# Camelopardalis: Final Report

**GROUP MEMBERS**

Timothy Fye
*fyet@oregonstate.edu*

Shannon Jeffers
*jeffesha@oregonstate.edu*

Matthew Morse
*morsmatt@oregonstate.edu*

## Introduction

Team Camelopardalis has developed a fully-functional, web-based employee recognition system to provide companies with a sleek and easy user-interface for creating awards, managing system users, and analyzing data related to employee recognition. The application is partitioned into user and administrative functions that are hidden behind a login screen. The system serves as a tool to enable managers/people in charge of awards at an organization to quickly and easily generate award certificates that can be emailed to the employee receiving the award. Camelopardalis' application also provides business intelligence services to assist organizations in delivering effective employee recognition and merit-based wage adjustments. Given the overall complexity of our site's features, it is safe to say our team learned a lot about web development over the course of this project.

**Table of Contents**

## System Description

There are two different types of user for this project, and what the system does from the user's perspective is dependant on the type of user currently utilizing the system. The two types are "standard users" and "admins." The one thing that is consistent between the two types of users is the login process. There are a couple different things that can happen from the login page. The first is that either type of user can enter valid credentials and be automatically directed to the home page for their type of account. Alternatively, they can enter incorrect credentials, be rejected, and receive an informative error message. Finally, if they have forgotten their password they can click the "forgot password" link which will take them to a page allowing them to enter their email to reset their password. This new, temporary password will be sent to them via email. After logging in, the perspective is dependant on the type of user that is logging in.

From an admin’s viewpoint, there is quite a lot that can be done with the website. Firstly, navigation is very easy and available from every page they encounter. From the home page admins have the ability to see a recent activity log, which lists all of the awards recently created by standard users. Moving to the “manage user” page, admins have the ability to quickly and easily add, delete, or modify users by simply providing their name and email address. They also have the ability to add, delete, or modify admins from the “manage admin” page. Admins can also access a myriad of information from the “business insights” page. The “business insights” page allows admin to view charts and breakdowns of award information based on users, recipients, managers, and branch location. This information can either be viewed directly on the site or downloaded as a CSV or PNG file for further use and manipulation. Finally, admins have the additional capacity to change their password and log out of their account.

There are also quite a few functions available from a standard user’s perspective. Just as with the admin account, standard users are able to easily navigate from page to page from anywhere in the standard user website. The user homepage allows standard users to view and delete the awards they have issued. When clicking on the “create award” tab the user is directed to a page allowing them to search for a recipient to give an award to. If the recipient exists the user is directed to a form allowing them to create the award they want to create. If the recipient doesn’t exist, the user is directed to a form to create a new recipient and is then redirected to the form to create the award for the new recipient, which is then emailed to the recipient. From the “my account” page standard users have the ability to change their name, their password or their signature. They can change their signature either by signing the signature pad or manually uploading a signature image. When accessing the “add recipient” page from the navigation bar, standard users are able to add a new recipient and be directed to the “edit recipient” page to see a list of all the recipients in the system. From the “edit recipient” page, standard users can edit and delete existing recipients. Finally, standard users can log out of their page and be returned to the login page.

## UML Activity Diagram

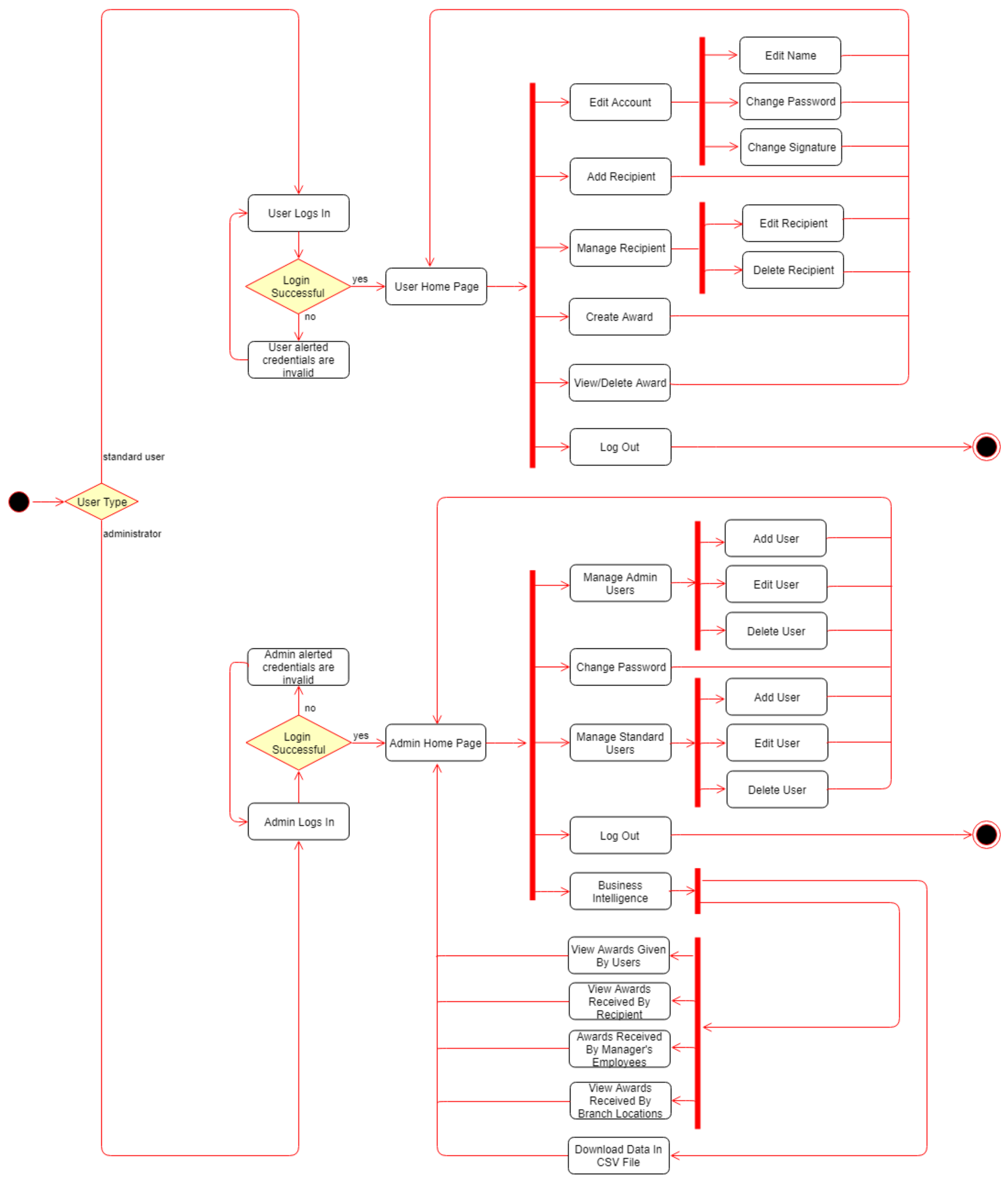


***Figure 1.*** *UML Activity Diagram*

## Usage Instructions

Given that our site is divided into two distinct halves, you will be required to use two separate sets of login credentials to access them. To get you started, we're providing the email address and password to an account we created for the sole purpose of handling our system's automated email messaging services.

