html. [Accessed 29 4 2015].

[10] A. O. Freier, P. Karlton and P. C. Kocher, "The SSL Protocol," Transport Layer Security Working Group, 18 11 1996. [Online]. Available: https://tools.ietf.org/html/draft-ietf-tls-ssl-version3-00. [Accessed 29 4 2015].

[11] Worcester Polytechnic Institute, "My SQL," 05 03 2015. [Online]. Available: http://www.wpi.edu/academics/CCC/Services/Databases/MySQL.html.

Appendix A

Use Case 1: Registering as a new user
Primary Actor: All Members
Priority: Vital
Preconditions
1. The user access the landing page of the application

Success Condition(s)
1. The user reaches the registration form
2. The user fills the form with a valid name, email, and password
3. The user submits form via the submit button
3a. The user's email is verified as unique
3b. The user sees an error stating that their email is not unique and to either go back and correct or speaks to an executive
4. The user lands on a success page with a message that they will be able to access the system once an administrator approves their registration
5. The information from the registration form is added as an entry in the member database table with the following information appended: date of registration, unique numerical user ID
6. All administrators or the designated administrator gets a system alert that there are user(s) registrations pending approval on their landing page upon logging into the system
6a. (optional) The adminstrator(s) receive an email notifying that a user has registered. This email can be once per day as necessary.

Trigger: User selects "register" link on the landing page

Extensions & Error Handling
1. Registration form unavailable: 404 error - contact administrator
2. Form missing values: Submission button not clickable until all sections of the form have appropriate entries
3. Invalid form entry i.e. password too short or not valid email: Submission button not clickable alert appears explaining error
4. Email not unique: After submission user is taken to an error page explaining that they may have already registered. They are given the option to go back and edit the email or talk to an administrator.

Required Database Entries
User Information Table in Database {Registration Time stamp, uniquely generated user id, first name, last name, unique email, password, Member Group, Approval Status}

Additional Information
Successor Use Case: Approving user registrations

Use Case 2: Logging into the application
Primary Actor: All Members
Priority: Vital
Preconditions
1. User has registered
2. User's registration has been approved by an administrator

Success Condition(s)
1. User reaching application landing page
2. User fills in email and password in login form
3. User selects submit
4. User successfully gets to their personalized landing page after logging in
5. User gets to an error page explaining what went wrong

Trigger: User completes login form on landing page and selects submit
Extensions & Error Handling
1. User not found in system: Display error page explaining a user with that email was not found in the system. Please either register or try logging in again, making sure to enter your email carefully.
2. Password incorrect: Display error page explaining the password did not match the password in the system. You may try again (link back) or you may click here if you forgot your password.

Required Database Entries
User Information Table in Database {Registration Time stamp, uniquely generated user id, first name, last name, unique email, password, Member Group, Approval Status}

Additional Information

Use Case 3: Viewing Personal Profile
Primary Actor: All Members
Priority: High
Preconditions
1. User has registered
2. User's registration has been approved by an adminstrator
3. User has logged on

Success Condition(s)
1. The user reached their profile page
1a. The profile page contains all accurate and current information for the user
1b. The profile page displayed is for the user logged in and no other user

Trigger: User selects "profile" link from any page
Extensions & Error Handling
1. Profile page unavailable: 404 error - contact Administrator
2. Profile page does not belong to user: 550 error - Your account does not permission to access this page.

Required Database Entries
Tasks Table in Database {Uniquely Generated ID, Task Title, Assigned Member Group, Completion Value}
Member Task List in Database {User ID, Task ID, Task Progress}
User Information Table in Database {Registration Time stamp, uniquely generated user id, first name, last name, unique email, password, Member Group, Approval Status}

Additional Information

Use Case 4: Editing Personal Information
Primary Actor: All Members
Priority: High
Preconditions
1. User has registered
2. User's registration has been approved by an administrator
3. User is logged into the system

Success Condition(s)
1. User reached edit information page
2. Edit information page functions properly

Trigger: User selects "edit information" on profile
Extensions & Error Handling
1. Edit page unavailable: 404 error - contact Administrator

