Camelopardalis: Project Plan

GROUP MEMBERS

Timothy Fye
fyet@oregonstate.edu

Shannon Jeffers
jeffesha@oregonstate.edu

Matthew Morse
morsmatt@oregonstate.edu

Table of Contents

Introduction

Team Camelopardalis will be creating an employee recognition system using several web-based languages/technologies. This web application will consist of a set of user and administrative functions that are hidden behind a login page. The overall goal of the system is to serve as a tool that enables managers/people in charge of awards at an organization to quickly and easily generate an award certificate that can then be emailed to the employee receiving the award. The website will also provide business intelligence services to assist organizations in delivering effective employee recognition and merit-based wage adjustments. Our team is looking forward to improving our web development skills through this project.

User Perspective

From the perspective of the end user, our product will provide a very useful service for generating awards to be given to their employees. Gone will be the days of trying to line up all of the information perfectly in a Word document in order to print it on a piece of special paper for the award. Our project will handle all the nuances for the customer, allowing them to simply worry about who is getting what award. Our product will also prove useful for the business side of things because the analytics features will allow data to be gathered that could better assist employers with things like managing salaries and bonuses.

Software Structure

Our group's software structure has been segmented into five individual components. These components are Systems Architecture, Database Schema, Class Structure, Activity Model, and User Interface Model. Each of these segments are defined in their respective sub-sections below.

1. Systems Architecture

Our group began with designing our high-level architecture. We will use Oregon State University's instantiated Apache web server (software) hosted on the school's FLIP servers (hardware). Student directories on the school's system will store our team's source codes for the web server to process. PHP scripting language will be leveraged by our group to perform all necessary server-side actions. These actions will, in part, consist of employing web services made available by the Bureau of Labor Statistics, Javascript SignaturePad, and Google Charts. Finally, our group will create a MySQL database to support our web application. We will use PHP to enable our web application to interact with the repository. Our systems architecture is represented by *Figure 1* below.

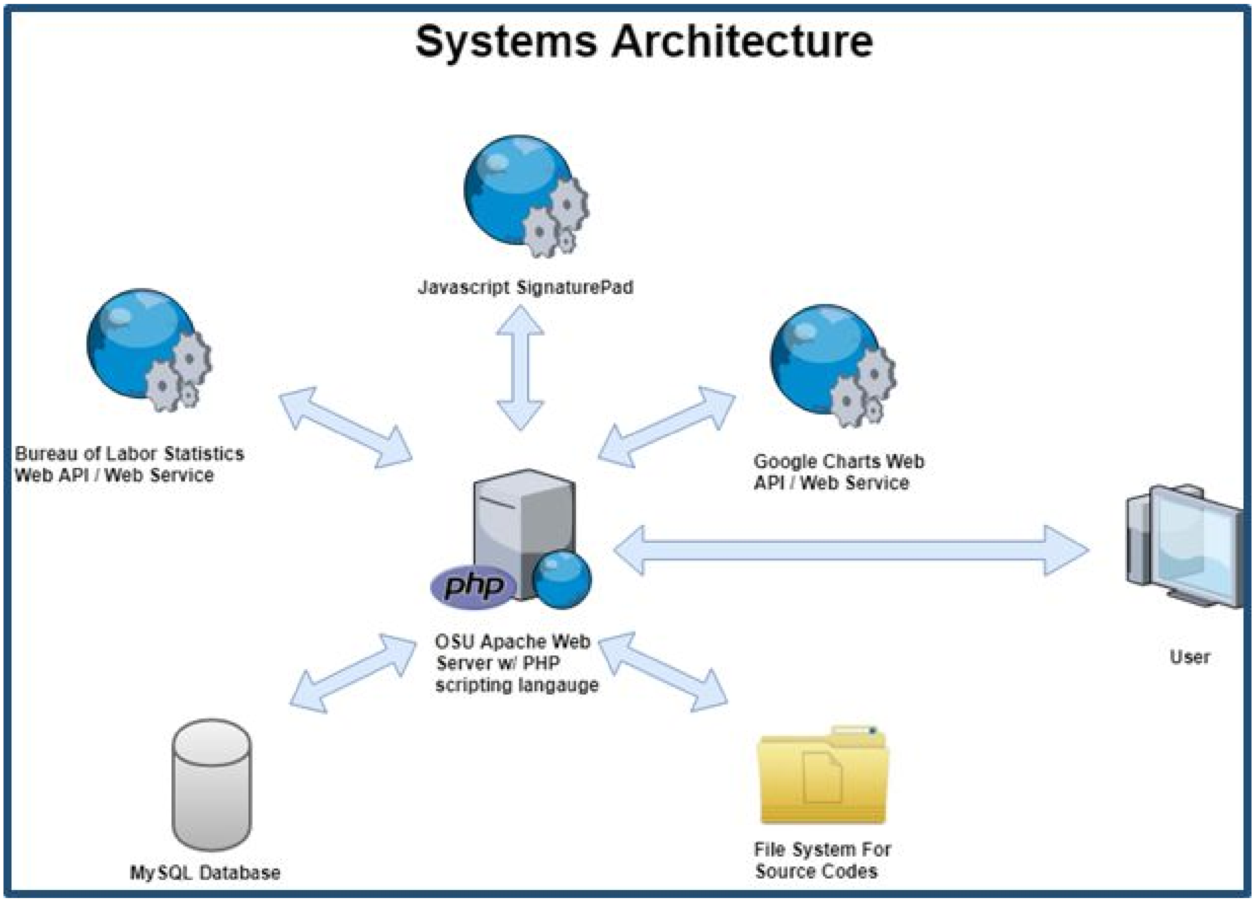

Systems Architecture
Javascript SignaturePad
Bureau of Labor Statistics Web API / Web Service
Google Charts Web API / Web Service
php
OSU Apache Web Server w/ PHP scripting langauge
User
MySQL Database
File System For Source Codes

Figure 1. *Systems architecture*

2. Database Schema

An integral aspect of any web application is database design. Our group modeled a database plan to support the functional requirements of the website. We will have a total of five tables in our database. The first is the *user* table. Though trivial to state, it will contain all data elements specific to our users. It is good to note the client requirement calls for two types of users, normal users and administrative users. Our *user* table contains an attribute for *account_type* that will allow us to identify each type accordingly. Fields that are not applicable to administrative users will be set to “null” upon creation of record. The primary key for this table is an auto-incremented ID number. A situation may exist where a person with the same email address has both an administrative account and a standard account. As such, our group determined the use of a numerical ID was preferable to using the email address as a primary key. The *users* table has a one-to-many relationship with the *award* table. A user can create many awards, but each award can only have one user. We will define the *award* table next.

The *award* table will contain all data points specific to the award itself. It will have a foreign key to the *user* table to retain which user submitted the award. The *award* table will have an attribute for *recipient_email*, which will be joinable to the *recipient* table. Like the *users* table, the *recipient* table will also have a one-to-many relationship with the *award* table. Each award can 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 must have stored records of recipients to generate data for them. The primary id for this table is email address, as there can 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 have a one to many relationship with the *recipients*. Many recipients can work for the same branch. Likewise, many recipients can work for the same manager. That said, a recipient can only work for one branch and can only have one (direct) manager. This information, along with the attributes in the *recipients* table, will only be collected in the event the recipient does not already exist. Provided the recipient exists, the client will not need to re-enter the information. The database schema is represented by *Figure 2* below.

