By: Colin Harrison, Christian McMurtrie, Timothy Nakhisa, and Zachary Rivera

3/14/2016

COLIN HARRISON, CHRISTIAN MCMURTRIE, TIMOTHY NAKHISA, ZACHARY RIVERA

pie user manual



Fourth Iteration Report

Table of Contents

[Introduction 1](#_Toc445667766)

[Website 1](#_Toc445667767)

[Project Overview 1](#_Toc445667768)

[Problem and Solution 1](#_Toc445667769)

[Stakeholders 1](#_Toc445667770)

[Scope 2](#_Toc445667771)

[Project Management Plan 2](#_Toc445667772)

[Project Organization 2](#_Toc445667773)

[Risk Analysis 3](#_Toc445667774)

[Hardware and Software Resource Requirements 3](#_Toc445667775)

[Work Breakdown 3](#_Toc445667776)

[Project Schedule 3](#_Toc445667777)

[Requirements 4](#_Toc445667778)

[Development, Operation, and Maintenance Environments 4](#_Toc445667779)

[System Model 4](#_Toc445667780)

[Textual Use Cases 4](#_Toc445667781)

[User Interaction 5](#_Toc445667782)

[Functional Requirements 6](#_Toc445667783)

[Primary Requirements 6](#_Toc445667784)

[Secondary Requirements 6](#_Toc445667785)

[Nonfunctional Requirements 6](#_Toc445667786)

[Feasibility 7](#_Toc445667787)

[Architectural Design 7](#_Toc445667788)

[General Constraints 7](#_Toc445667789)

[Data Design 8](#_Toc445667790)

[Program Structure 8](#_Toc445667791)

[Alternatives Considered 8](#_Toc445667792)

[Detailed Design 9](#_Toc445667793)

[Component in Detail 9](#_Toc445667794)

[Page Descriptions 9](#_Toc445667795)

[Database Usage 9](#_Toc445667796)

[Current Methods 9](#_Toc445667797)

[Quality Assurance Plan 10](#_Toc445667798)

[Document Standards 10](#_Toc445667799)

[Coding Standards 10](#_Toc445667800)

[User Interface Guideline 11](#_Toc445667801)

[Change Control Process 11](#_Toc445667802)

[Testing Process 11](#_Toc445667803)

[System Testing 12](#_Toc445667804)

[Conclusion 12](#_Toc445667805)

[User Manual (appendix) 12](#_Toc445667806)

[Introduction 12](#_Toc445667807)

[Installation 13](#_Toc445667808)

[System Requirements 13](#_Toc445667809)

[Special Cases and Error Handling 13](#_Toc445667810)

[Quotation Marks in Text Fields 13](#_Toc445667811)

[Unsuccessful Queries 14](#_Toc445667812)

[Redirection to Database Connection Error Page 14](#_Toc445667813)

[Attempting to Access File Directories 14](#_Toc445667814)

[Accessing User Images through Address Bar 14](#_Toc445667815)

[Appendix A 15](#_Toc445667816)

[Use Cases 15](#_Toc445667817)

[Appendix B 18](#_Toc445667818)

[System Models 18](#_Toc445667819)

[Appendix C 21](#_Toc445667820)

[System Structure 21](#_Toc445667821)

# Introduction

Our team is Computer Applications: Knowledgeable Engineering (CAKE). The members of this team are Colin Harrison, Christian McMurtrie, Timothy Nakhisa, and Zachary Rivera. We are working with our client, Dr. Yvonne Chueh, to create a website similar to “Meal Train”. This website is intended to help the socially isolated in planning, organizing, and executing social events. This document will tell you more about how we will manage our project, a project overview, the requirements, and our plan for Quality Assurance.

# Website

For our team website, we elected to use the website building service at wix.com. This allowed us to quickly create a professional quality site without cost to our team. Two of the biggest reasons for choosing wix.com is the ease in updating and maintaining our site. This allows for visitors to have update to information on our project. Our website can be found at the address listed below.

<http://caketeamcwu.wix.com/cake>

# Project Overview

This project will aim to help people who are facing social isolation. A website will be created to promote social activities as well as more personal outings for our target demographic. It will allow for members of the local senior, disabled, and otherwise isolated communities and their distant friends and families to stay connected.