Required Database Entries
User Information Table in Database {Registration Time stamp, uniquely generated user id, first name, last name, unique email, password, Member Group, Approval Status}

Additional Information

Use Case 5: Creating an Event
Primary Actor: All Members
Priority: High
Preconditions
1. User has registered
2. User's registration has been approved by an administrator
3. User is logged into the system

Success Condition(s)
1. User successfully reaches "create event" page
2. User is succesfully directed to appropriately fill in event form

3. User submits form
4. User receives success message for form submission
5. Information from form is entered into the event table in the database
5. Calendar of events is updated to display the newly created event
6. The event is added to the user's list of events and displayed on their profile

Trigger: User selects "create event" link from navigation bar
Extensions & Error Handling
1. Event Creation Form unavailable: 404 error - contact administrator
2. Form missing values: Submission button not clickable until all sections of the form have appropriate entries
3. Invalid form entry i.e. start and/or end times not in appropriate Time format: After submission user is taken to an error page explaining the error. They are given the option to go back and edit the form or email the administrator for questions.

Required Database Entries
Event Information Table in Database {Uniquely Generated ID, Title, Event Date, Start Time, End Time, Location, Attached Task}

Additional Information

Use Case 6: Creating a Task
Primary Actor: Administrative Members
Priority: High
Preconditions
1. User has registered
2. User's registration has been approved by an administrator
3. User is logged into the system
4. User has administrative privileges

Success Condition(s)
1. User successfully reaches "create task" page
2. User successfully directed to appropriately fill in task form
3. User submits form
4. User receives success message for form submission
5. Information from form is entered into the task table in the database
6. Task ID is added to the appropriate members' task list

Trigger: User selects "create task" link from navigation bar
Extensions & Error Handling
1. Task Creation Form unavailable: 404 error - contact administrator
2. Form missing values: Submission button not clickable until all sections of form have appropriate entries
3. Invalid form entry i.e. non-numerical entry for task completion value: After submission user is

taken to an error page explaining the error. They are given the option to go back and edit the form or email the administrator for questions.
4. User is not an administrator: 550 error - Your account does not have permission to access this page.

Required Database Entries
Tasks Table in Database {Uniquely Generated ID, Task Title, Assigned Member Group, Completion Value}
Member Task List in Database {User ID, Task ID, Task Progress}
User Information Table in Database {Registration Time stamp, uniquely generated user id, first name, last name, unique email, password, Member Group, Approval Status}

Additional Information

Use Case 7: Editing an Event
Primary Actor: All Members
Priority: High
Preconditions
1. User has registered
2. User's registration has been approved by an administrator
3. User is logged into the system

Success Condition(s)
1. User reaches "edit event" page for selected event
2. User is able to make edits to event title, event date, start time, end time, location, and/or attached task
3. User submits editted form
4. Updated form entries are changed in the database for that event's table entry.

Trigger: User selects "edit event" link next to one of their events
Extensions & Error Handling
1. Event edit page unavailable: 404 error - contact administrator
2. Form missing values: Submission button not clickable until all sections of the form have appropriate entries
3. Invalid form entry: After submission user is taken to an error page explaining the error. They are given the option to go back and edit the form or email the administrator for questions.
4. Event does not belong to user: 550 error - Your account does not have permission to access this page.

Required Database Entries
Event Information Table in Database {Uniquely Generated ID, Title, Event Date, Start Time, End Time, Location, Attached Task}

Additional Information

Use Case 8: Editing a Task
Primary Actor: Administrative Members
Priority: Medium
Preconditions
1. User has registered
2. User's registration has been approved by an administrator
3. User is logged into the system
4. User has administrative privileges

Success Condition(s)
1. User reaches "edit task" page for selected task
2. User is able to make edits to task title, assigned member group, and/or completion value
3. User submits editted form
4. Updated form entries are changed in the database for that task's table entry

Trigger: Administrative user selects to edit a task from task list page
Extensions & Error Handling
1. Task edit page unavailable: 404 error - contact administrator
2. Form missing values: Submission button not clickable until all sections of the form have appropriate entries
3. Invalid form entry: After submission user is taken to an error page explaining the error. They are given the option to go back and edit the form or email the administrator for questions.
4. User is not an administrator: 550 error - Your account does not have permission to access this page.