Email address: camelopardalis.awardhub@gmail.com
Gmail account password: **cs467group!@#$**

For this demonstration, we've also given this email account administrative privileges on our site.

AwardHub administrative account password: **Dogs4ever**

To ensure you have access to all parts of our site, we recommend that you explore the system using the following instructions in the order that they are provided. If you find yourself locked out of the site for any reason, please contact us by using the contact information listed above. We can easily set up a new account for you.

## Initial Login

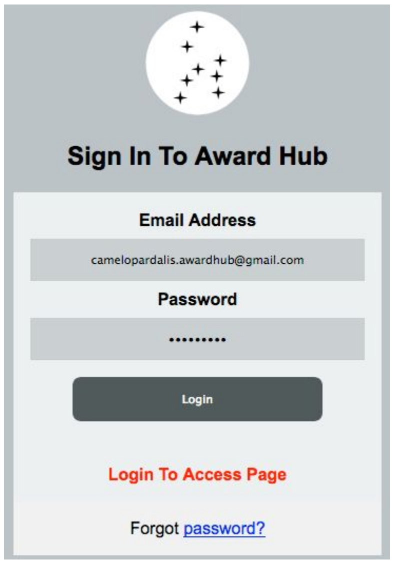
To begin, you'll need to log in to the administrative portion of our site. Enter the following address into your browser of choice (Chrome, Firefox, Safari, Edge, and IE10+ are supported--though, the rendering of some elements will vary):

http://18.188.194.159/award-hub/admin-side/admin-home.php
Or
http://18.188.194.159/award-hub/login-system/login.php

Now enter the following email address and password in the corresponding login form fields:

Email: camelopardalis.awardhub@gmail.com
Password: **Dogs4ever**

After submitting these credentials, you should be taken to the admin home page. While you're there, try resizing your browser window and watch for the responsiveness of the page elements. Also, feel free to test out the "activity log" table date-range filter.



Sign In To Award Hub

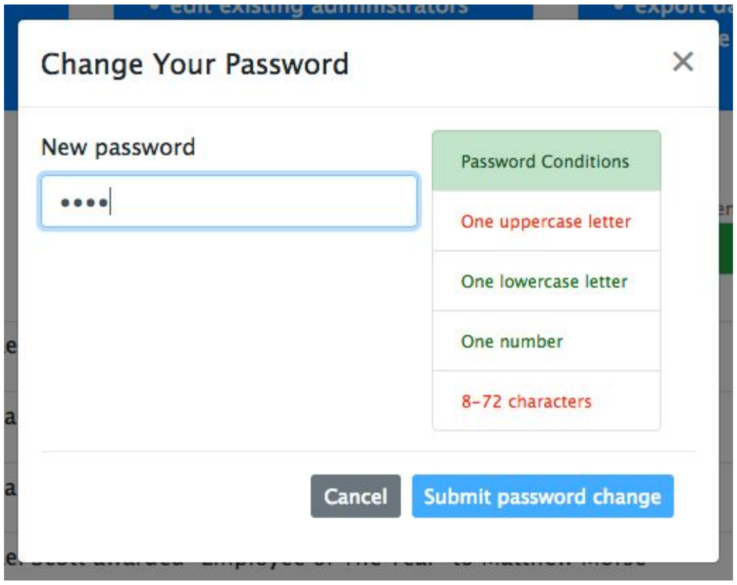
Email Address: camelopardalis.awardhub@gmail.com

Password: •••••••••

***Figure 2**. Login form*

## Change Admin Password

At any point while exploring the admin-side of our site, feel free to use the reset password feature which can be accessed from the button located in the top-right corner of our site. The password provided must meet a given set of conditions. Once approved and submitted, an email will be sent to the address you're logged in with notifying you that your password was updated.



Change Your Password

New password: ••••

Password Conditions
- One uppercase letter
- One lowercase letter
- One number
- 8-72 characters

***Figure 3.** Admin password change modal*

## Create a New User Account

The next order of business you'll have to attend to is creating an account to access the user portion of our site. To do this, click on "manage users" in the sidebar or the card by the same name on the home page. Once there, select "add new user" in the top-right corner of the page. The following modal will appear:

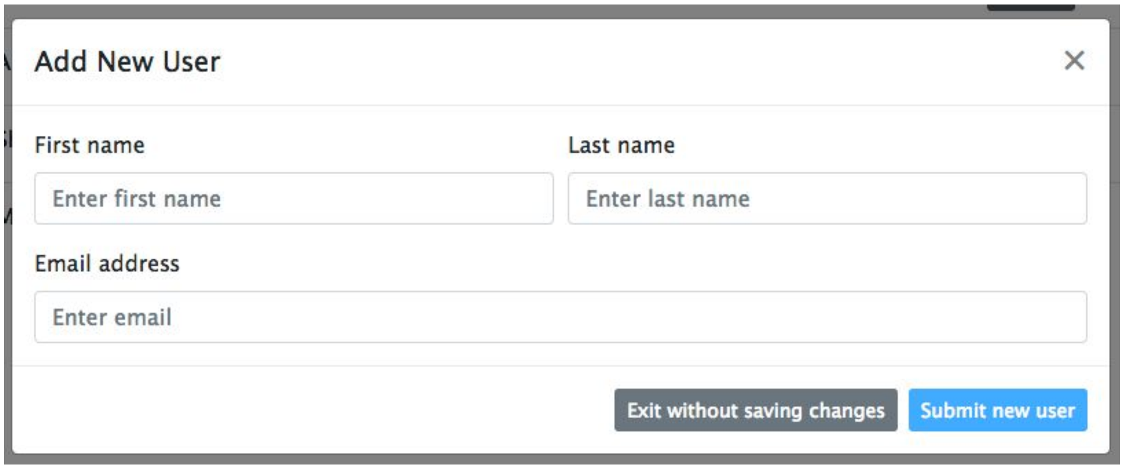


***Figure 4.*** *Add new user modal*

Enter your name and email address (you can delete your record later). As you're doing so, notice that the "Submit new user" button is disabled until you've supplied valid input for all fields. If you'd like, you can try entering one of the existing emails listed in either the "manage users" or "manage admin" pages. Submission of the form will be disabled and an error message will be displayed below the field if it matches a record in the database. This prevents both duplication and allowing one account to access both sides of the site with the same email address.

Once you've created your account, you should see an entry added to the table of users. If the email address has been input correctly, you will receive an automated email at the address you gave with a temporary password and a link to the login page (http://18.188.194.159/award-hub/login-system/login.php). In practice, this can either happen instantly or take up to a minute. The sender should be listed as "Award Hub". Check your spam folder if the process takes longer than expected.

## Manage admin and user accounts

While you're on the "manage users" page, feel free to edit any of the user accounts by clicking on the "edit" button corresponding to a table record. You're welcome to try creating and deleting accounts using the given controls--just be warned that the charts in the Business Intelligence section will start to look pretty drab if you delete existing records. Also, make sure that your own email address is corrected before you go to log in to the user-side of the site.

