Web 2.0 Application for a Large Proprietary Software Company

A Senior Project
Presented to
The Faculty of the Computer Engineering Department
California Polytechnic State University, San Luis Obispo

In Partial Fulfillment
Of the Requirements for the Degree
Bachelor of Engineering

By

Justin Rollin Eakins

December, 2009

Senior Project Final Report

Justin Eakins

California Polytechnic State University, San Luis Obispo

1. Introduction

The purpose of this particular project is to design and implement a Web 2.0 application for a group of usability engineers at a high profile company specializing in enterprise software. Many of the individuals at this company believe the common practice of e-mail for communication to be fairly unorganized. Cluttered inboxes lead to lost information and therefore result in an insufficient means of collaboration. The primary goal of the project is to design a tool that facilitates and organizes easy information sharing between all parties involved.

1.1. Client

The client for this project is a small team of Usability Engineers at a large proprietary software company. If the project is successful within this core group of engineers, then potentially the entire company would make use of the application, not only as means of communication but also as a product to market to other companies with similar interests. It should be mentioned the company involved with this project has asked to remain anonymous and much of the detailed implementation to not be disclosed in this report. There is a strong chance this application will be used to share private and secure information and any details leaked could spell trouble for the company. They also intend to market this product to various businesses and do not need the competition to get a jump on their endeavors. For these reasons any further info on the specific client is omitted.

1.2. Related Work

During the planning phase of the project much research and investigation was done on the variety of similar projects readily available to the outside world. Some of the features planned for this application were loosely based off of these projects. Here are a few of the related projects with similar components:

- Digg is probably the most popular social bookmarking site on the web. People from all over the internet can share, vote, and comment on user-submitted bookmarks. This is very powerful because it allows a very large group of people to share content. At the same time, it allows for an individual to discover something that they may find interesting. However, a main weakness of Digg does not allow for small groups to share with each other. There are also many open source clones, which duplicate Digg's functionality such as a project called Drigg.

- Google Reader is referred to as a feed reader. A feed reader aggregates all syndicated web content (such as news, blogs, etc) into a single place. This allows a user to easily view and keep track of all of the content they read regularly. In regards to our project, the main weakness of Google Reader is that it is catered to the individual (we want something where small groups can share and discuss bookmarks). Google Reader does offer some sharing capabilities, but not on the scale that we are looking for.

2. Formal Product Definition

2.1. Problem

The client needs a Web 2.0 application to handle information sharing, storage, and feedback, specifically within small groups to replace their current through-email system.

2.2. Objectives

To deliver an application as described above that reduces the amount of work put in by the user and conveniently and clearly organizes the information for browsing.

2.3. Requirements

2.3.1. Marketing Requirements

The application should...

- ...be fast.
- ...be easy to use.
- ...cater to small working groups.
- ...facilitate and organize information sharing.
- ...be reliable.
- ...be secure.

2.3.2. Engineering Requirements

- Pages should take less than 1 second to load
- The application should work on Internet
- Explorer, Firefox and Safari
- Application will have option to create and send to specific groups
- Application will allow thread creation in 1 click
- The application should allow users to post and, reply to comment
- The application should be accessible 24 hours and crash less than 1% of the time
- The system will have a concept of groups, subgroups, and friends.
- Search must be available.
- Many to many relationships must exist for both tags and groups in relation to shared websites.
- Coding best practices must be followed including use of the MVC (Model-View-Controller) design pattern.

2.3.3. Constraints

The team of usability engineers asking for this project currently uses a Ubuntu web server running both Apache and MySQL.

Must use:

- Ubuntu Web Server running Apache
- MySQL database

2.4. Criteria

The criteria for this project are limited due to it lacking hardware and being an internal system for a private company. Due to the influence of the current clients, this application will need to be easy to set up and install. With a potentially large scale of users interested in the product in the future, it will require seamless compatibility across all major browsers.

2.5 Legality

The final product must contain no third party technologies or code whose use or retail distribution violates said components' licenses in any way. This application also must not contain any third party technologies or code whose use or retail distribution requires a paid license without the expressed and informed permission of the company responsible for this product.