Required Database Entries
Tasks Table in Database {Uniquely Generated ID, Task Title, Assigned Member Group, Completion Value}

Additional Information

Use Case 9: Creating an Information Pages
Primary Actor: Administrative Member
Priority: Low
Preconditions
1. User has registered
2. User's registration has been approved by an administrator
3. User is logged into the system
4. User has administrative privileges

Success Condition(s)
1. User successfully reaches "create information page" page
2. User fills desired information into form
3. User submits form

4. An html information page is generated and saved to server
5. A link is added to the system homepage for the information page

Trigger: Administrative user selects "create information page" link
Extensions & Error Handling
1. Create information page unavailable: 404 error - contact administrator
2. Form missing values: Submission button not clickable until all sections of the form have entries.
3. User does not have administrative privileges: 550 error - Your account does not have permission to access this page.

Required Database Entries

Additional InformationReason for priority: Information pages can easily be hosted elsewhere. It however, would be nice to have a place to post information that is available only to those affliated with the organization.

Use Case 10: Viewing the Calendar of Events
Primary Actor: All Members
Priority: High
Preconditions
1. User has registered
2. User's registration has been approved by an administrator
3. User is logged into the system

Success Condition(s)
1. User reaches "calendar of events" page
2. The events entered into the database are properly displayed on the calendar

Trigger: User selects "calendar" link from main page
Extensions & Error Handling
1. Calendar of events unavailable: 404 error - contact administrator

Required Database Entries
Event Information Table in Database {Uniquely Generated ID, Title, Event Date, Start Time, End Time, Location, Attached Task}

Additional Information

Use Case 11: Printing the Attendance List for an Event
Primary Actor: All Members
Priority: Medium

Preconditions
1. User has registered
2. User's registration has been approved by an administrator
3. User is logged into the system

Success Condition(s)
1. User successfully views a printer friendly list of users registered for the event

Trigger: User selects "print attendance list" link from an event's information page
Extensions & Error Handling
1. Attendance list is unavailable: 404 error - contact administrator

Required Database Entries
List of Registered Users Table {User ID}

Additional Information

Use Case 12: Editing the Member Group of User(s)
Primary Actor: Administrative Members
Priority: Top
Preconditions
1. User has registered
2. User's registration has been approved by an administrator
3. User is logged into the system
4. User has administrative privileges

Success Condition(s)
1. User successfully reaches "edit member group" page
2. Users are listed with their current member group
3. User is able to make necessary changes to the group assignments
4. Upon submission the changes are made to the user table in the database

Trigger: Administrative user selects "edit member group" link
Extensions & Error Handling
1. Member group edit page unavailable: 404 error - contact administrator
2. User attempts to edit their group: Display error explaining the user is unable to edit their own member group. Offer to return to the member group edit page
3. User does not have administrative privileges: 550 error - Your account does not have permission to access this page.

Required Database Entries
User Information Table in Database {Registration Time stamp, uniquely generated user id, first name, last name, unique email, password, Member Group, Approval Status}

Additional InformationUsers will be unable to edit their own member group to ensure that there is always at least 1 administrator in the system. Additionally it ensures that the administrator is theoretically a member with the ability to make changes to the system in the foreseeable future.

Use Case 13: Submit Event Attendance
Primary Actor: All Members
Priority: Medium
Preconditions
1. User has registered
2. User's registration has been approved by an administrator
3. User is logged into the system
4. User is submitting attendance for an event they created/have ownership of

Success Condition(s)
1. User successfully reaches attendance submission page for a particular event
2. Users are displayed successfully
3. User is able to select the users which attended the event
4. Upon submission any assigned task completion values given to the event are updated in the member task list table

Trigger: User selects "submit attendance" list from an event's information page
Extensions & Error Handling
1. Attendance submission page unavailable: 404 error - contact administrator
2. No users selected upon submission: Display error page explaining "There were no selected users in the attendance list" They are given the option to go back to the "submit attendance" page.
3. User does not own event: 550 error - Your account does not have permission to access this page.