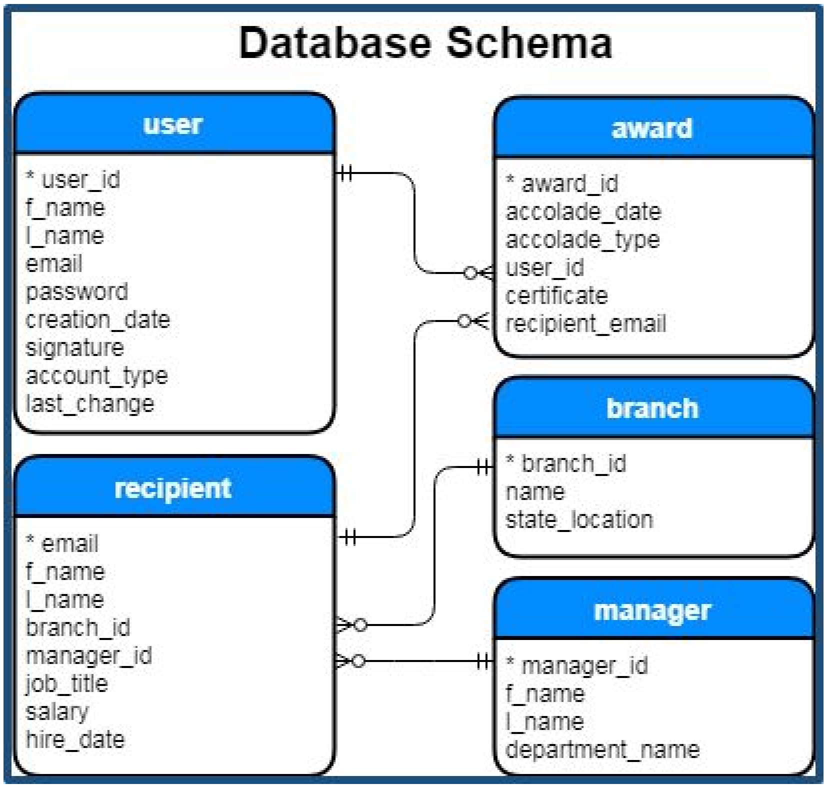


Figure 2. *Database schema*

3. Class Structure

The next stage of developing our software structure was defining classes. Our UML Class Diagram is represented in *Figure 3* below. There are three notable features to point out in the class diagram. The first is inheritance. The *user* class' attributes and methods are inherited by the *normalUser* and *adminUser* classes. This is designated by white arrow in *Figure 3*. The diagram shows the *normalUser* has its own attributes outside of those inherited by the *user* class. This is because the client has defined data elements that are only pertinent to standard users (i.e name, signature). The client has also defined actions that are to be unique to each type of user. As such, both *normalUser* and *adminUser* have extra methods that are not inherited from the parent class as well. These methods allow them to functionally perform the actions that are applicable to their respective sub-classes.

The second notable feature is relationships. The *normalUser* class has a composite relationship with the *Award* class (designated by the black diamond in *Figure 3*). This means that an award cannot exist without the *normalUser* that created it. These two classes also have a one to many relationship. A user can create many awards, but each award can only be tied to one user. Similarly, the *award* class has composite relationship with *recipientData* class. The recipient data cannot exist without an award. These two classes have a one-to-one relationship. Moving along, the *adminUser* class has an association with *businessIntelligence* class since *adminUsers* can invoke the *accessBusinessIntelligence()* method. Lastly, the *businessIntelligence* class has an association with the *recipientData* class and the *user* class since it will use its data.

The third notable feature in the UML diagram is scope. Each class has attributes that are private (designated by the “-” in front of the attribute or method). These data points or actions are only used by the class they reside in. Public methods (designated by the “+”) are methods that can be invoked outside of the class. The two public options our group has created is the *getUserData()* and the *getRecipientData()* methods. Even though the *businessIntelligence* class cannot directly access or manipulate the data within either of its associated classes, it can still retrieve the data through the public methods.

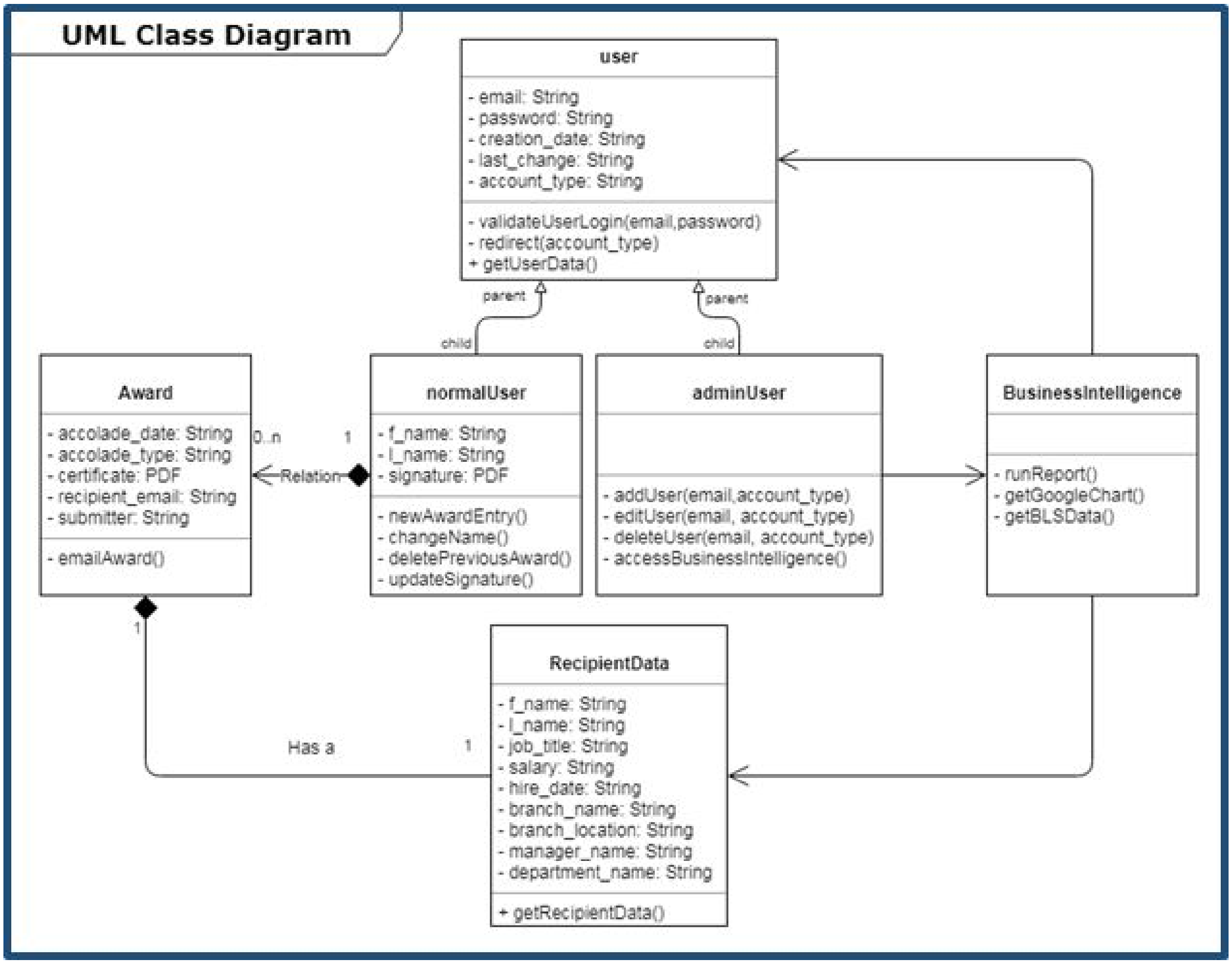


Figure 3. *UML class diagram*

4. Operation/Activity Structure

The next important aspect of our group's proposed software structure is the flow of activity. We created a visualization in *Figure 4* below to represent this model. The visualization is fairly comprehensive so little explanation is necessary. Effectively each edge leads to an action, the flow is represented by the direction of arrows, and the flow branches when a decision condition (designated by a diamond) is met. We have two major branches, one for the activities that result when a standard user logs in and another that results from when an administrator logs in.