The functionality and layout of the "manage admin" page is very similar to the "manage users" page. Test its features as you see fit. Again, be careful not to delete the "camelopardalis.awardhub@gmail.com" account. You won't be able log back into this side of the system without it unless you create a new admin account with a valid email address beforehand.

## Business Intelligence Tools

While you're looking at the admin-side of the site, please be sure to check out the business intelligence tools accessible from the "Business Insights" link in the sidebar. Tables and graphs in these pages were developed using Google Charts and are populated with data from award granting activities. Queried data may be downloaded as CSV files to your desktop using the links provided. The charts in the sub-pages may also be exported to a new tab to save as files or print.

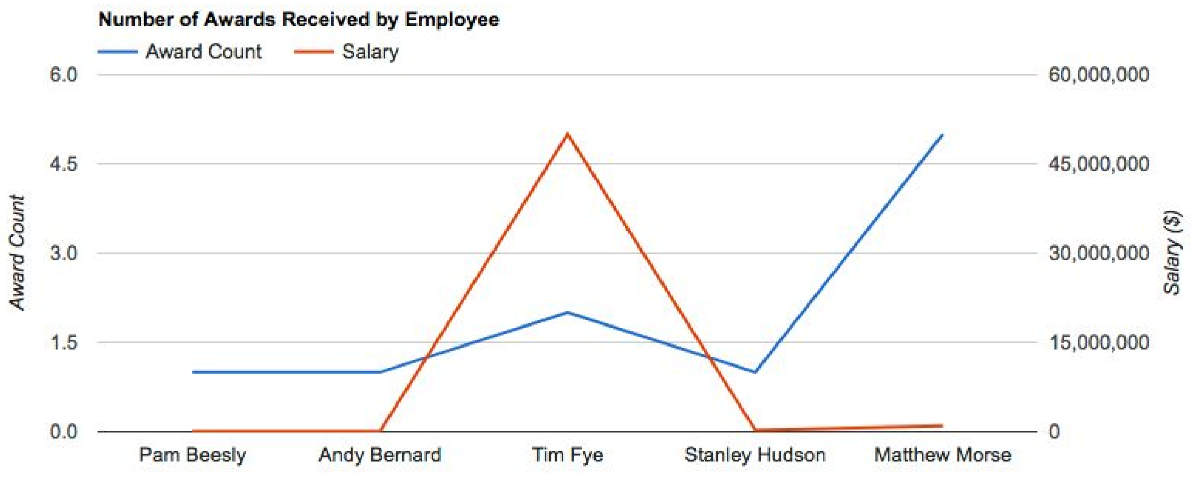
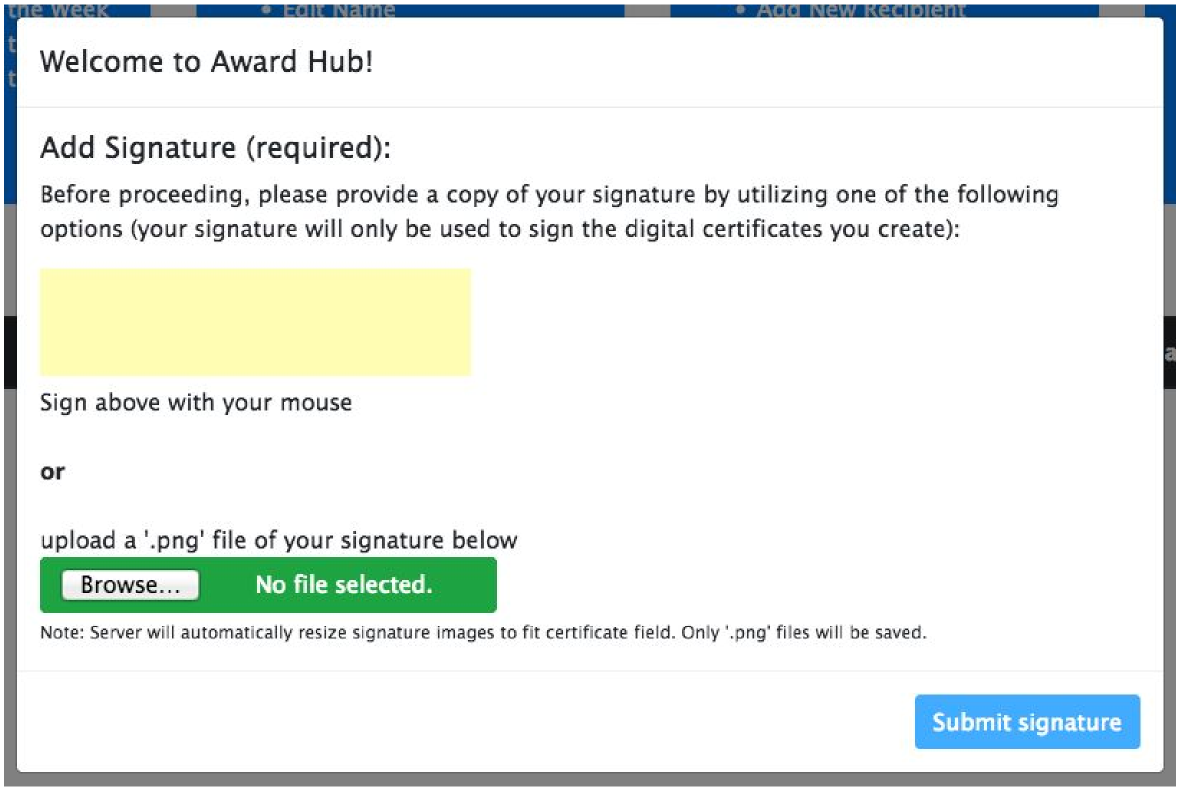


***Figure 5.*** *Business Insights section chart*

Once you're ready to head over to the user-side of the application, just click on one of the page's logout links. Enter the email address of user account you just created along with the temporary password sent to you in the account creation message.

## Signature creation

Upon accessing the user-side of our system (through http://18.188.194.159/award-hub/login-system/login.php) with the credentials you created while following instructions on page seven above, you'll be required to supply a signature if you don't already have one stored in the database. You are provided two options: write one using the canvas element or upload a .PNG file with your signature. The form will be disabled until an input element is altered. You can change your signature later by using tools found in the 'my account' page.


Welcome to Award Hub!