3. Design

3.1. Team Process

Work was completed in conjunction with the company’s engineers as well as two additional interns. It was discussed agreed upon the best course of action would be to take an agile, team process approach. This involved weekly revisions to the sit and weekly conference meetings conducted over the phone. Tasks were handed out to individuals through electronic tickets, where team members could view their tasks and assignments through Eclipse PDT and in conjunction with the Mylyn plugin. Once ticket was completed by the assigned team member, the resulting code was submitted through Subversion to the test server. Testing was then done at each step of development. This process is laid out in more detail under the "Test Plan" section below.

3.2. System Architecture

The system architecture for this product is typical of what you might see on any other web application. The user uses a web browser to communicate with an Apache server. The Apache server renders and serves the application's code (in PHP), and communicates with a MySQL database for storing and retrieving information.

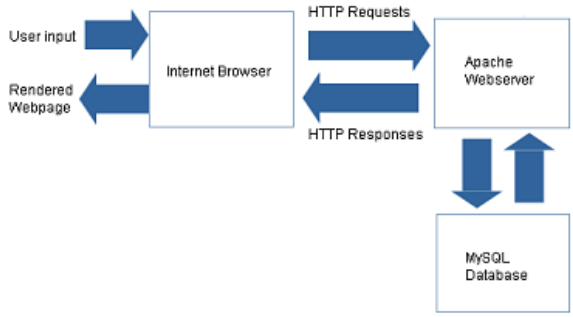


Figure 1: System Architecture

3.3. Database Schema

The database schema (included in the appendix) reflects the tables necessary to accommodate all of our base features including group memberships, social bookmarking with the ability to share bookmarks with groups, and the ability to comment on and tag bookmarks. User statuses can also be updated and tagged. The normalized database structure and naming conventions are chosen to comply with CakePHP's naming conventions and the models which are related to each table. The models define the relationships between the tables within CakePHP and allow CakePHP to automatically create optimized and recursive queries for related information.

3.4. Development Server

Until the final product is ready for deployment on the company's internal web server, the development and testing of the site will be from a development server running on similar software to our target platform. The development server is running on Ubuntu JeOS (Just enough Server OS) 8.04.1, the latest bare bones server version of Ubuntu. The OS itself is installed within a VMware instance running in VMware player on a Windows XP host. However, since we are running a virtualized server, we can move the development server onto any hardware and if desired by the client, the development server's image can simply be copied to the client's server on top of VMware Server (free) with little necessary configuration.

The server is running the latest version of the XAMPP package which includes Apache, MySQL and PHP as well as many necessary extensions such as OpenSSL and LDAP. All administration functions are protected by basic HTTP authentication. phpMyAdmin is installed on the server for easy web based MySQL management.

A web based push script was written in order to facilitate the updating of the code on the web server directly from the SVN repository being used. The push script can be used to push out either the HEAD revision or any other specified revision from /trunk to the web server. The ease of using a web based script to synchronize the web server with SVN allows for the ability to test the most recent HEAD revision without having to worry about breaking the entire site. If any problems are encountered, a previous revision can be pushed to the web server until serious bugs are resolved.

3.5. Design Decisions

Most of our design decisions were made during the technical requirements stage of the planning process. Much research and investigation was put into using various open source products as a starting platform for this specific application; however, it was later deemed not the most productive decision. It was decided that a cleaner and easier to build product would be possible if the site was built from ground up using various frameworks.

The CakePHP framework was agreed upon as the smartest choice due to its ease of use, large community support, and strong enforcement of the Model-View-Controller (MVC) design pattern. Using MVC separates business logic from data models and presentational logic (also known as templates). This allows for easier to read code and allows frontend designers to work mostly independently of backend engineers.

The Prototype and Scriptaculous javascript libraries are used for all javascript and will not only ease development but will also ensure cross-brower compatibility. Using CakePHP will also help optimize MySQL queries, automatically prevent SQL injection attacks, and provide helper libraries for authentication and many other components.

Overall, the specific design choices were made to ease learning and development, encourage good coding practices, increase security, and ensure cross-browser compatibility for all frontend elements.