## Problem and Solution

Rapidly advancing technology has allowed people to remain connected even when half a world apart. A person in Seattle can speak with and see a person in London in real time. Better yet, fly from New York to Los Angeles in just a few hours. These advances have allowed people to move freely around the world and still remain in touch with family and friends from their hometowns.

These changes have also led to a group of increasingly isolated individuals. As more families choose to relocate for work or personal reasons, the parents and grandparents often stay put in their hometowns. As they age and mobility diminishes, socializing becomes more difficult if not nearly impossible. Other target groups follow similar trends. A lack of socializing can lead to reclusion which can contribute to deteriorating health, both mentally and physically.

The solution is to promote social outings for the isolated population, whether by organizing group outings or having the other community members to step in when a family is not available. The vision of Dr. Chueh is to have an online site where such activities can be arranged. This would include having a system where families away from relatives can contribute towards funding events or meal outings that allow the friends and family members to socialize outside of their own home.

While software promoting social activities is not unique, it often misses some of the most vulnerable demographics. This software would solely focus on those who may not be familiar with or able to use computers or the internet well and who may have trouble getting out on their own.

## Stakeholders

The client is Dr. Chueh. She is a Professor of Mathematics at Central Washington University in Ellensburg, WA. She was inspired to do something for the isolated community during a trip to the east coast. During this trip, she attended a funeral where the family of the departed received food and condolences from members of an online site called “Meal Train”. She recognized the good that can come from a community banding together in times of need and thought of her own family. She saw the limitations of the Meal Train site and wanted to create something that was more proactive.

The socially isolated in communities across the globe are also stakeholders. For this project, we will limit this to the isolated in Ellensburg. They currently cope with the problem of isolation by attending (when possible) small gatherings at churches, event halls, or sometimes events hosted by local retirement homes. There is no prevailing solution.

The friends and families of the isolated who do not necessarily live near-by and are unable to visit regularly are also groups of stakeholders. These people care for their friends and family members but are rooted in homes across the country making their ability to help impractical with current resources. This site would give them a tool that will allow them to help their friends and relatives maintain healthy social lives.

## Scope

The software will help alleviate the problem by providing a dedicated site to increasing social health among the socially isolated. It will provide an easy to use, senior and disabled friendly interface that will help people connect to other members of the community.

It will also create a place where distant family members can help their loved ones stay healthy and connected to society. This should reduce some of the stress that can be caused worrying about a friend or family member.

The site will provide a much needed tool to promote healthy social living, but the weight of utilizing the tool will still rely on people. Family members and other connections will have the ability to donate to the isolated, who may not have sufficient income to pay for their own outings. Other members of the community will be allowed to volunteer their time, and friends and family members will need to use good judgement when accepting a request to take out the isolated individual.

# Project Management Plan

Our project management plan will follow a Kanban/Agile method of development described below in more detail. We chose this style because of its focus on quality over pushing out code as fast as we could. This will help us to ensure the client receives the product that they have requested at a highest potential quality.

## Project Organization

We will be using an agile Kanban method for solving our problem to keep it organized. This will make it so we can pair program, keep in contact, and have a good developer to quality assurance ratio to constantly push out a refined product each sprint for the client. To manage our Kanban board we will use a tool called “Trello”. Our board will have six columns a backlog, defined, and developer finished for the developers to use on the left side of the board. The right side will consist of accepted and deployed for the quality assurance. Each person will be assigned one task at a time, and this task will not leave a column until it is finished, but it can move from one column to another if it needs to. This will help our client see a consistent growth, and help us keep on track of where are project is at, where it’s going, and what changes might have to be made. Currently we will have three developers (Christian, Tim, and Colin) and one quality assurance tester (Zach). We also have designated our scrum master to be Zach, our documentation manager is Colin, and version control system manager is Tim. We will be rotating these rolls every four weeks to make sure that everyone can experience each roll.