Add Signature (required):

Before proceeding, please provide a copy of your signature by utilizing one of the following options (your signature will only be used to sign the digital certificates you create):

Sign above with your mouse

or

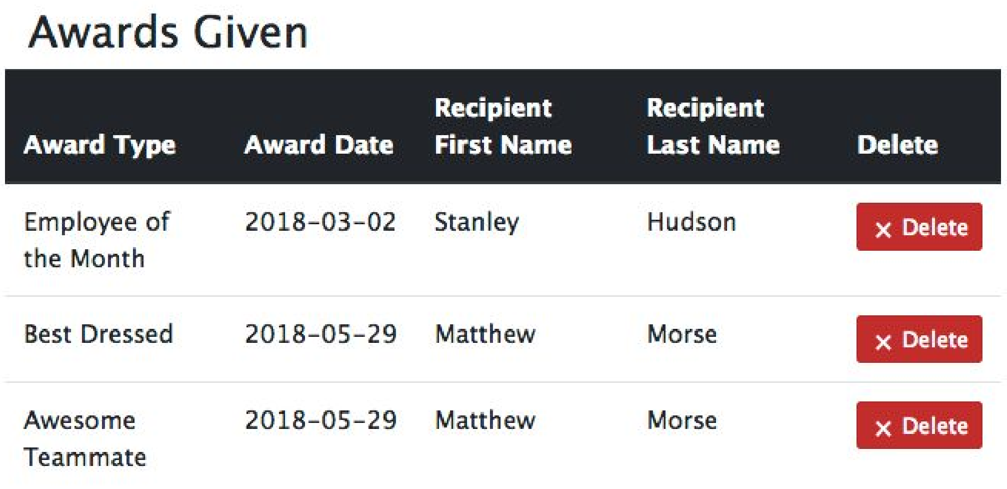
upload a '.png' file of your signature below

Note: Server will automatically resize signature images to fit certificate field. Only '.png' files will be saved.

***Figure 6.*** *New user welcome modal*

## Delete Previously Given Awards

When you get to the user home-page, you'll have immediate access to a table listing awards you've given. This will be empty the first time you login. However, you can utilize the delete buttons in this table to remove an award record from the database after you create one.



Awards Given

| Award Type | Award Date | Recipient First Name | Recipient Last Name | Delete |
|---|---|---|---|---|
| Employee of the Month | 2018-03-02 | Stanley | Hudson | × Delete |
| Best Dressed | 2018-05-29 | Matthew | Morse | × Delete |
| Awesome Teammate | 2018-05-29 | Matthew | Morse | × Delete |

***Figure 7.*** *Awards Given table*

## Creating Awards

It would be somewhat simpler at this point to create awards if our database were modeling a real company. Alas, Dunder-Mifflin is the stuff of fiction. We'd recommend sending award certificates to the OSU email addresses of the Camelopardalis team (fyet@oregonstate.edu, jeffesha@oregonstate.edu, morsmatt@oregonstate.edu) or any other existing address of your choosing. Just not the fake Dunder-Mifflin ones. As for receiving an award, the simplest option would be to set yourself as the recipient and watch your email for a new message from "Award Hub" with your digital award certificate attached.

To begin, select the "create award" link in the sidebar. You will be asked to enter the email of a target recipient. If a record is not already associated with that address, you will be prompted to create one before sending the award.

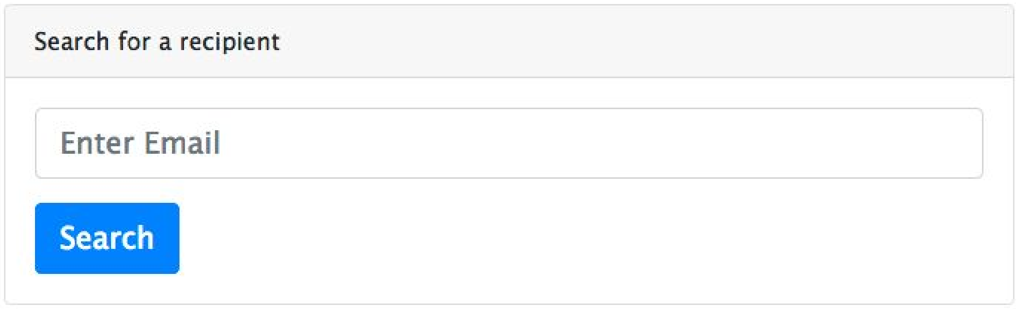
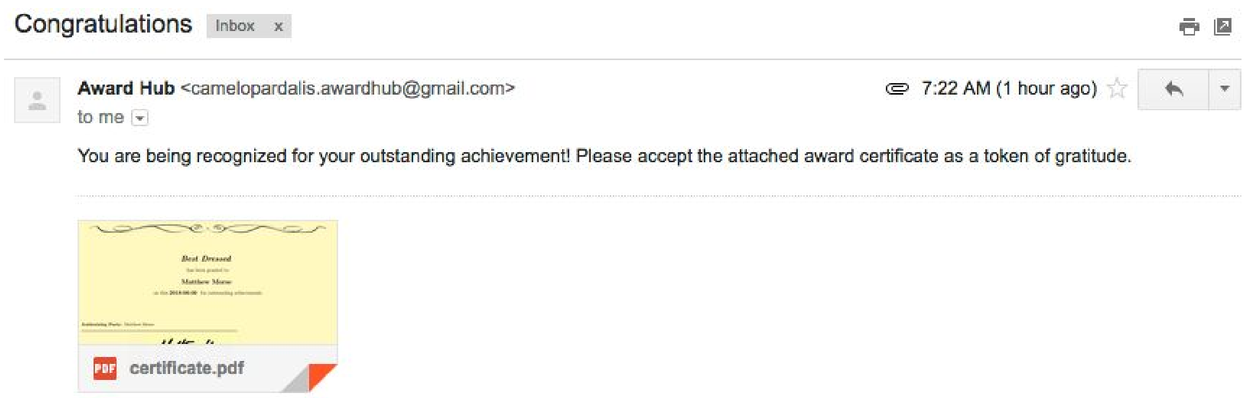


***Figure 8.*** *Recipient email search form*

When the given email is already in the system, the award form fields will be pre-populated and you will only need to select the award type, time (this field format varies depending on browser--AM/PM must be specified in Firefox) , and date. Once the award is submitted, a certificate is created at the server and sent to the specified recipient as an email attachment. This process can take a few minutes to complete.