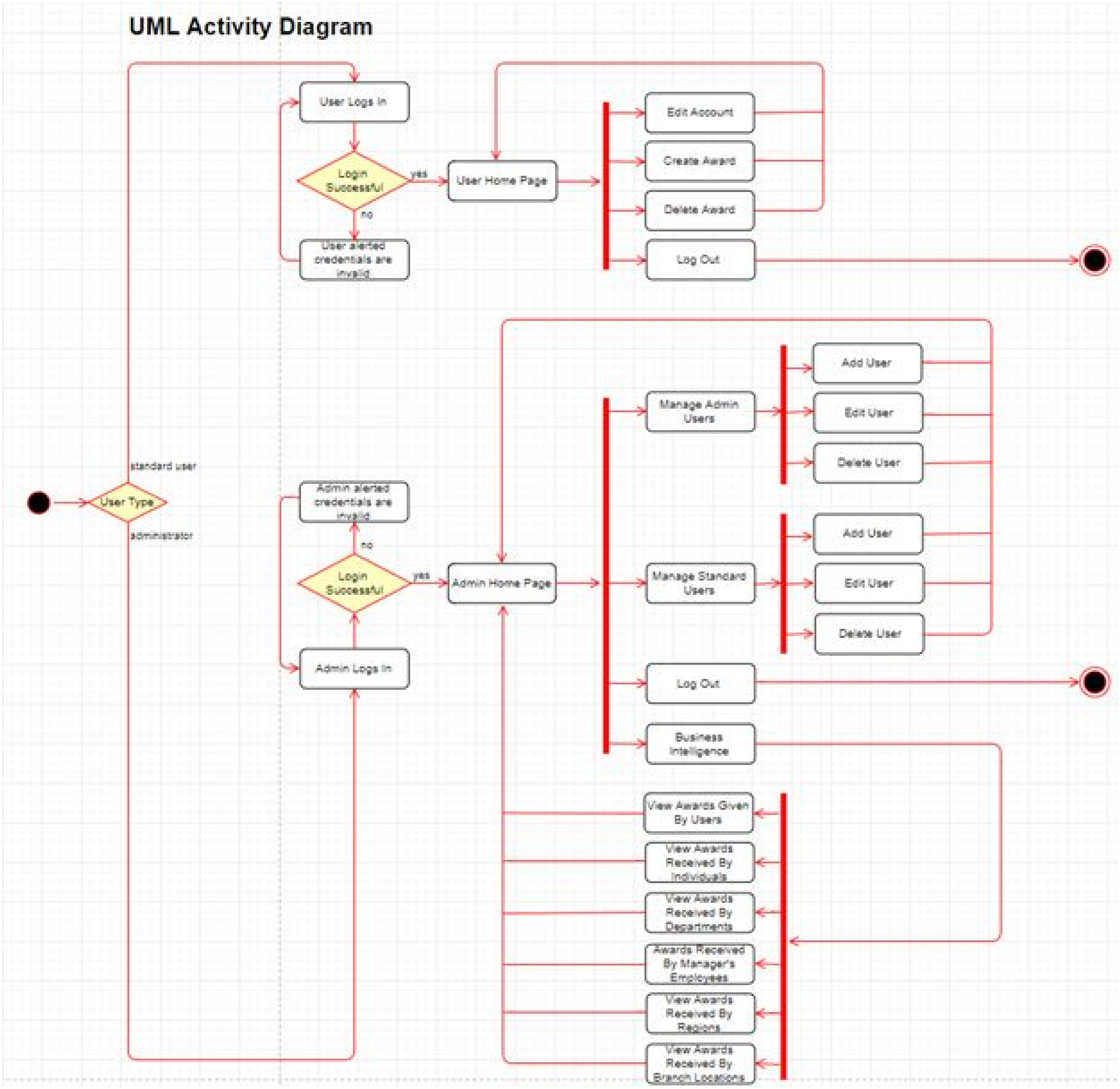


Figure 4. *UML Activity Diagram*

5. User Interface Model

The last software structure component our group designed was the user interface model. This consists of the layout of the webpages themselves. We have created a comprehensive set of high-fidelity prototypes to represent our UI Model. These can be reviewed in the “Addendum” section at the end of this report.

Software Requirements

Languages

- Database development using MySQL
- Server and database communication in PHP
- Client-side scripting in Javascript and jQuery
- Client-side page structure and styling in HTML/CSS

Supported web browsers:

- Chrome (website should be fully-functional)
- Firefox (website should be fully-functional)
- Safari (website should be fully-functional)
- Microsoft Edge (website should be fully-functional)
- Internet Explorer 10+ (website should be usable)

APIs:

- US Bureau of Labor Statistics

Libraries

- Google Charts
- Javascript SignaturePad library
- jQuery

Version Control:

- Git

Host:

- OSU Engineering Server

Team Member Tasks

Timothy Fye

Week	Task Description	Est. Time (hrs)
3	● Set up database ● Configure files and landing pages to view DB contents for testing	15
4	● Research and implement LaTeX PDF generation & storage in DataBase	14
5	● Research and implement backend email service of PDF award certificates	12

6	● Research and implement: ○ login controls ○ possible session instantiation/retention ○ account-type recognition ○ redirection/viewing of applicable pages	17
7	● Continue research and implementation of: ○ login controls ○ possible session instantiation ○ account-type recognition ○ redirection/viewing of applicable pages	17
8	● Assist Matt as needed with: ○ B.I. features ○ configuring API to BLS ○ Incorporating Google Charts into site ○ data rendering to front-end ○ Testing business intelligence features	15
9	● Week to get caught up with any residual duties left over from previous weeks ● assist either Matt or Shannon with remaining front-end page configuration	15
10	● Final testing ● Final report ● Demo file creation	10
Total hours		**115**