3.5.1. Design Changes

Initially, the plan was to integrate all user logins with the company's internal corporate LDAP server. However, it was learned much later that access to the company's internal LDAP server would not be given. As a result, a modification design would be necessary. To solve this issue it was decided to store all user login information locally.

4. Test Plan

It was decided to use an agile development process that results in new revisions weekly. Each revision will be tested by the individual developer for bugs before it is checked into the path /trunk. Once the code was checked in, other team members did a code review to verify that best practices have been followed and that no obvious revisions should be made. Code will then be pushed to the development server for live testing on a weekly basis.

Once the code was live on the development server, it was tested by all individuals involved, the usability engineers and design interns. Any bugs discovered were reported via Trac's ticketing system.

4.1. Test Results

Once a working version of the site was released to the group of usability engineers, meticulous testing commenced. As a result, all major bugs were resolved including cross-browser compatibility.

5. Conclusion

All major project requirements were successfully completed. Users can share links, rate bookmarks, share user statuses, and add/join groups. We also closely followed the client's specific wireframes, and users can search the site, and filter content the way outlined by the usability engineers.

Overall, the engineers asking for this application were very pleased with the work completed and even sent out an invitation to continue work on the application as an intern. As with any application additional features are always welcomed and encouraged. The final application is a fully operational site that meets all of the initial requirements the company's engineers had requested.

6. Budget and Justification

This was purely a software project, so nothing needed to be purchased. All software packages used in this project are freely distributed.