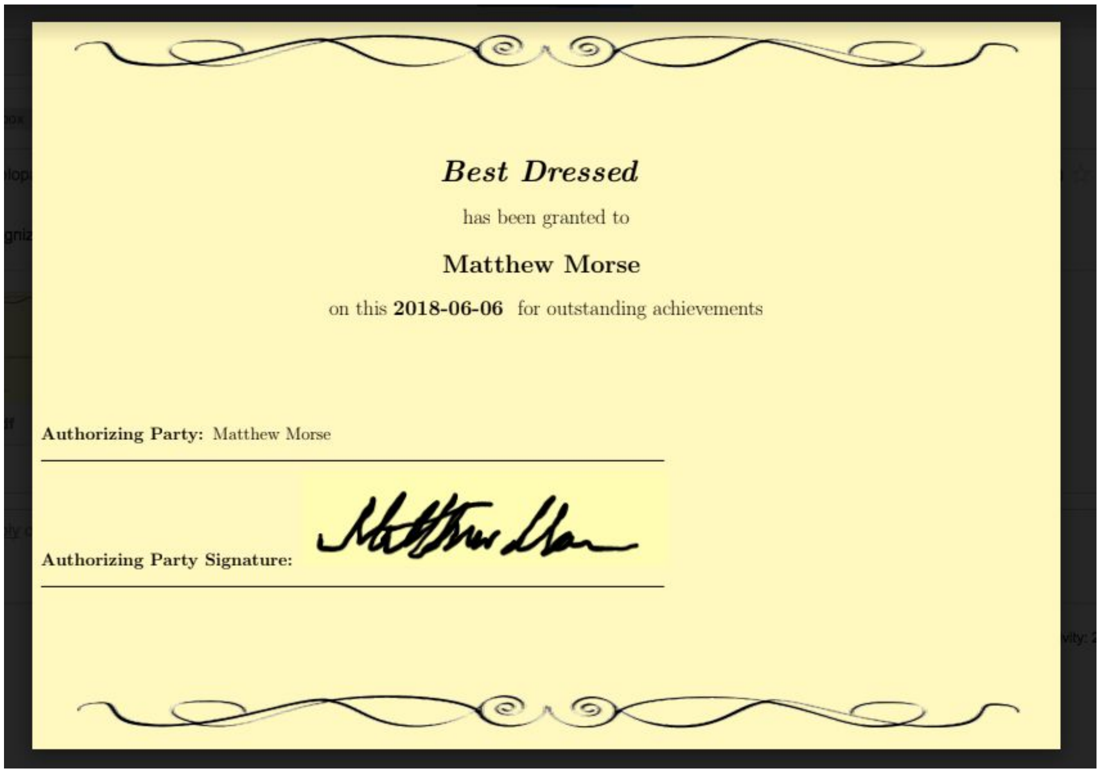

Congratulations Inbox x

Award Hub <camelopardalis.awardhub@gmail.com>
to me
7:22 AM (1 hour ago)

You are being recognized for your outstanding achievement! Please accept the attached award certificate as a token of gratitude.

certificate.pdf

***Figure 9.*** *Award recipient email example*


*Best Dressed*

has been granted to

Matthew Morse

on this **2018-06-06** for outstanding achievements

**Authorizing Party:** Matthew Morse

**Authorizing Party Signature:** [illegible]

***Figure 10.*** *Digital certificate example (awesome mouse-made signature included)*

## Updating Your Account Information as a User

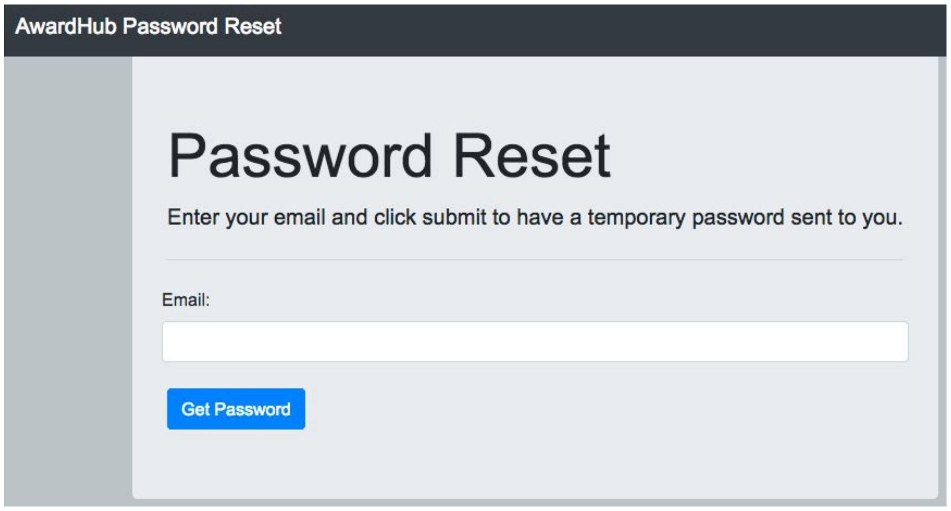
If while exploring the user-side of your site, you get the urge to change your name, password, or signature, you can do all of these things on the “my account” page which is accessible from the sidebar link.

## Delete and Modify Existing Recipient Records

Not satisfied with manipulating your own account information? Well, you're in luck! Award creators can also change the information of previous award recipients by going to the "manage recipients" page. The tools here should look very similar to the ones found in the administrative side of the site, but it should be noted that more form input types are available here for editing and validating. Like the admin-side, the form validation feedback is styled and processed using Bootstrap 4 and custom jQuery scripting. Be aware, however, that if you choose to delete a recipient, any awards they have received will also disappear.

## Regaining Account Access After Forgetting Password

Once you've had your fill of creating awards and selective amnesia sets in, you can test out our account recovery feature by logging out of the site. After being returned to the login page, select the hyperlink in the "forgot password?" text at the bottom of the login form. This will take you to a new page asking for your account email address. If the email address exists in the records, you will receive an email at the given address with a temporary password allowing you to log back in.


AwardHub Password Reset

Password Reset

Enter your email and click submit to have a temporary password sent to you.

Email:

***Figure 11.*** *Password reset page*

## Wrapping Up

Now that you've had a chance to recognize the hard work and determination of your favorite employees, you're welcome to check out the data you've created by logging back into the admin-side of the site and going to the Business Insights pages. Or just call it a day. Giving awards can be tiring. We hope our system made the process a little easier for you.

## Software Architecture

This section discusses how our software and systems function together. To adequately convey the relationship between these two components, we will begin by providing an overview of our system architecture. The importance of each systems piece, in addition to its resulting effect on our software, will be highlighted as we step through the architectural model.

Our group engaged Amazon Web Services for our hardware and created a new EC2 instance. We chose to adopt Ubuntu Server 14.04 as the server OS. Ubuntu made it possible for our group to manage our file system, save/modify our source code, install dependencies/resources, and configure necessary user permissions for our software to function. In essence, Ubuntu was the backbone that allowed all software (including ours) to run on the AWS machine. Amazon assigned our EC2 instance a public DNS in addition to a public IP. This was utilized to connect to our server via SSH, but eventually enabled access to our completed website via HTTP as well.

With the physical hardware and OS configured, our group turned to the procurement of a web server. Our selection was Apache2. The web server was responsible for enabling the software we built to be hosted, thereby making it accessible by visiting our public IP in a browser. Though web servers are capable of hosting files, it was necessary for us to select a server-side scripting language that would act as a foundation on which to build our software. We determined PHP5 was suitable for the scope of the project.

A database was also a chief component of our systems architecture. We leveraged MySQL to fulfill this piece. The MySQL database allowed our software to access persistent data and use it in a functional way. PHPMyAdmin was also a component of our architecture, and the interface was used by our team to build the database. It was instrumental in testing queries throughout the development process.

A handful of resources were necessary to support our actual software, and a full list of these can be found in the section below. We will briefly cover the most pertinent components of the software infrastructure here. TeX Live Software was installed on our server. It enabled us to compile a latex document on our EC2 instance. The SignaturePad Library was also installed directly to our server. It offered methods which allowed a user to draw a signature in the browser. The Google Charts service allowed us to visualize and display data. Finally, the PHPMailer library enabled us to set up an email service within our software. The following image is a graphical representation of our systems architecture.