Required Database Entries
Tasks Table in Database {Uniquely Generated ID, Task Title, Assigned Member Group, Completion Value}

Additional InformationTo accomodate for last minute attendees all users will be listed here. The registered users should appear at the top of the list.

Use Case 14: Editing Existing Information Page
Primary Actor: Administrative Members
Priority: Low
Preconditions
1. User has registered
2. User's registration has been approved by an administrator
3. User is logged into the system

4. User has administrative privileges
5. At least one information page has been created in the system

Success Condition(s)
1. User successfully reaches page to edit the Information Page form
2. User is able to make necessary edits to the form
3. Upon submission the editted information page .html file is regenerated to reflect the edits

Trigger: Administrative user selects "edit information page" link on an Information page
Extensions & Error Handling
1. Information page edit page is unavailable: 404 error - contact administrator
2. Form missing values: Submission button not clickable until all sections of the form have entries
3. User does not have administrative privileges: 550 error - Your account does not have permission to access this page.

Required Database Entries

Additional InformationThis becomes mandatory if Use Case 9 is fulfilled.

Use Case 15: Viewing the list of tasks
Primary Actor: Administrative Members
Priority: medium
Preconditions
1. User has registered
2. User's registration has been approved by an administrator
3. User is logged into the system
4. User has administrative privileges

Success Condition(s)
1. User successfully reaches "list of tasks" page
2. All tasks in the database are listed accurately on the page

Trigger: Administrative user selects "tasks" link from main page
Extensions & Error Handling
1. List of tasks page unavailable: 404 error - contact administrator
2. User is not an administrator: 550 error - Your account does not have permissions to access this page.

Required Database Entries
Tasks Table in Database {Uniquely Generated ID, Task Title, Assigned Member Group, Completion Value}

Additional Information

Use Case 16: Approving user registrations
Primary Actor: Administrative Members
Priority: Top
Preconditions
1. User has registered
2. User's registration has been approved by an administrator
3. User is logged into the system
4. User has administrative privileges
5. There are pending registrations in the system

Success Condition(s)
1. The user reaches page with list of pending user registrations
2. The user is able to review the entry user registration entry
3a. The user approves the registration and the approval status is changed to a 1 in the User Information table in the database
3b. The user reject the registration and it is deleted from the database
4. The user is able to approve or reject all pending registrations

Trigger: Administrative user selects "process pending registrations" link
Extensions & Error Handling
1. Pending registrations page unavailable: 404 error - contact administrator
2. User is not an administrator: 550 error - Your account does not have permission to access this page.

Required Database Entries
User Information Table in Database {Registration Time stamp, uniquely generated user id, first name, last name, unique email, password, Member Group, Approval Status}

Additional InformationIt might be preferable to not email the applicant if the registration is rejected. Future additions might add automatic emailing to the applicant, but it is unlikely that applicants will be rejected unless they are spam related or entirely unaffiliated with the organization.

Use Case 17: Viewing administrative mode
Primary Actor: Administrative Members
Priority: Low(optional)
Preconditions
1. User has registered
2. User's registration has been approved by an administrator
3. User is logged into the system
4. User has administrative privileges

Success Condition(s)
1. User is able to see options related to administrative priviledges

Trigger: Administrative user selects "view administrative mode" link on main page
Extensions & Error Handling
1. Administrative mode page unavailable: 404 error - contact administrator
2. User is not an administrator: 550 error - Your account does not have permission to access this page.

Required Database Entries
User Information Table in Database {Registration Time stamp, uniquely generated user id, first name, last name, unique email, password, Member Group, Approval Status}

Additional InformationThis is an optional addition to the system. It serves to clean up the main page to hide administrator links from non-administrative members.