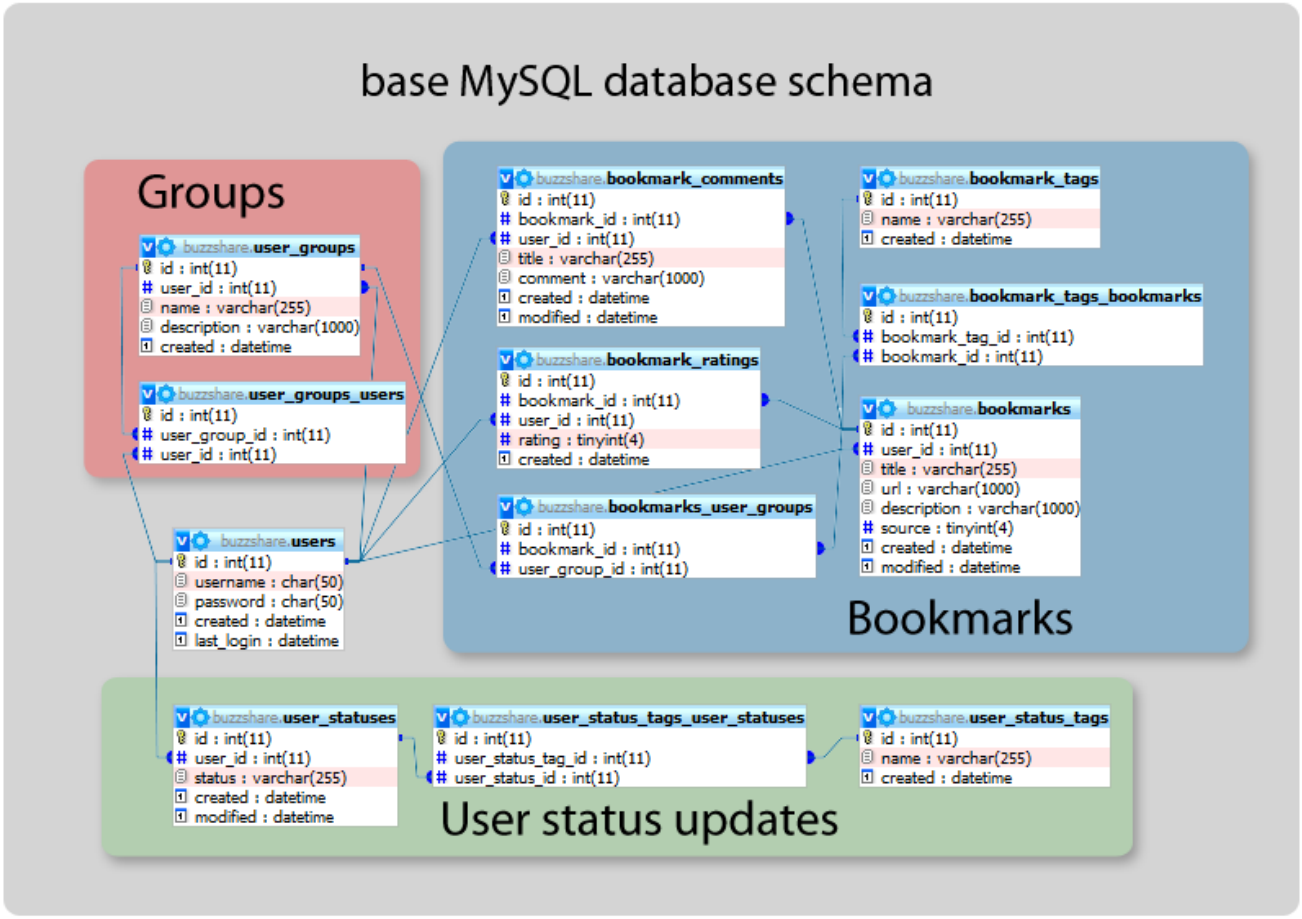
7. Bill of Materials

Nothing was purchased over the course of this project.

8. References

[1] "All things CakePHP :: The Cookbook", Cake Software Foundation, 10 Dec. 2008 < http://book.cakephp.org/ >.

Appendix A: Database Schema



Appendix B: User Deployment Manual

CAL POLY – COMPUTER ENGINEERING DEPARTMENT

Completed Application Deployment Manual

By

Justin Eakins

December,2009

System Overview

The site relies on a few key technologies revolving around CakePHP. A server should be running Apache with PHP support and MySQL to run the application.

System Requirements

This site is operating system agnostic and has been tested on both Windows and Linux platforms. It should work on any server with Apache and MySQL.

Requirements are as follows:

- Apache web server
    - PHP 5 support
    - mod_rewrite enabled
    - .htaccess files enabled
- MySQL server (5.0+ preferred)
    - The application has only been tested on MySQL 5.0+ but it may also work on older versions of MySQL.

Configuring the server

If you do not already have a server set up, you can download XAMPP (http://www.apachefriends.org/en/xampp.html) to set up Apache, PHP, and MySQL in one step. As of this writing, the latest version of XAMPP (1.7.0) requires no configuration except changing the 'root' password in MySQL.

MySQL configuration

MySQL's most of default settings will work fine. The site expects it to be listening on localhost at port 3306. The only change you must make is to change the 'root' password in MySQL to the password given.

Apache configuration

Apache must be configured with mod_rewrite enabled, .htaccess files enabled, and PHP 5 support. If you need to enable mod_rewrite or .htaccess files, you can usually do this by editing the 'httpd.conf' file. If you need more information, please refer to the Apache documentation.

Deployment

To deploy the site, the database and tables must be created in MySQL and the application's code must be placed into the Apache htdocs folder.