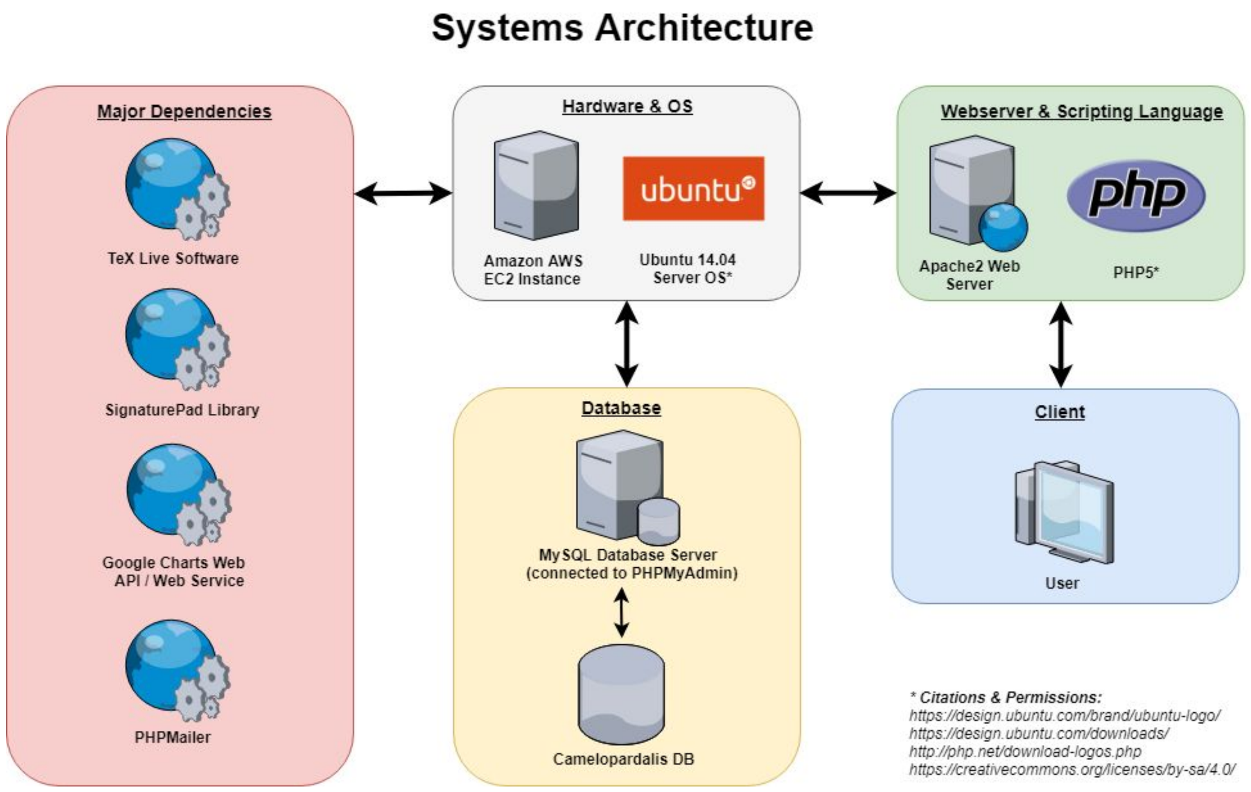

Systems Architecture
Major Dependencies
TeX Live Software
SignaturePad Library
Google Charts Web API / Web Service
PHPMailer
Hardware & OS
ubuntu
Amazon AWS EC2 Instance
Ubuntu 14.04 Server OS*
Webserver & Scripting Language
php
Apache2 Web Server
PHP5*
Database
MySQL Database Server (connected to PHPMyAdmin)
Camelopardalis DB
Client
User
* Citations & Permissions:
https://design.ubuntu.com/brand/ubuntu-logo/
https://design.ubuntu.com/downloads/
http://php.net/download-logos.php
https://creativecommons.org/licenses/by-sa/4.0/

***Figure 12.*** *Systems architecture*

It was briefly touched on above, but our software would not be able to function adequately without a well-designed database. Our MySQL server was hosted on our EC2 instance, and it was one of the most utilized components of the system by our software. We defined our database thoroughly in our project plan, and were able to implement it exactly as intended with only slight modification. A review of the database design as specified in our project plan is featured below, along with a copy of our database schema in *figure 13*.

We had a total of five tables in our database. The first was the *user* table. Though trivial to state, it contained all data elements specific to our users. It is good to note the client requirement called for two types of users, normal users and administrative users. Our *user* table contained an attribute for *account_type* that allowed us to identify each type accordingly. Fields that were not applicable to administrative users were set to “null” upon creation of record. The primary key for this table was an auto-incremented ID number. The *users* table had a one-to-many relationship with the *award* table. A user could create many awards, but each award could only have one user. We will review the *award* table next.

The *award* table contained all data points specific to the award itself. It had a foreign key to the *user* table to retain which user submitted the award. The *award* table had an attribute for *recipient_email*, which was joinable to the *recipient* table. Like the *users* table, the *recipient* table also had a one-to-many relationship with the *award* table. Each award could only have one recipient, but one recipient may have earned many awards. The *recipient* table was necessary to add to support the client's business intelligence requirements. Recipients may not always be users, and therefore not exist in our system. We needed stored records of recipients to generate data for them. The primary ID for this table is email address, as there could only be one email address per recipient.

To support the client's business intelligence requirements further we also designed two additional tables, *branch* and *manager*. Both had a one to many relationship with the *recipients*. Many recipients could work for the same branch. Likewise, many recipients could work for the same manager. That said, a recipient could only work for one branch and could only have one (direct) manager. This information, along with the attributes in the *recipients* table, was only collected in the event the recipient did not already exist. Provided the recipient existed, the client did not need to re-enter the information. The database schema is represented by *figure 13* below.

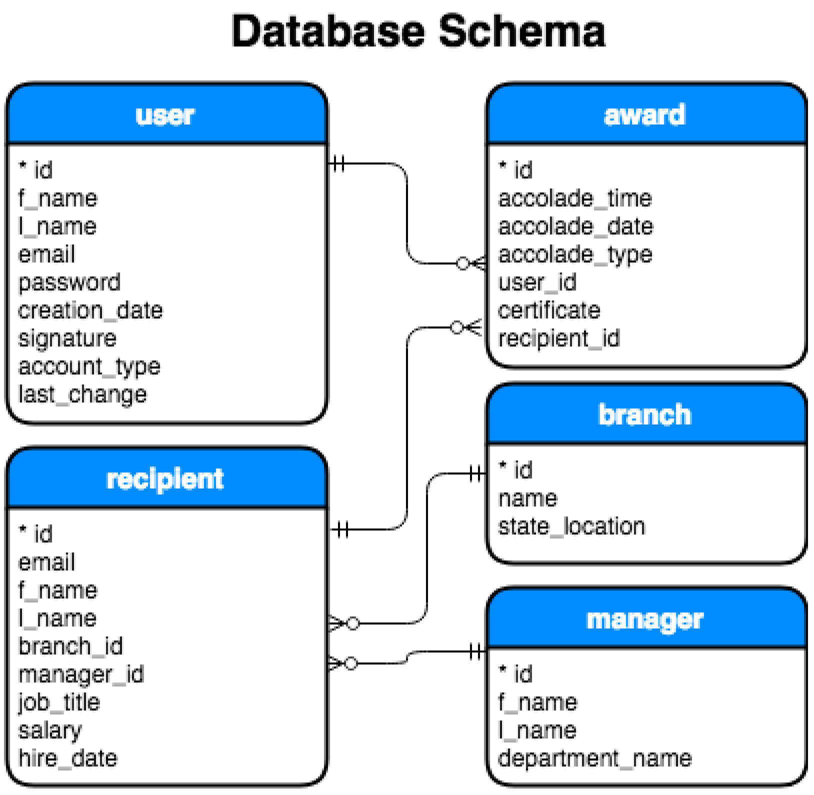


***Figure 13.*** *Database Schema*

## Resources

In the following list of resources, we discuss the tools and languages we used to the greatest extent. We also employed the knowledge and work of many other web developers during the course of this project. To ensure the proper citation of all borrowed code and resources, we have done our absolute best to credit their contributions where they applied by clearly documenting them in our source code.