Shannon Jeffers

Week	**Task Description**	**Est. Time (hrs)**
3	● Implement the login front-end for users/admins ● Implement the page for award creation ○ this includes a method for indicating which type of award is being issued	17
4	● Implement the user home page and include: ○ links to all other user pages ○ link to log out ○ querying for all awards created by user	16
5	● Implement the modify user account information page ● Implement the navigation bar for users that will appear on all	15

	user pages and allow easy navigation throughout the user site	
6	● Research ways to use account information throughout the user account pages without having the user manually enter any information ● Implement a signature pad which will allow users to more easily update their signature	12
7	● Implement the back-end for the award creation page ○ include a way to detect which user is responsible for the submission without manual entry	15
8	● Test user pages for bugs/ fix any bugs that arise ● Assist Tim and Matt with any testing they need assistance with	12
9	● Catch up on any unfinished tasks ● Help Matt and Tim with any of their remaining tasks ● Refactor code if necessary.	15
10	● Final testing ● Final report ● Demo file creation	10
	Total hours	**112**

Matthew Morse

Week	Task Description	Est. Time (hrs)
3	● Implement create a new administrator feature for admin pages ● Implement “create a new user” feature for admin pages ● Implement menu for admin-side of web application	17
4	● Research Bureau of Labor Statistics API ● Research Google Charts API ● Work on admin-side homepage ○ Implement “Activity Log” feature ○ Create link cards to site pages ○ Logout button	16
5	● Begin implementation of “Business Intelligence Section” ○ Create main page with links to report generation pages	15

	○ Create “award granter” report section ■ Make graph generator using Google Charts API ■ Make query results table generator ■ Add filter tool ■ Add option to download report as CSV file ■ Add option to download graph as PNG file	
6	● Continue implementation of “Business Intelligence” section ○ Create “award recipient” report section ■ Add graph generator using Google Charts API ■ Add graph that compare average salaries versus award winner salaries using BLS API data ■ Add query results table generator ■ Add filter tool ■ Add option to download report as CSV file ■ Add option to download graph as PNG file ○ Create “award by region” report section ■ Add graph generator using Google Charts API ■ Add query results table generator ■ Add filter tool ■ Add option to download report as CSV file ■ Add option to download graph as PNG file ○ Create “awards by branch” report section ■ Add graph generator using Google Charts API ■ Add query results table generator ■ Add filter tool ■ Add option to download report as CSV file ■ Add option to download graph as PNG file	14
7	● Finish implementation of “Business Intelligence” section ○ Create “awards by manager” report section ■ Add graph generator using Google Charts API ■ Add query results table generator ■ Add filter tool ■ Add option to download report as CSV file ■ Add option to download graph as PNG file ○ Create “awards by department” report section ■ Add graph generator using Google Charts API ■ Add query results table generator ■ Add filter tool ■ Add option to download report as CSV file ■ Add option to download graph as PNG file ● Begin testing admin pages	16
8	● Continue testing admin pages	13

	● Apply bug fixes ● Refactor code as necessary	
9	● Review code written by team members ● Refactor code as necessary ● Catch up on any unfinished tasks	13
10	● Final testing ● Final report ● Demo file creation	13
Total hours		**117**

Documentation Delegation

Assignment	Due	Format	Assignee to Submit
Project Plan	April 15	PDF	Everyone
Week 4 Progress Report	April 23	Web-hosted video	Everyone
Week 5 Progress Report	April 30	Web-hosted video	Everyone
Week 6 Progress Report	May 7	Web-hosted video	Everyone
Mid-Point Project Check	May 14	.zip file including: ● Instructions PDF ● Source code files ● URL of project site	Shannon
Week 8 Progress Report	May 21	Web-hosted video	Everyone
Week 9 Progress Report	May 28	Web-hosted video	Everyone
Create Final Report	June 8	PDF	Tim
Demonstrate Project	June 8	.zip file including: ● Instructions PDF ● Source code files ● URL of project site	Matt

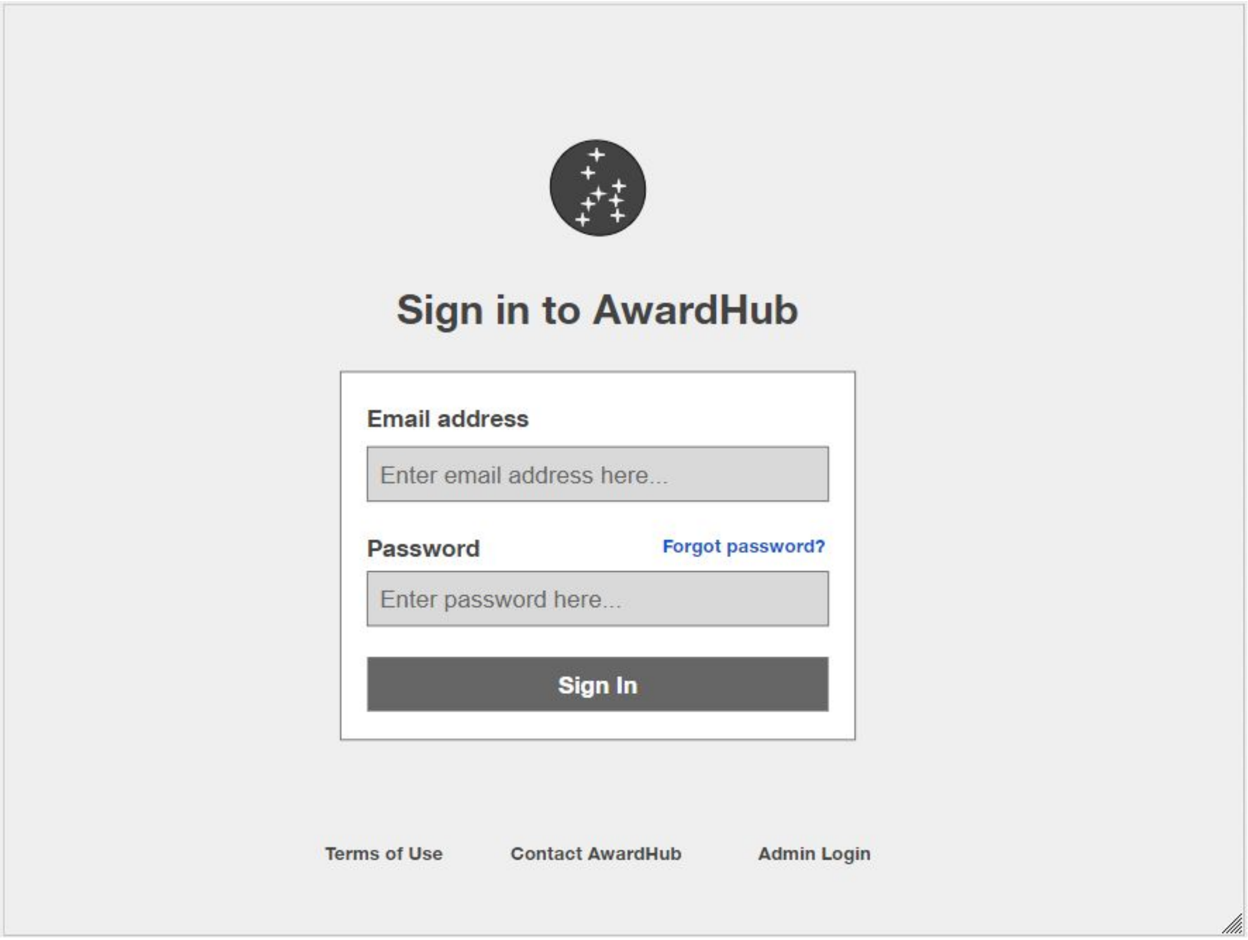
Conclusion

As shown by the preceding sections of this project plan, the Camelopardalis team hopes to create a web-based employee recognition system that enables specific users to grant digital award certificates to other individuals. This application will also support administrative capabilities for user management along with tracking and analysis of business data using a variety of metrics related to the granting of awards. Our primary development languages will be HTML/CSS, Javascript, and PHP. By our estimates, this project will take over 300 hours to complete and should be ready within the agreed upon timeframe.