## Risk Analysis

A large risk for our project would be feature overload. To make this website fully functional we must implement a lot of features, but we most focus on them one at a time. The agile Kanban organizational method will help us to not get overwhelmed, and consistent interactions with our client to be able to update, and manage expectations. Below in Figure 1 is a list of different risks that could arise, the impact on the product, the feasibility to overcome, and the priority to get the risk resolved.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Issue** | **Impact** | **Feasibility** | **Priority** | **Resolution Strategy** |
| Compatibility | HIGH | HIGH | HIGH | Use multiple systems and common browsers for testing |
| Readable Text | HIGH | HIGH | HIGH | Larger and user sizable text |
| Volunteers | HIGH | MED | MED | Talk to sociology department and Elmview workers for connections |
| Share Photos | MED | HIGH | HIGH | Look into a server that could handle larger photos |
| Paid Option | LOW | MED | LOW | Research how to implement paid options |
| Languages | LOW | LOW | LOW | Add Google Translate to the page |

Figure 1

## Hardware and Software Resource Requirements

For our project we will each need a computer with the most widely used browsers installed, such as Firefox, Chrome, Edge, and Safari to be able to test our website on. There are also third party programs we can use that would let us tap into a multitude of different browser and operating system configurations called Sauce Labs. Furthermore, we will be using “GitHub” for our repository and Microsoft Office for reports and presentations. We will also need html editors such as IntelliJ, and MySQL or another way to manage databases of user information. Our target installation will be at least the main browsers Firefox and Chrome, then hopefully expanding to others as the project goes on. The user will also need a reasonably low specification computer described in the requirements section later in this document.

## Work Breakdown

We will be using an iteration “onion layering” breakdown where we will start by trying to get a homepage set up. After that we will populate the home page, and get tab placeholders for the next pages. We will tackle each feature one task at a time to get the project done efficiently. Each major feature will be a milestone, and will get broken up into many different “stories/tasks” so that we can add it to our Kanban board. Every two weeks we will pick a feature and create the stories needed to finish that feature, and at the end of the two weeks we will look through what we have done and what milestone is next. That way we have a base to build upon getting closer to our end product the client has requested while maintaining a functional product.

## Project Schedule

Our goal is to reach each milestone every sprint (which is every two weeks), and updating our tasks during our scrum meetings (a few times a week). We will hopefully be continuing to build upon each previous sprint so our tasks would be changed to correspond with the milestone goal. We would have a retro after a sprint and document what got done to help project, our next sprint and if we are on track. With Kanban we could tackle our project in two ways: having separate developers each working on one task at a time or by using paired programming for features to continue the “conveyor belt” while having quality assurance continuously checking the developer’s latest features implemented.

# Requirements

The scope of the project leads to a reasonable amount of requirements, which are listed below. These include: hardware, system, user, functional, and non-functional requirements. We have included the tools that will be needed to achieve our goals and meeting our client’s requirements. These will help ensure that our project stays on track, not straying away from Dr. Chueh’s vision. We will present our requirements list to Dr. Chueh for review.

## Development, Operation, and Maintenance Environments

The hardware and software resources needed to build and maintain the project will be listed below. These requirements may change over the course of the project if issues arise, but conceptual requirements should not change significantly.

The hardware that will be necessary to build and maintain the project will include:

* Computers with multiple operating systems and common browsers
* Server, localhost will be sufficient for development
* MySQL database

We will be using a wide variety of software development tools including but not limited to:

* Notepad++ or similar
* IDE such as Eclipse, IntelliJ, Sublime Text, etc.
* Apache2 server
* GitHub
* Trello
* Word documents
* Sauce Labs
* Multiple internet browsers such as Firefox, Edge, Chrome, etc.

## System Model

We have used textual use cases to describe certain interactions between user and site. These use cases are not all inclusive, but rather show the most commonly used interactions. Visual use cases are also included in the appendices.

### Textual Use Cases

Create Account

1. Create Profile

* 1. Username not available, try new username
  2. Password does not meet requirements, choose different password

2. Create Connections

2.1 Send out connection invites

2.2 Accept invited connections

2.3 Reject invited connections

Create Event

1. Select Create Event button
2. Select Event Type
   1. Select type potluck
   2. Select type outdoor activity
   3. Select type indoor activity
   4. Select type restaurant
3. Create Name
4. Create Date(s)
5. Invite
   1. Invite connections
   2. Make public

Login

1. Open site
2. Enter Password
   1. Incorrect password
      1. Reenter password
      2. Account Locked after 3 attempts
   2. Incorrect username
      1. Reenter username
3. Login

Donate

1. Login
2. Select recipient
   1. Select from connections
   2. Select donate to site
3. Donate funds
4. Recipient receives funds

(See Appendix A for use cases and Appendix B for system models)

## User Interaction

The program for the user is straight forward. The user will be able to interact with the website, and see other people in the area that want to participate in other social activities. Whether they be the host, looking for a friend or family member, or a group to have a meal with. This information will be on the main page of the site for fast accessibility. The user will also be able to upload pictures or videos that they desire, that is related to organizing meals or giving other users an idea of what outdoor activities they are in to or content that has to do with their PIE experiences.