Use Case 18: Registering for an event
Primary Actor: All Members
Priority: High
Preconditions
1. User has registered
2. User's registration has been approved by an administrator
3. User is logged into the system

Success Condition(s)
1. User receives success notification of registration for the event
2. User is added to the table "list of registered users" in the database for the event

Trigger: User selects "register for event" on an event's information page
Extensions & Error Handling
1. Registration unsuccessful: User is displayed a message stating the registration for the event was not successful and to try again later. If issues persist please contact an administrator.

Required Database Entries
List of Registered Users Table {User ID}

Additional Information

Use Case 19: Mass emailing task completion statuses
Primary Actor: Administrative Members
Priority: High
Preconditions

1. User has registered
2. User's registration has been approved by an administrator
3. User is logged into the system
4. User has administrative privileges

Success Condition(s)
1. User receives a message of success
2. Users receive an automatically generated email detailing the status of their task completion

Trigger: Administrative user selects "email task completion status" link from mail management page
Extensions & Error Handling
1. Unexpected issue with sending emails: Display an error page that describes (if possible) the issue that occured
2. User is not an administrator: 550 error - Your account does not have permission to access this page.

Required Database Entries
Tasks Table in Database {Uniquely Generated ID, Task Title, Assigned Member Group, Completion Value}
Member Task List in Database {User ID, Task ID, Task Progress}
User Information Table in Database {Registration Time stamp, uniquely generated user id, first name, last name, unique email, password, Member Group, Approval Status}

Additional InformationThis may get the feature to select individual users or specific user groups.

Use Case 20: Sending emails with event information to registered users
Primary Actor: All Members
Priority: high
Preconditions
1. User has registered
2. User's registration has been approved by an administrator
3. User is logged into the system
4. User is sending emails for an event they created/have ownership of

Success Condition(s)
1. User receives a message of success
2. Users registered for the event receive an email containing the event's information (date, start time, end time, location)

Trigger: User selects "email event details" from an owned event's information page
Extensions & Error Handling
1. Unexpected issue with sending emails: Display an error page that describes (if possible) the issue that occured

2. User does not own the event: 550 error - Your account does not have permission to access this page.

Required Database Entries
Event Information Table in Database {Uniquely Generated ID, Title, Event Date, Start Time, End Time, Location, Attached Task}
List of Registered Users Table {User ID}

Additional Information

Use Case 21: Retrieving lost password
Primary Actor: All Members
Priority: Medium
Preconditions
1. User has registered
2. User's registration has been approved by an administrator

Success Condition(s)
1. User is displayed a success message that their password has been emailed to the registered email
2. User receives an email with their password

Trigger: User selects "forgot password" link upon failure to login
Extensions & Error Handling
1. Unexpected error occured: Display message saying, "There was an unexpected issue with retrieving your password. Please try again later. If problems persist contact an administrator."

Required Database Entries
User Information Table in Database {Registration Time stamp, uniquely generated user id, first name, last name, unique email, password, Member Group, Approval Status}

Additional InformationThe level of security may vary between versions of the system.

Use Case 22: Deleting an Event
Primary Actor: All Members
Priority: Medium
Preconditions
1. User has registered
2. User's registration has been approved by an administrator
3. User is logged into the system 4. User is attempting to delete an event they created/have ownership of

Success Condition(s)
1. User is displayed a success message that the event has been removed from the system
2. The event information is removed from the database

Trigger: User selects "delete event" link from an event's homepage
Extensions & Error Handling
1. Unexpected error occured: Display message saying, "There was an unexpected issue with deleting the event. Please try again later. If problems persist contact an administrator."

Required Database Entries
Event Information Table in Database {Uniquely Generated ID, Title, Event Date, Start Time, End Time, Location, Attached Task}