Addendum

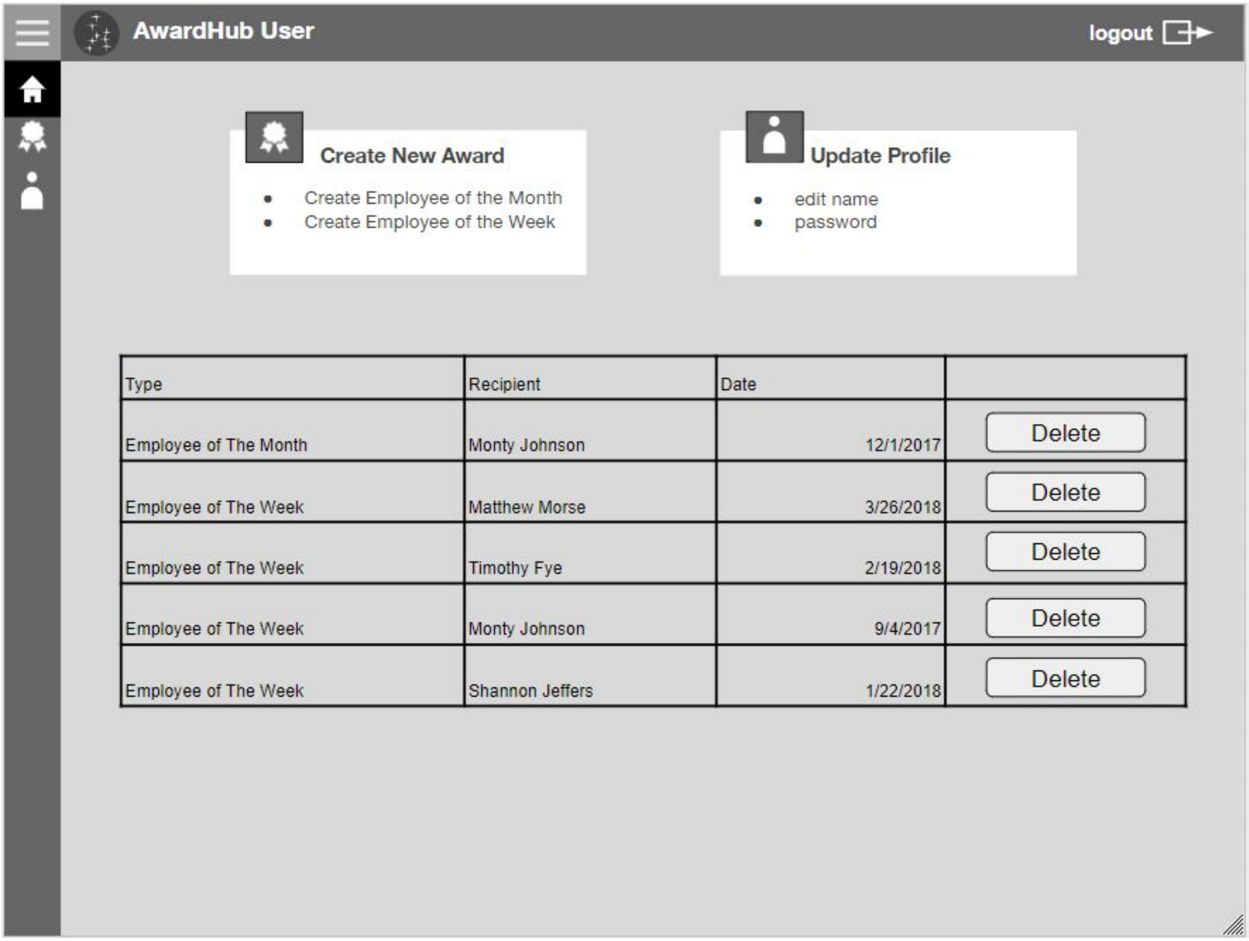
User-side Interface Prototypes

I. User Login



Sign in to AwardHub

II. User Home Page

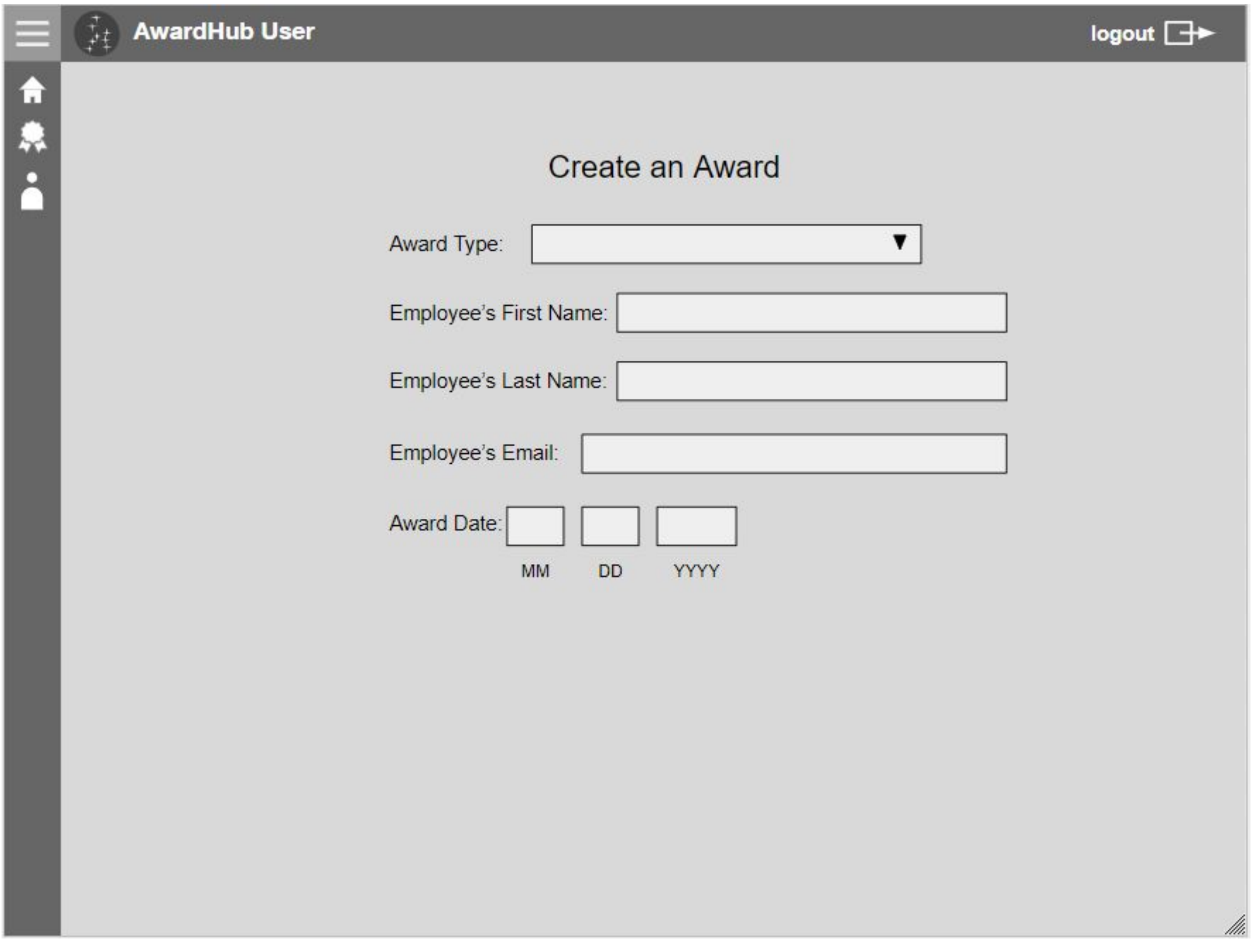


- Create Employee of the Month
- Create Employee of the Week

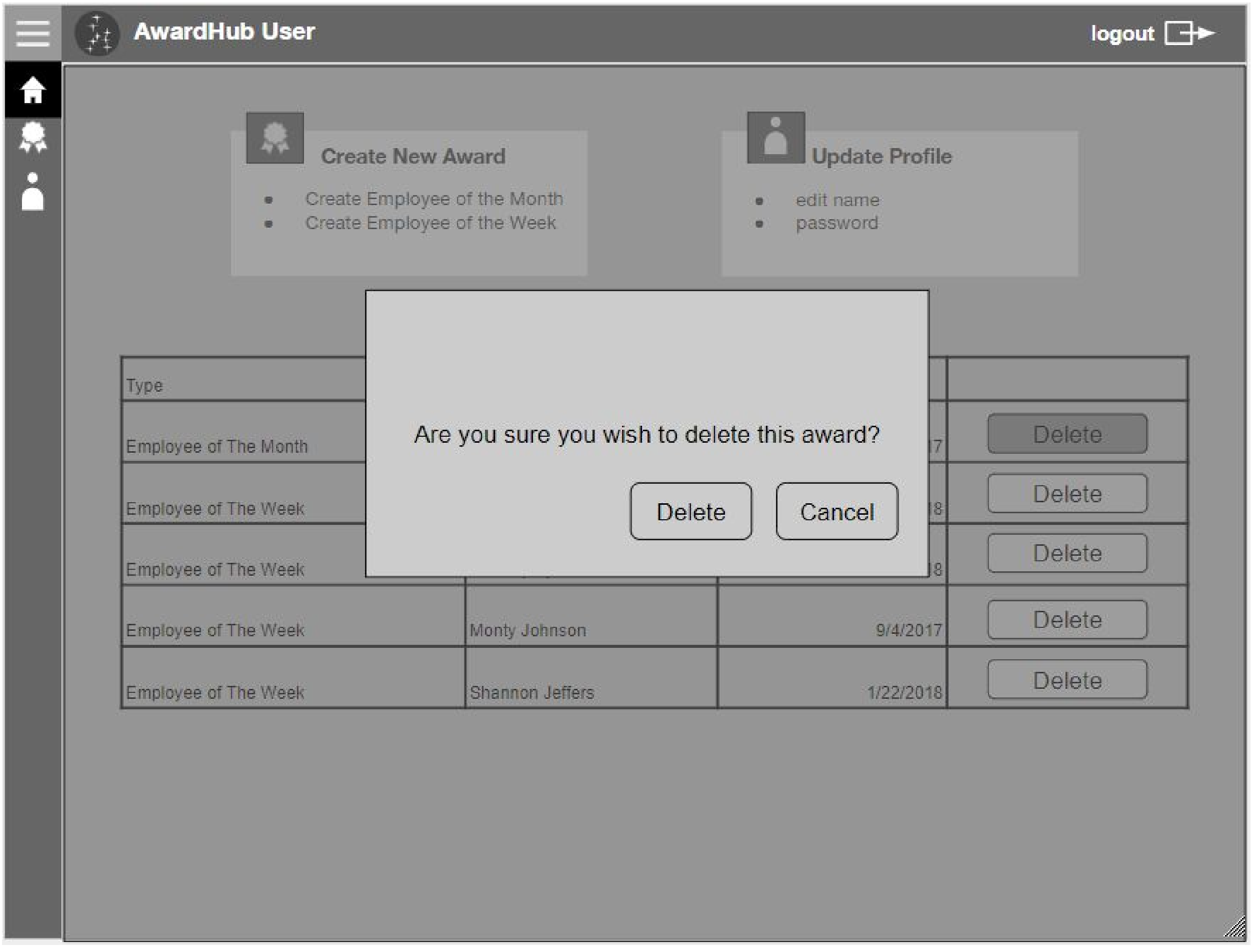
- edit name
- password

Type	Recipient	Date	
Employee of The Month	Monty Johnson	12/1/2017	Delete
Employee of The Week	Matthew Morse	3/26/2018	Delete
Employee of The Week	Timothy Fye	2/19/2018	Delete
Employee of The Week	Monty Johnson	9/4/2017	Delete
Employee of The Week	Shannon Jeffers	1/22/2018	Delete