Inappropriate images or posts will not be tolerated neither do we condone such actions, this is not a dating or chatting site. The user will also have the ability to fill out a form for when they want to volunteer and host a meal or vice versa.

## Functional Requirements

The functional requirements for our project are included below. They include primary and secondary requirements. Primary requirements are vital to the production of our site and will need to be completed. Secondary requirements are optional, functions that we would like to see added to the site, but are not necessary to complete.

### Primary Requirements

* We need to host the website on a server
* We need a database to store information for the forms
* We will have a calendar for users to book days where they will be organizing meals or other outdoor activities.
* The website will be free for the time being until further development.

### Secondary Requirements

* Determining how users will pay for the services the website offer. WePay, Amazon pay, etc.
* Translating to different languages for native speakers.
* Get people to join the website.
* Testing our scheduling services from a user’s perspective on the site.
* We plan on having a third party service that we can implement to check the background of users/volunteers who get added on the site. This will be for security purposes.

## Nonfunctional Requirements

A user will be able to use our website for their needs anywhere in the world since it is in the internet. Our project is based to be used in the United States as far as the physical environment is concerned. Specifically, we will focus in Ellensburg for testing purposes but the functionality will not differ by city or state, except for the restaurants in that particular town and we can adapt to that by embedding a location service/GPS API.

Conceptually, how efficient it will be in countries that are not as advanced as the United States is something that we cannot fully determine currently. Some of the factors that might hinder for our project to be something of value to others across is the world would be things such as: background checks for new users, paying system, and restaurants. But, if someone out of the country would like to assist their friend/family member in the United States, they would be able to do that with minimum limitations.

For nonfunctional requirements here is how the system will adapt to some of those instances:

* **Efficiency**- The system itself will be quite efficient in terms of providing the service that the user expects the program to provide. This will be achieved since our website is catered to a few specific functionalities that ensure the vision statement is met without too much jargon. The user interface is designed with the concept of simplicity and no learning curve. The text and diagrams are large enough so elderly people will have little to no problem to navigate through the site.
* **Reliability**- This will not be an issue for the user since our implementation is streamlined concisely with what is necessary. The only factor that may affect the system’s reliability is whether users actually sign up and interact with other users and arrange meals and outdoor activities, because if this is not accomplished, the site serves no purpose for the user.
* **Portability**- This won’t be a major issue since majority of users have access to internet. Whether that be via a cellphone or a computer. For those who may not have either, libraries, school labs, and cyber cafes are other considerable options. The only downside with a user accessing it on a cellphone is that all of the content may not fit on the screen or be displayed as it would have been on a computer. Further down the road, this will be something to consider, a mobile version for our website.
* **Problem Size**- The problem size is big enough to make the project fun and challenging. For instance, we have to figure out how we will approach background checks without making the user feeling uncomfortable. We have to find a way of translating languages in a reliable-efficient manner. We are certain that as we implement some of our core requirements, the problem size will be affected, hopefully not too drastically.

## Feasibility

In terms of feasibility, we are certain that we can finish our project by the end of winter quarter with no problem, but we are aware that some minor set backs are inevitable. For example, feature overload is a possibility. Our client wants the website to have a lot of functions, but realistically, we will finish the primary requirements before working on the secondary to ensure a working product by the end of winter quarter.

# Architectural Design

The architectural design section of our project consists of general constraints we came across, the data design, our program structure, and the alternatives we have considered along the way. We have worked both with our client, our user network, and quality assurance to discover these section and bring our project to where it is at today. With all the sections below considered, we are planning to execute all our primary goals for this project.

## General Constraints

In this project, constraints there were both hardware, interface, and external data that we came across. In the hardware aspect of constraints we had issues getting a proper server to test on because none of us had our own server, and were relying on the school to be able to provide one. We overcame this by asking a lot of people how to get a school server for our website, and contacted the computer science department information technology lead. We got a test server through him which we can test on campus only. We will later host this website on a different server so that users can visit the site and we can test on more computer environments, but needed something temporary and free for now.