MySQL deployment

To create the database and tables for the site, run the following queries in MySQL:

```
SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";

CREATE DATABASE `buzzshare` DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;
USE `buzzshare`;

CREATE TABLE IF NOT EXISTS `bookmark_comments` (
  `id` int(11) NOT NULL auto_increment,
  `bookmark_id` int(11) NOT NULL,
  `user_id` int(11) NOT NULL,
  `title` varchar(255) default NULL,
  `comment` text,
  `created` datetime NOT NULL,
  `modified` datetime NOT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

CREATE TABLE IF NOT EXISTS `bookmark_ratings` (
  `id` int(11) NOT NULL auto_increment,
  `bookmark_id` int(11) NOT NULL,
  `user_id` int(11) NOT NULL,
  `rating` tinyint(4) NOT NULL,
  `created` datetime NOT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

CREATE TABLE IF NOT EXISTS `bookmark_tags` (
  `id` int(11) NOT NULL auto_increment,
  `name` varchar(255) NOT NULL,
  `created` datetime NOT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

CREATE TABLE IF NOT EXISTS `bookmark_tags_bookmarks` (
  `id` int(11) NOT NULL auto_increment,
  `bookmark_tag_id` int(11) NOT NULL,
  `bookmark_id` int(11) NOT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

CREATE TABLE IF NOT EXISTS `bookmarks` (
  `id` int(11) NOT NULL auto_increment,
  `user_id` int(11) NOT NULL,
  `title` varchar(255) NOT NULL,
  `url` varchar(1000) NOT NULL,
  `description` text,
  `rating` float NOT NULL,
  `numratings` int(11) NOT NULL,
  `numcomments` int(11) NOT NULL,
  `source` tinyint(4) NOT NULL default '0' COMMENT '0 - default, 1 - bookmarklet',
  `created` datetime NOT NULL,
  `modified` datetime NOT NULL,
  PRIMARY KEY  (`id`),
  FULLTEXT KEY `url` (`url`,`description`,`title`)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

CREATE TABLE IF NOT EXISTS `bookmarks_user_groups` (
  `id` int(11) NOT NULL auto_increment,
  `bookmark_id` int(11) NOT NULL,
```

```
  `user_group_id` int(11) NOT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

CREATE TABLE IF NOT EXISTS `user_groups` (
  `id` int(11) NOT NULL auto_increment,
  `user_id` int(11) NOT NULL COMMENT 'creator',
  `name` varchar(255) NOT NULL,
  `description` text,
  `created` datetime NOT NULL,
  PRIMARY KEY  (`id`),
  FULLTEXT KEY `name` (`name`,`description`)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

CREATE TABLE IF NOT EXISTS `user_groups_users` (
  `id` int(11) NOT NULL auto_increment,
  `user_group_id` int(11) NOT NULL,
  `user_id` int(11) NOT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

CREATE TABLE IF NOT EXISTS `user_status_tags` (
  `id` int(11) NOT NULL auto_increment,
  `name` varchar(255) default NULL,
  `created` datetime default NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

CREATE TABLE IF NOT EXISTS `user_status_tags_user_statuses` (
  `id` int(11) NOT NULL auto_increment,
  `user_status_tag_id` int(11) default NULL,
  `user_status_id` int(11) default NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

CREATE TABLE IF NOT EXISTS `user_statuses` (
  `id` int(11) NOT NULL auto_increment,
  `user_id` int(11) NOT NULL,
  `status` varchar(255) NOT NULL,
  `created` datetime NOT NULL,
  `modified` datetime NOT NULL,
  PRIMARY KEY  (`id`),
  FULLTEXT KEY `status` (`status`)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

CREATE TABLE IF NOT EXISTS `users` (
  `id` int(11) NOT NULL auto_increment,
  `username` varchar(320) NOT NULL,
  `first_name` varchar(50) NOT NULL,
  `last_name` varchar(50) NOT NULL,
  `password` char(50) default NULL,
  `created` datetime default NULL,
  `last_login` datetime default NULL,
  PRIMARY KEY  (`id`),
  FULLTEXT KEY `username` (`username`,`first_name`,`last_name`)
```

Webserver deployment

Using your preferred SVN client, copy the contents of /trunk from the website provided to Apache's htdocs folder.

Congratulations! If you have configured the server correctly, your site should now be accessible on your server in any browser at http://localhost. To access your site from other boxes via http://<your_server_name>, ensure that your server's firewall (if it has one) is properly configured to allow Apache requests to come in on TCP port 80 (and 443 if you want to access it via https).

If you are still having problems, see the Troubleshooting section for more information.

Troubleshooting

If the site is not working, please verify the following items:

- Apache is running with PHP 5 support
    - .htacess files are enabled
    - mod_rewrite is enabled
- MySQL is running on localhost on port 3306 with a private 'root' password
- If you can access the site from http://localhost on the server but not from other machines via http://<your_server_name>, ensure that port 80 (TCP) is open on the server's firewall. If you want https support, you must forward port 443 also.
- If you are getting errors about a missing database or tables, make sure you have executed all of the SQL in the Deploying Application > MySQL deployment section.
- Ensure that the contents of trunk are in the htdocs folder. The trunk folder itself should not be present.

If none of the above steps resolve your problem, try running all of the above steps on a fresh server without a previous installation of Apache or MySQL. Installing the XAMPP package should require very little configuration as of version 1.7.0.