III. User Create Award



IV. User Delete Award



Create New Award

- Create Employee of the Month
- Create Employee of the Week

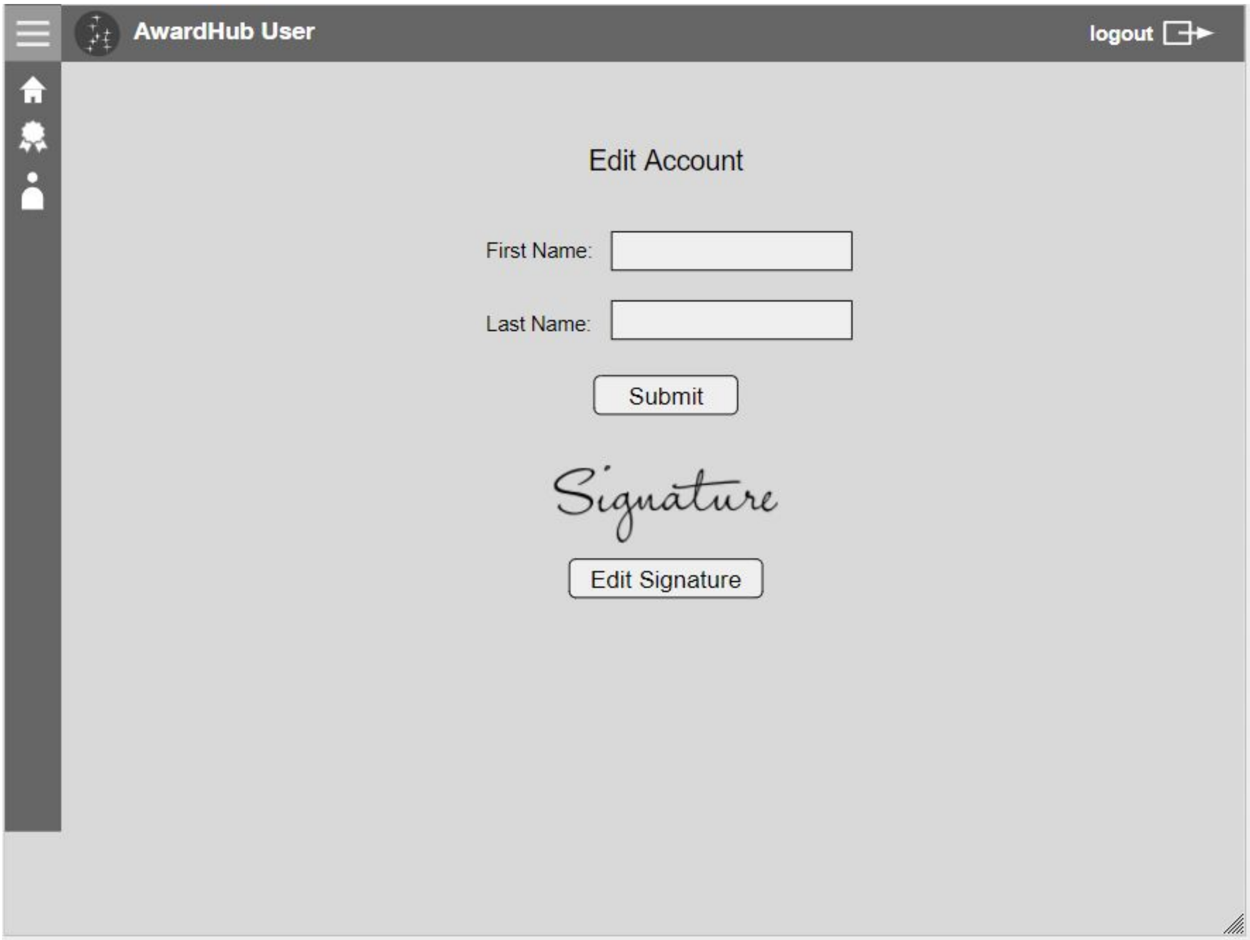
Update Profile

- edit name
- password

Type			
Employee of The Month		7	Delete
Employee of The Week		8	Delete
Employee of The Week		8	Delete
Employee of The Week	Monty Johnson	9/4/2017	Delete
Employee of The Week	Shannon Jeffers	1/22/2018	Delete

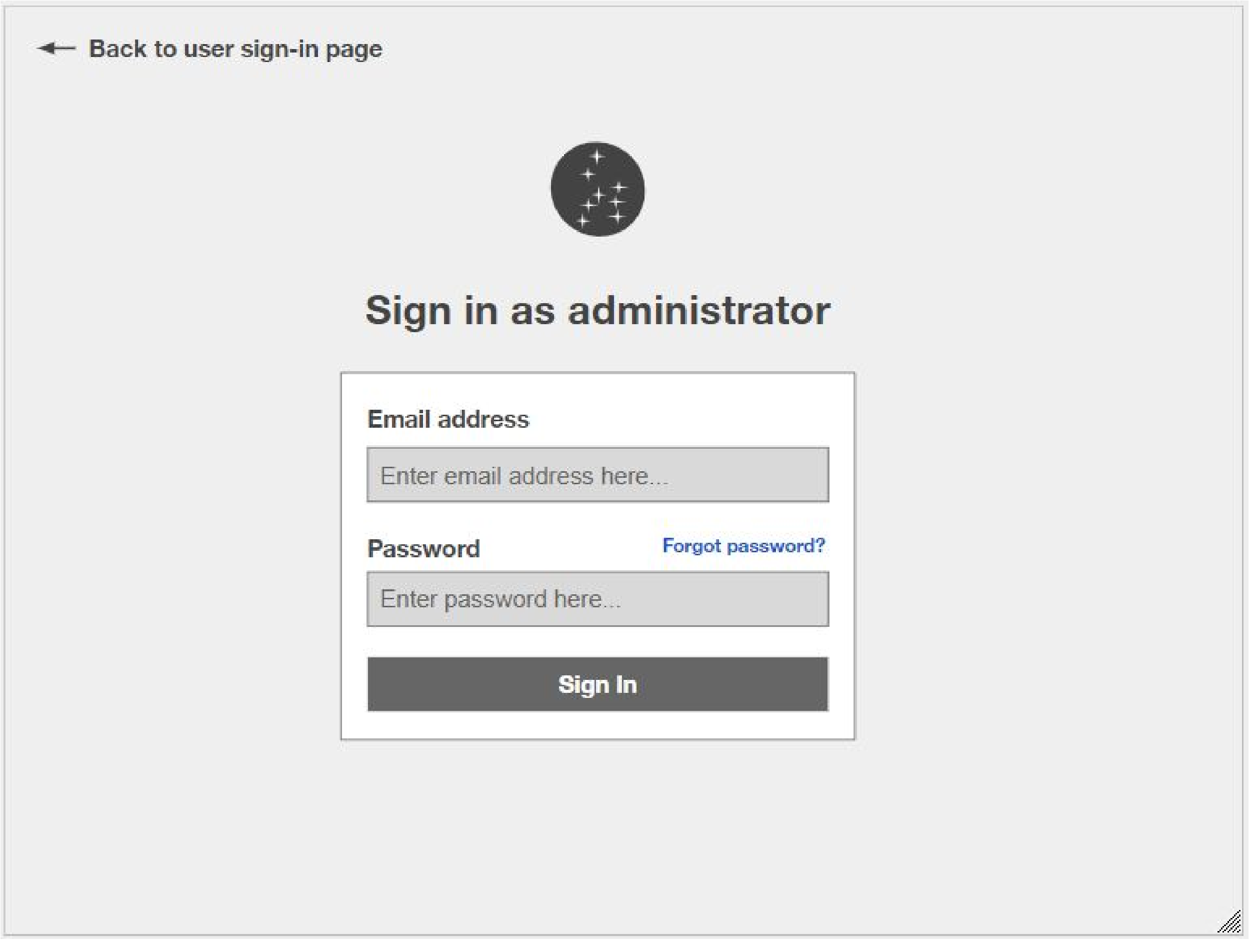
Are you sure you wish to delete this award?