On the interface we were hoping to make our website both desktop and mobile compatible, but it seems like we might not be able to get the mobile version completed in time because of time constraints. This is a time constraint because we have to think about all the different mobile platforms, browsers and screen dimensions a phone can have. We as a group have both Android and iOS to make testing on these platforms a possibility, but will start with the basic desktop website before considering to move on to a mobile solution.

For external data, we were trying to get global positioning system to find near restaurants for the users, but after doing extensive research our best route is going through a company called Groupon. The problem with this is a time constraint again because they would have to approve our website once it has been published, and we would have to be a Groupon affiliate. We have not heard how long this process takes, but are projecting that it will take at least a month after the website has been published giving us another time constraint. For now we will overcome this by starting locally in Ellensburg Washington and going to the business webpages and manually pulling the local deals. With all these constraints in mind we are finding ways to overcome them and still plan to implement our features, but are not sure if it will be before the quarter is finished.

## Data Design

The structure of the database that we are currently using is not too complex. Some of the fields that we have that are essential to make sure that the database is usable are the following:

* Username
* Password
* Events
* Calendar

Since our website is not heavily focused on processing heavy amounts of data, we have not found it necessary to use any core data structures or algorithms to analyze the data that the user submits to us.

## Program Structure

The structure of our website can be seen in the two figures in Appendix C. The first of which shows the architecture of the site. The second page will show the connection between user viewable pages and our pie database, which has two tables.

In the first figure will be a high level overview of the system architecture where the connections between different files is displayed. For example, signUp.php is a viewable page which contains a form for a new user to sign up. This page will send the form information to a background (non-viewable) page, register.php. Register will connect with the database.php file and upon successful entry of new user, will redirect to the profile page. If entry of the user is not successful, an error message will be generated, the page will redirect to signUp.php where the error will be displayed.

In the second figure, the database connections show how each table connects with the various pages. For example, the register page connects to the user table. Register will make a query on the user table to make sure the username does not already exist, otherwise it will only insert into the database table. The create event page only shares a connection with the events table. This is because the create event page does not make any queries on the table, as there are no unique fields to check. Rather the page will only insert into tuples into the events table. This is because it must be able to edit and display events as well as user information.

The model we used was the Hierarchical Web Architecture. The base of the site is the home page from which, all other pages can eventually be reached. Due to the simple design of our site, most pages are able to connect to one another and there is not much depth where a user can get lost in the site. This model seemed most appropriate for our site due to its relative simplicity.

## Alternatives Considered

We have changed both our page architecture and our full website experience architecture, considering many options along the way. On a macro level we originally considered having all of our web pages in HTML, with a lot of child sites using external CSS and PHP files, but we decided to go with mostly PHP and CSS files and not having a separate file for each interaction on a website. For example, when clicking to edit the users profile we had planned to have the user redirect to a new page where they would edit the profile page, but we decided to change it to a pop up style editor. This is so the user doesn’t feel like they are always being redirected and the fields appear as though they are being edited instantaneously. We also decided to go with mostly PHP pages instead of HTML because it was easier to use PHP to work with the database.

On a more micro level, we have considered many ways to lay out the pages to make them visually appealing and logically functioning. We intended our color scheme to be a light blue and gray because they went together well, but our group and our client agreed that the pages were too bright and hard to read. Now we have a consistent color scheme between all pages with a darker blue to help users with poor vision. We have also created a CSS template for each page and different divisions that allow a very consistent layout for each page. This is so the users feel like they are smoothly transitioning from page to page without surprises. We originally got our ideas for the page from the Meal Train website, but have considered many social media sites such as Facebook, Myspace, and Twitter due to their success.

# Detailed Design

This section includes a detailed description of our site’s design and functionalities. Many of the site’s pages use the same JavaScript and CSS files for layouts and animation. The “Components in Detail” section elaborates on how each file works and what connections are necessary.

## Component in Detail

### Page Descriptions

Our project is using multiple linked PHP pages, CSS pages for styling, JavaScript and jQuery for animation, and a MySQL database for storing and retrieving information. The “Home” page uses HTML and PHP to display a basic description of the site. This page is linked to four other PHP pages including an “About” page that holds more information about our project and the impact that the project has had on the community thus far, a “Help” page that allows users and prospective users to email our group with questions, and “Register” and “Login” pages to allow users to sign up or log in to their profiles. Additional PHP files are used for creating events, an events calendar, database testing, a header and footer, and a sign out page. A FAQ’s page will be added when our team gets more inquiries about the site.