## Version Control in Git and GitHub

Git and Github were used throughout the term for sharing code and providing much needed version backups. Tim used GitHub to share his excellent guides on accessing our server, using Git, and connecting to our database.

## PHPMyAdmin

This open-source tool was installed on our Amazon EC2 instance and used regularly to modify tables and check that our database was behaving correctly. It also helped when troubleshooting many of the issues that arose during development.

## Chrome and Firefox Developer Tools

Shannon and Tim did most of their development using Chrome. Matt did his on Firefox. Both browsers provided similar diagnostic features and they were used frequently to view the DOM and access error messages in the console. To a much lesser degree, built-in developer tools were utilized in Internet Explorer, Safari, and Edge to address issues in supporting those browsers.

## jQuery

The syntax and methods of this JavaScript library appear extensively in our source-code. The simplicity with which it accesses DOM elements, handles asynchronous requests, and binds event listeners was greatly appreciated while working on this project.

## PHP

When we began this term, only Shannon had any significant experience with this server-side language. Once everyone was up to speed, it proved to be valuable not only for communicating with our MySQL database but also for organizing our HTML into modular components that could be plugged into multiple pages on our website. PHP session variables were also used to handle account authorization and provide personalized form field pre-population.

## Google Charts Library

Our business intelligence features relied heavily on the Google Charts library. Multiple chart types were implemented along with tables. These were modified so that the were resizable and so that the data could be rearranged by changing the target table column’s ordering.

## Bootstrap 4 Framework

Though we did implement custom CSS mainly to provide additional responsiveness, the bulk of our styling was based on this framework. Bootstrap made it much easier for us to build a user-friendly, responsive layout. Bootstrap modals, tables, buttons, forms, cards, and navigation elements were employed extensively in our site's design. The documentation provided on Bootstrap's site was our main reference.

## SignaturePad Library

Our group leveraged the JavaScript SignaturePad library, available on GitHub by szimek. This library enabled us to instantiate a canvas-based drawing pad in the browser, which allowed a user to sign their signature with a mouse. The JavaScript SignaturePad library comes with a robust API, offering supportive functionality enabling us to customize the color of the signature pad in addition to saving the image in various formats.

## PHPMailer

PHPMailer, available on GitHub by PHPMailer, is a bona fide library that allows developers to easily send emails via PHP. Our group adopted this solution so our server could send emails containing new account information, password resets, and certificates. The library has an excellent API with the ability to customize subject lines, body content, recipients, sender aliases, and attachments.

## GitHub Octicons Library

Our application employed SVG files to improve the look and usability of our buttons and navigation menus. Using SVGs allowed us to dynamically resize and recolor button icons with CSS. The bulk of these image files were drawn from the GitHub Octicon library.

# Individual Accomplishments

## Timothy Fye

| *Task* | *Description* |
|---|---|
| **Server Configuration** | Created an Amazon AWS account, researched & configured EC2 instance (with Ubuntu Server 14.04, Apache2 webserver, security group supportive of HTTP/ HTTPS /SSH /TCP /UDP /PO3 /IMAP /SMTP, dependencies consisting of libapache2-mod-php5, mysql-server, php5-mysql, php5, |

| | |
|---|---|
| | phpMyAdmin, texlive-latex-base, phpmailer, signaturepad), created custom user groups so "Apache" (aka 'www-data'), "ubuntu", and "mysql" had appropriate directory permissions. |
| **Database Implementation** | Built group's db.sql file which executed a set of queries to construct and initialize database. Also created group's database resources consisting of a shared "config.db" file to instantiate a new connection so website could query database. Implemented landing pages that displayed the contents of database tables for group to use during testing. |
| **Latex/Certificate Logic** | Built a handler file with a function that accepted certificate parameters. Function created temporary directory, injected parameters into LaTex-formatted string, wrote latex formatted string to .tex file, extracted signature png from database and saved to temp directory, executed sub process to compile .tex document to pdf, pushed resulting certificate pdf to database while calling email logic to email certificate to recipient, cleaned up temp files and directory. |
| **Email Logic** | Built a handler file which accepted email parameters. Saved parameters in variable structure defined by PHPMailer, used PHPMailer to send an email to intended recipient with intended data points (Matt also made some contributions to this file that enabled it to be usable by multiple PHP scripts. Initially it was built solely to support certificate delivery). |
| **Signature Logic** | Inserted a form into Shannon's user-edit page. Form allowed user to either write a signature with mouse in browser or upload png. Leveraged SignaturePad library to provide a pad on which to draw signature with mouse. Developed php script that would check to see if signature pad was used or if document was uploaded. Contents of signature pad would be converted from URI encoded data to base64. Data would then be dumped into png file and the contents pushed to database. Contents of uploaded file would be extracted, code was built to take the base64 contents of file and resize it to a specified height/width. The contents of resized image would then be pushed to database. |
| **Login System** | Built a login system. User credentials were validated before allowing entry to site. Session was instantiated if user was found. Users are alerted if credentials are invalid. A session handler ensures an active session was instantiated before allowing a user to hit a page. If a session was not instantiated user is redirected to login page and asked to sign in. If a session was instantiated the site ensures admin type users is not trying to access a standard user page and vice versa. Matt/Shannon handled code surrounding password resets. |
| **Refactor Directory Structure** | Went through all files and updated paths to support a more organized directory structure. Debugged errors that resulted from the refactor. |