Additional Information

Appendix B

Perl Modules installed on WPI's web server

Archive::Tar
Archive::Zip
Authen::SASL
Bit::Vector
CGI
CGI::Ajax
CGI::Application
CGI::Application::Plugin::Config::Simple
CGI::Enurl
CGI::LogCarp
CGI::SSI
CGI::Session
CGI::Simple
CPAN
Captcha::reCAPTCHA
Carp::Clan
Class::Accessor
Class::Singleton
Compress::Bzip2
Compress::Raw::Bzip2
Compress::Raw::Zlib
Config::Simple
Crypt::SSLeay
Cwd
DBD::Oracle
DBD::SQLite
DBD::mysql
DBI
DBI::Shell
DCOP
Data::Dumper
Data::ShowTable
Date::Calc
Date::Manip
DateTime
DateTime::Locale
DateTime::TimeZone
Devel::CoreStack
Digest::HMAC
Digest::MD5
Digest::SHA
Digest::SHA1
Digest::SHA2
Embperl
ExtUtils::CBuilder
ExtUtils::MakeMaker
FCGI
File::HomeDir
File::Temp
File::Which
HTML::Parser
HTML::SimpleParse
HTML::Table
HTML::Template
HTML::Tiny
IO::Compress
IO::Socket::SSL
IO::Stringy
IO::Tee
IO::Zlib
IPC::Run3
List::MoreUtils
MD5
MIME::Base64
Mail
Mail::Sender
Module::Build
Net
Net::LDAP
Net::SMTP::TLS
Net::SSLeay
PDF::API2
Package::Constants
Params::Validate
Parse::CPAN::Meta
Perl
PlotCalendar
Pod::Escapes
Pod::Simple
Probe::Perl
Storable
Sub::Uplevel
Term::ReadKey
Term::ReadLine
Test::Exception
Test::Harness
Test::Pod
Test::Script
Test::Simple
Text::Glob
Text::Reform
Text::Soundex
Time-modules
Time::HiRes
Time::Piece
TimeDate
URI
XML::Filter::BufferText
XML::NamespaceSupport
XML::SAX
XML::SAX::Base
XML::SAX::Writer
YAML

Appendix C

Readme.txt included with the source code.

Javascript calendar courtesy of Full Calendar - fullcalendar.io

Licensing Information for Full Calendar - http://fullcalendar.io/license/FullCalendar is released under the MIT license, which puts almost no restrictions on how you can use it. You can even freely use it in a commercial project. The only stipulation is that you leave the copyright header at the top of each file intact. This is the same license that the jQuery project uses.

****

About initialization of this program

1. There is a script that needs to be run once. This will set up your database with all the tables and initial values necessary for you to get started.

2. You have control over what user groups will exist in the system. Please spend time thinking about what user groups you'd like your system to have. It is my recommendation that user groups are not created unless you will have to the need to restrict access. For example if you have associate members and general members, unless you need to restrict areas between them, one user group for members should be made. The user groups are used only for restricting access and have no representation in the program otherwise.

3. MAKE SURE TO REMOVE THE INITIALIZATION SCRIPT - the initialization script should only be run if you would like to restart the system. All data will be lost including user groups.

4. The initialization step allows you to set an initial admin. This can be an actual member, or a designated administrative account. A user's group can be changed by admins once the application is initiated. It's important that there is at least one admin at all times in the application.

5. Once initialized, your application should be ready to use.

****

Setting up the project

1. Unzip the source code from the .zip file

2. Move the "APP" directory and all its contents into your public_html folder

3. Acquire a MySQL database from http://www.wpi.edu/academics/CCC/Services/Databases/MySQL.html

4. Find the db.txt file by going to directory APP> scripts

5. Edit the first line following the format DBI:mysql:[DATABASE NAME HERE];host=mysql.wpi.edu;port=3306

6. Edit the second line so it's the username for the database (it will be the user/organizational account that created the database)

7. Edit the third line so it's the password for the database. Note that you are supplied a password upon creating the database.

8. Direct your browser to https://users.wpi.edu/~[YOUR USERNAME HERE]/APP/public/initialize.html

9. Enter the information for the first initial administrator and submit

10. Go to https://users.wpi.edu/~alyssagraham/APP/public/setUserGroups.html and set any user groups you would like for your application (e.g. alumni, perspective, etc)

11. You can see your newly created application at https://users.wpi.edu/~[YOUR USERNAME HERE]/APP/public/app

12. Remove the initialize.cgi and setUserGroups.cgi scripts from the APP> scripts directory