### Database Usage

The majority of the site’s functionality comes from forms that submit user information to the “user” and “events” databases. Upon registration, the user’s information is checked for validity then stored in the database. If a user has registered, they can log in to their profile page. This profile page will be slightly different for each user and will be open for modification to some extent. For example, the “user” database will hold information for a user’s name (first, last, and user names), the date they joined PIE, their email, their description, their birth date, and any photo information. The description, and photos will be changeable by the user. However, only the user name, email, and date joined will be required (to avoid security risks).

### Current Methods

Since we only have a few JavaScript files written for our site, we do not have many methods yet. Currently, one of our JavaScript files, “tabs”, has methods for initiating and showing the tabs, getting the children of the tabs (the information displayed below the tab), and getting hash codes (for specific ID’s). Our other JavaScript file, “sidebar”, uses jQuery to create an animated cascading sidebar for the profile page.

# Quality Assurance Plan

In order to make sure that our website will be of the highest quality, we need to have a solid quality assurance plan in place. This will help guide the team through each stage of development by outlining the quality requirements. Some of the tools we are using are key to the quality of our final product, such as Trello, which promotes quality assurance by keeping tasks organized and well defined. The plan includes more than just tools and is broken up into relevant sections below.

## Document Standards

To ensure consistent and correct documentation occurs in all phases of the project, we have a role called “Documentation”. This role is assigned to one individual who is responsible for maintaining the documentation throughout the current iteration. This role is kept through the entirety of the project and will rotate among team members.

Currently, this role includes: documenting all meeting notes, posting them to the repository, ensuring that the repository folders are kept orderly, and creating meeting agendas. However, the PowerPoint document and the Iteration reports are the responsibility of all team members.

The template for the Iteration reports were selected by the group and it is the template we will use for all iteration reports. The meeting notes format has had minor changes in the layout, but the general format has been consistent; noting start and end time, brief mention of topics discussed and details about the discussion of each topic.

## Coding Standards

Our coding standards list is fairly basic. We developed the list after researching coding standards for other software projects and including rules that we believed would make the files easier to read. The following list contains the coding standards for our project. See figure to for an example of code following these rules is included below.

* Semi-colon on end of all JavaScript lines.
* Use brackets to encapsulate nested JavaScript.
* Close connection to database after use.
* Handle exceptions appropriately.
* Always use alt attribute with images.
* Avoid long code lines, max characters for one line about 80. Should not need to scroll page left-right to read code.
* Declare language at the head of all files.
* Use comments as needed. Do not overuse.
* Save with the right extension - .html, .css, .php, etc.
* Indent all lines four spaces, all nested code four additional spaces.
* Separate all code blocks with single empty line.
* All file names start with lowercase.
* No spaces in filenames.
* If filename is multiple words, first word is lowercase additional words are uppercase. (i.e. signUp.html)

Example:

|  |
| --- |
| <head>  <title>About</title>  <link rel="stylesheet" type="text/css" href="template.css">  <link rel="stylesheet" type="text/css" href="buttons.css">  <div id='cssmenu'>  <ul>  <li><a href="home.html">Home</a></li>  <li><a href="about.html">About</a></li>  <li><a href="helpPage.html">Help</a></li>  </ul>  </div>    <h1>Making Outings as Easy as PIE</h1>  </head> |

Figure 2

## User Interface Guideline

The guidelines for our user interface were developed with our target demographic in mind. We are focusing on easy to see text, large buttons, and minimal side content. The site will be as easy to use as possible, while still providing extensive features.

The consistency of the website will be accomplished by using minimal templates, and reusing the basic ones (such as navigation bar at top of page) on each page. This will allow the user to be comfortable navigating each page of the site, as the key buttons/links will be in the same place no matter where they are on the site.

Since we are developing a site with a senior demographic in mind, we must assume that their internet familiarity is minimal. Even more, we must assume that a significant portion of the users will have at least moderately impaired vision. These two assumptions have significantly shaped our user interface guidelines. They are the driving reason behind large buttons, large text and minimal unnecessary content.

## Change Control Process

Our group will continue to use GitHub as the control change as we find it easy to use and there is little to no learning curve embedded in it.