## Shannon Jeffers

| *Task* | *Description* |
|---|---|
| **General Information** | Coming into this project, Shannon was pretty much a complete novice with web development. Her only experience was CS290 and CS340 both of which basically hand-held and force-fed the information rather than being taught how to do the required tasks. A lot of learning about Bootstrap 4, CSS, HTML, and JQuery happened over the entire course of this project. |
| **Login Page** | Shannon handled the front end of the login page as well as the “forgot password” feature. The “forgot password” feature did, however, utilize the email functionality that Tim created. |
| **Navigation** | Most of the hard part of the navigation was already in place thanks to Matt. Shannon modified the existing admin navigation so that it worked for the user side. |
| **User Home Page** | Shannon was responsible for coding the cards and table on the user home page. They were done in the same styling as Matt’s home page and tables for website consistency. This styling heavily relied on Bootstrap 4. She also set up the back-end functionality to the delete button. |
| **Create Award** | This was broken up into two pages, both of which were handled by Shannon. At first she attempted to create a live search using some fancy JavaScript action, but that took hours and hours and ultimately ended in failure. She then created the simple search bar and back end functionality that is currently implemented. The search checks for the email in the recipient table of the DB and then reroutes accordingly. The create award form and back-end for submitting the form were also handled by Shannon. |
| **My Account** | Shannon was responsible for the styling of the my account page, as well as the edit name and password functionality. Tim wrote the code for editing signatures and Shannon popped it into a card on the my account page. This page mainly consists of Bootstrap cards and forms with a PHP back-end. |
| **Add Recipient** | Shannon created the form and layout for the add recipient page. The verification was a group effort in that Matt originally generated form |

| | |
|---|---|
| | verification code and Shannon figured out how to adapt it to work with the salary and date input. She was also responsible for the back-end of this page that handled the actual submission and page rerouting dependant on where the page was accessed from. This page was reworked after the mid-point check to make it more IE friendly. |
| **Edit Recipient** | Shannon was responsible for generating the table and setting up the edit and delete modals for this page as well as handling the back end functionality. Like with the add recipient page validation, this was a group effort. Shannon attempted to get it implemented and Matt had to helped to fix some issues with the validation functions. |
| **Testing and Debugging** | Shannon did testing of both sides of the site using Chrome and IE. She accidently stumbled upon bugs involving the login feature and form verification. |

## Matthew Morse

| *Task* | *Description* |
|---|---|
| **User and administrator account creation, modification, and deletion** | Matt developed two separate pages on the admin-side of the site for managing user and admin accounts. The pages follow the same layout with a Bootstrap-styled table listing users and admin records with responsive Bootstrap-style buttons linking to modals for adding, editing, and deleting user and admin accounts. User inputs are validated with custom jQuery functions and the data is submitted to a request handler on the back-end written in PHP which updates the MySQL database accordingly. Creating new accounts prompts an automated message to be sent to the new system user or administrator with a securely-generated random password. |
| **Award activity log with date range filter** | Matt implemented an activity log on the admin home page that lists awards given within a specified date range. The feature uses a combination of HTML5 form elements, Bootstrap styling of the table and buttons, jQuery request generation, and PHP request handling. |
| **Responsive sidebar and buttons** | Matt created a sidebar component for the admin-side pages using SVG images from GitHub's Octicon library. Custom CSS was written based on research regarding similar elements to make the sidebar adapt to different window sizes to improve usability across a range of devices. PHP was used to modularized the sidebar and add it to multiple pages using include statements. The current page button is highlighted dynamically using a snippet of jQuery. |

| | |
|---|---|
| **User-side add signature modal** | Matt constructed a Bootstrap modal that appears when users who don't have a signature on file try to access the award creation side of the system. PHP session variables and jQuery functions are used to check for a signature, set event listeners, and disable form submission until the user supplies a signature. The SignaturePad tools in the modal were adapted from code written by Tim. |
| **Business intelligence tools** | The Business Insights pages allow administrators to download CSV files for queried data, open charts in new tabs for easier printing and downloading, and alter chart data-ordering using connected tables. These features were developed using a combination of PHP, Bootstrap, Google Charts, jQuery, JavaScript, and HTML/CSS. |
| **Admin password update modal** | Matt adapted Shannon's jQuery and HTML code from the user-side of the site into a Bootstrap-styled modal that is available from any page on the admin site. Like the feature on the user-side, the updated password must meet a set of conditions before submission is enabled. |

## Deviations from Project Plan

Our team deviated from the initial strategy twice throughout the term. The first deviation was related to the systems architecture. We initially planned to leverage Oregon State University's servers as the medium to host our project. OSU offered the group a configured Apache webserver with PHP installed. Everything went smoothly until the end of week five, when we needed to implement functionality where the webserver would compile a LaTex document into PDF form. Our team attempted to accomplish this by instantiating a sub-process with PHP's *shell_exec()* function, passing a command line instruction to trigger the compilation engine. We found this was problematic on multiple levels. First and foremost, we were utilizing TeX Live as the software to compile our code. Though this software was installed on OSU's flip servers and the path environment was configured to allow the software to be run in any directory, it did not appear as though this engine was added to the path environment variable on the web server. Furthermore, it seemed the Apache user on OSU's server did not possess the appropriate execute permissions to support our code.

Though we could not verify these speculations on account of not being systems admins, we performed an inordinate number of tests to ensure the complications resided with the server configuration. During our debugging, we determined it was imperative we have control of server configurations. As such, we created an Amazon AWS account and instantiated a new EC2 instance. We then configured Apache, PHP, TeX Live, MySQL, amongst other dependencies discussed in the "Software Architecture" section above. Once we had control of our own server, we created a new user group with the "chown" command. This user group was comprised of the following users: "Apache" (aka 'www-data'), "ubuntu", and "mysql". We modified the new user group's permissions with the "chmod" command, delegating them rwx permissions as appropriate. Once this was accomplished, the original code we built that would not function on the OSU servers did work as designed on our EC2 server. This positive outcome reinforced our decision to stay with EC2 and led us to transition all our source codes from OSU to our AWS implementation.

The second deviation between the project plan and our final implementation regarded the Bureau of Labor Statistics API. We included this component in our project plan, however we did not have a clear vision on how we would utilize the resource. After research, we found that BLS did not offer data sets that we felt appropriately correlated to the data we were collecting from our site. Our group collectively decided to remove the BLS API objective from our systems paradigm, as we did not find a coherent way to insert this API into the flow our website. The outcome of this decision was favorable, as our user interface is not adversely affected by information that is not germane to the purpose of the system.

# Conclusion

At the beginning of this term, we planned to put over 300 hours of combined work into developing a web-based employee recognition system. Through all the unforeseen issues and amendments to our original plans, we certainly met that time goal. Though there may have been aspects of our first designs that never saw the light of day, we have plenty of things to be proud of that go well beyond the initial project specifications. For instance, our site users can forego the hassle of uploading image files by creating digital signatures with the tools available in our web pages. We also implemented a feature that saves users' time by allowing them to enter an employee's information once and retrieve it in the future just by typing the email address into a search bar. Furthermore, our website ended up being compatible with screen sizes ranging from mobile phones to desktop computers.

It is only because of countless others that we were able to successfully deliver this project. No doubt, creating this system was an exercise in communication. Not only did we have to work together effectively but we were challenged to make various tools and resources coexist constructively in our application. The majority of our site's pages were built with varying degrees of HTML, CSS, PHP, JavaScript, and jQuery along with many other software libraries, frameworks, and tutorial references.

Though we may have been humbled on many occasions, we have grown significantly as software developers as a result of working together on this project. We faced adversity and we adapted in order to meet our goals as a team. Like all engineers, we would always like to change one last thing or do more testing but we are happy with what we’ve accomplished. We hope you enjoyed reading our report and exploring our web-based employee recognition system.