The process our group will use to protect us from having unseen or creeping requirements is by continuing to use Trello to set our priorities right. We have been setting deadlines on our core objectives and keeping a tight communication to make sure that those requirements are taken care of. Every once in a while the requirements get altered due to unforeseen roadblocks that make it impossible to finish a particular requirement. In such instances we help each other to make sure that we have resolved everything and that the objective is completed.

## Testing Process

Our team will have a multi-step process for testing including unit testing, automation testing, and user testing. This will ensure that we cover functional testing, and that the product will work on multiple different platforms. We will first start by user testing with quality assurance’s personal computer and browser configurations. We will write down our configuration, and steps we did to produce a bug so that the developers can reproduce the bug and try to fix the code. After we have done this we will add the unit test to check and make sure that that bug is not introduced again after we add more features in our code. Then, once we have finished a unit test we will add it to our automation suite so that every time we check in new code, our build agent (Teamcity) will run the unit tests against the new code using Sauce Labs as the host the run the unit test among many different browser and operating system configurations. Doing this will make sure that we have covered unit testing, integration and system testing. We will continuously do these steps while developing our project to make sure that our program covers all aspects of testing.

In order to validate that our site meets the standards of our client Dr. Chueh, we would like to put our site through a series of acceptance tests. However, we do not have guidelines clearly defined. We were given free rein on design without clear standards for the site’s functionality. The only major requirement was to keep her vision intact. To ensure that our design is not straying from this, we will demo our prototype to Dr. Chueh every two weeks during our meetings.

There were however, some features that she asked be included, such as the ability to share photos and create events. These functions will be included in the site, but the testing for them is rather straightforward. Their functionality will be tested during unit and integration testing.

# System Testing

# Conclusion

For next term we would like to integrate a calendar to our website. The calendar will be able to show the days that events are taking place. This could range to simple social activities like walking downtown and needing a companion, or needing a friend while going to have lunch at a local restaurant. We also want to be able to use our database to store more user information like pictures and contacts. We also want to include an event slider under the events tab, start online hosting, and begin user testing. We will also like to start conversations with local restaurants and see if we can promote their pages on our website.



pie user manual

Cake Team



March 14, 2016

COLIN HARRISON, CHRISTIAN MCMURTRIE, TIMOTHY NAKHISA, ZACHARY RIVERA

Central Washington University

# User Manual

# Appendix A

# Introduction

Personal Interactive Environment (PIE) is a social networking website aimed to help potentially socially-isolated individuals connect with others in their community, as well as family and friends across the country. PIE keeps in mind those who may not have a great deal of experience using the internet by prioritizing ease-of-use and simplicity of design.

The site was created at the request of Dr. Yvonne Chueh. Dr. Chueh is a professor and director of the Actuarial Science Program at Central Washington University. She was inspired to create a site that proactively reached out to a sometimes forgotten group of individuals. This idea was actualized by connecting with the university’s department of computer science and proposing the site as a potential senior project.

The purpose of this document is to provide a detailed guide to using PIE and give insight to its inner workings. A tutorial on basic operations will be included along with screen shots detailing several of the steps to be taken. Given the complexity and plethora of possible actions a user can take, the illustrated tutorial will not be all-inclusive, but rather show the most necessary tasks such as; registering to the site, logging in, creating an event, manipulating user profile, searching and adding contacts, etc.

The target demographic of the site includes socially isolated individuals, people with disabilities, and friends/family of the previously mentioned groups. There will be a varying degree of technical expertise among these groups. While some may be well versed in technology, it is important that those who have little to no experience are able to use PIE effectively and with minimal frustrations. Those with the latter amount of experience were considered more carefully than the former.

At the time of writing this manual, there is only one user type, the basic user. An administrative user type should be added in the future for site maintenance. Currently, the site is lacking in community boards, and as those are implemented, it will become increasingly important to have some type of moderation which further presses the need for an administrative account type. Basic accounts can only edit and change their profile and create/modify events.

# Installation

There is no installation required to use PIE. The user need only to have a browser capable of accessing the internet. There may be compatibility issues with exceptionally old browsers, but this should only degenerate some of the styling features of the site, not the functionality.

## System Requirements

The site will function probably with all major browsers including mobile browsers. There is some degeneration on mobile, but this only effects small styling features such as button layout. PIE is still fully functional. The hardware requirements are trivial. PIE does not contain video or any content that would stress a computer. However, the browser must allow for JavaScript to run, otherwise PIE will be unusable.

## Special Cases and Error Handling

Due to the focus on simplicity of design, there are not many special cases or errors that should occur. Some of the most common ones will be listed below, but the list is not all-inclusive. If errors are encountered, please email the team at [caketeamcwu@gmail.com](mailto:caketeamcwu@gmail.com)

### Quotation Marks in Text Fields

During the early stages of PIE the insertion of user-entered text into the database was done in a manner that produced errors when quotation marks were used. This issue was caught during testing and has been fixed in most places. It is possible that an insertion statement has been overlooked and the error may still occur in a rarely used area. If this problem is encountered, please email the team and it will be resolved.

### Unsuccessful Queries

The database queries are checked for success with Boolean logic. In most instances, the error will be handled and the user redirected. In some cases, upon failure, a user will be dropped on a page with an error displayed. The user can click on the back button of their browser and resubmit their request. An error message and redirection will be implemented as development continues.

### Redirection to Database Connection Error Page

If the database is experiencing an outage or some other technical problem, the user will be redirected to an error page informing them of such. The site will be inaccessible as most pages depend on the database connection for retrieving information.

### Attempting to Access File Directories

File directories cannot be accessed during normal use of PIE. A user would have to manually type the address of the directory in the address bar. If the user tries to access the directory, nothing will be displayed except an empty file directory page or a Forbidden message. No files will be listed and the user will not be redirected.

### Accessing User Images through Address Bar

Currently, individual images the user has uploaded are visible to any user that knows the full file path to the image. It is not recommended that user’s upload photographs they do not want everyone to see at this time. The issue is being addressed and restriction to these files will be implemented in the near future.

Accessing site images displayed throughout our site is allowed through the file path. There are no plans to hide these or prevent access through the file path as they are Public Domain Images.

# Appendix B

## Use Cases









# Appendix C

## System Models

Create Connections



Create Event

Create Profile



Donate

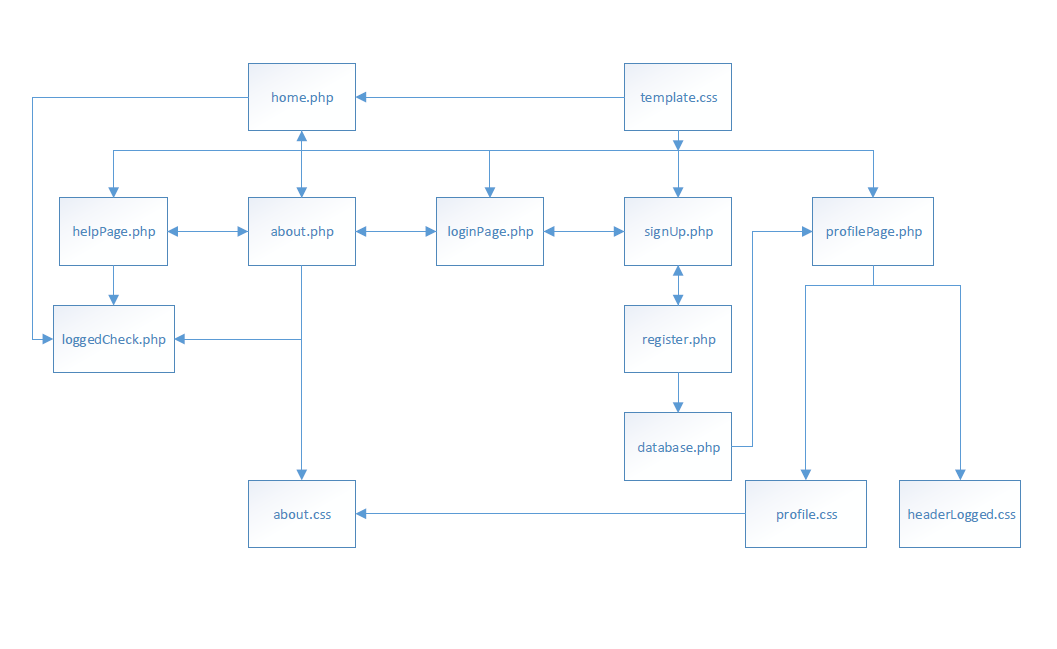


Login



# Appendix D

## System Structure



Web Site Architecture Fig. 1



Database Connections Fig. 2