V. User Edit Account

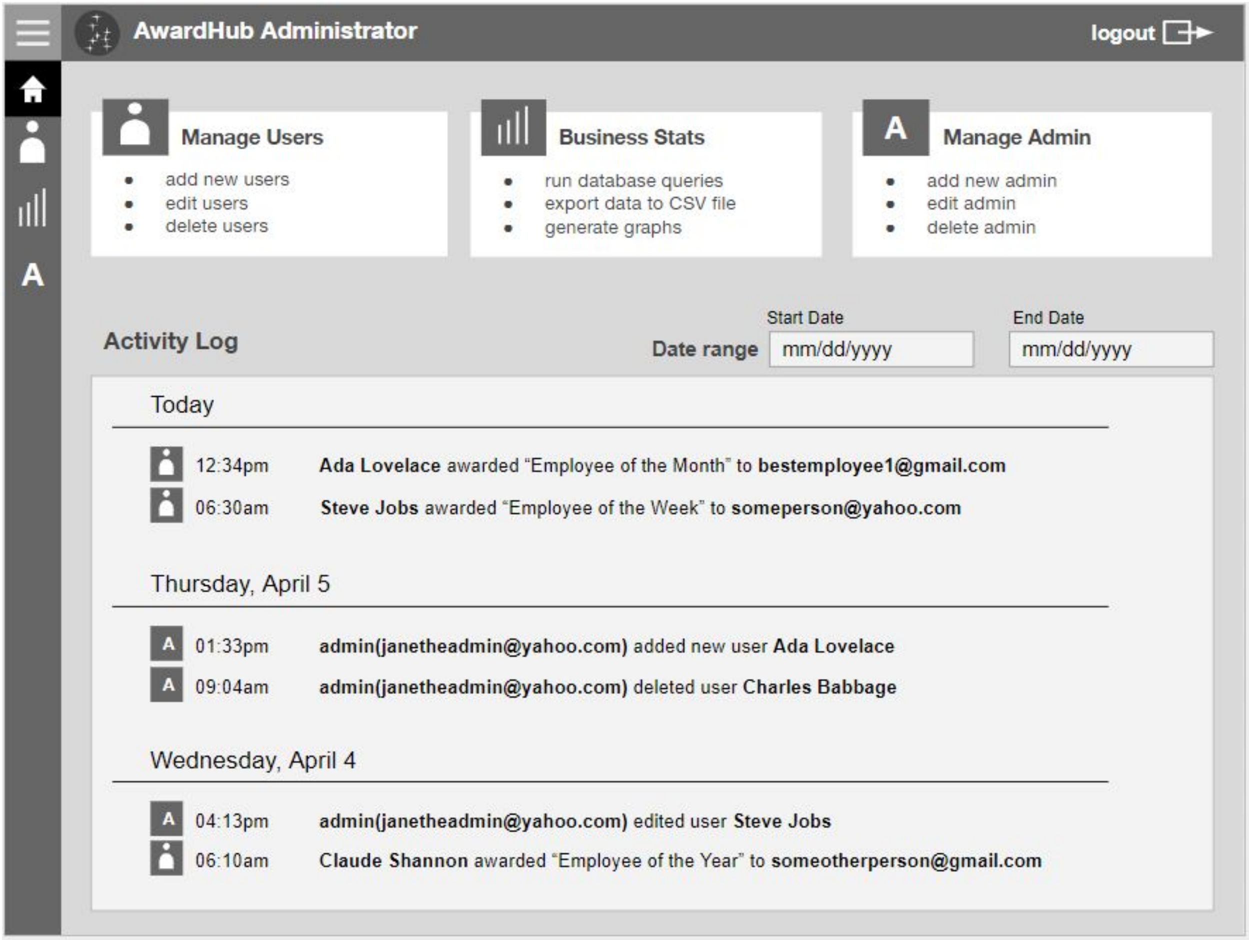


Administrator-side Interface Prototypes

I. Admin Login



II. Admin Home Page



Manage Users

- add new users
- edit users
- delete users

Business Stats

- run database queries
- export data to CSV file
- generate graphs

Manage Admin

- add new admin
- edit admin
- delete admin

Activity Log

Date range Start Date mm/dd/yyyy End Date mm/dd/yyyy

Today

12:34pm **Ada Lovelace** awarded "Employee of the Month" to **bestemployee1@gmail.com**

06:30am **Steve Jobs** awarded "Employee of the Week" to **someperson@yahoo.com**

Thursday, April 5

A 01:33pm **admin(janetheadmin@yahoo.com)** added new user **Ada Lovelace**

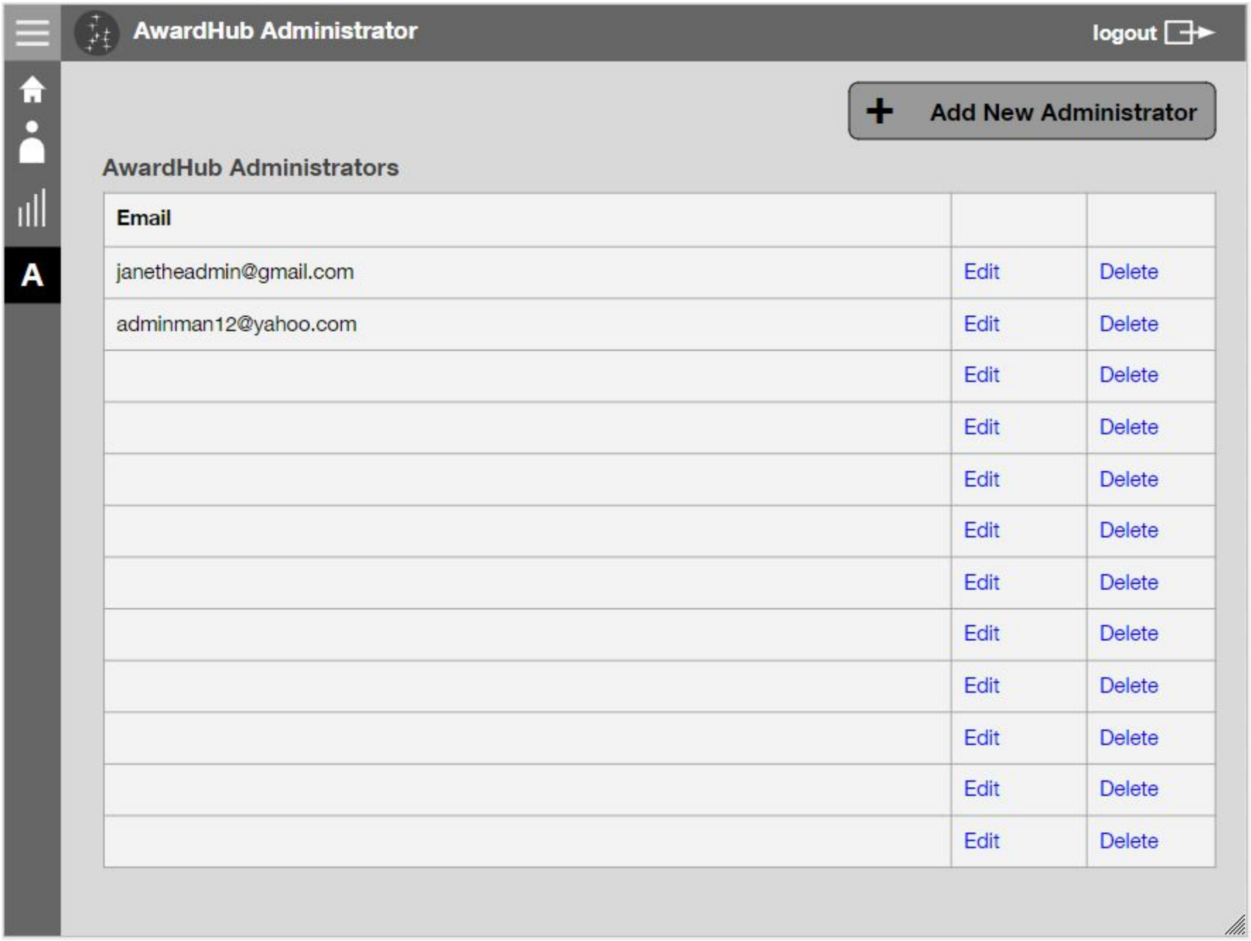
A 09:04am **admin(janetheadmin@yahoo.com)** deleted user **Charles Babbage**

Wednesday, April 4

A 04:13pm **admin(janetheadmin@yahoo.com)** edited user **Steve Jobs**

06:10am **Claude Shannon** awarded "Employee of the Year" to **someotherperson@gmail.com**

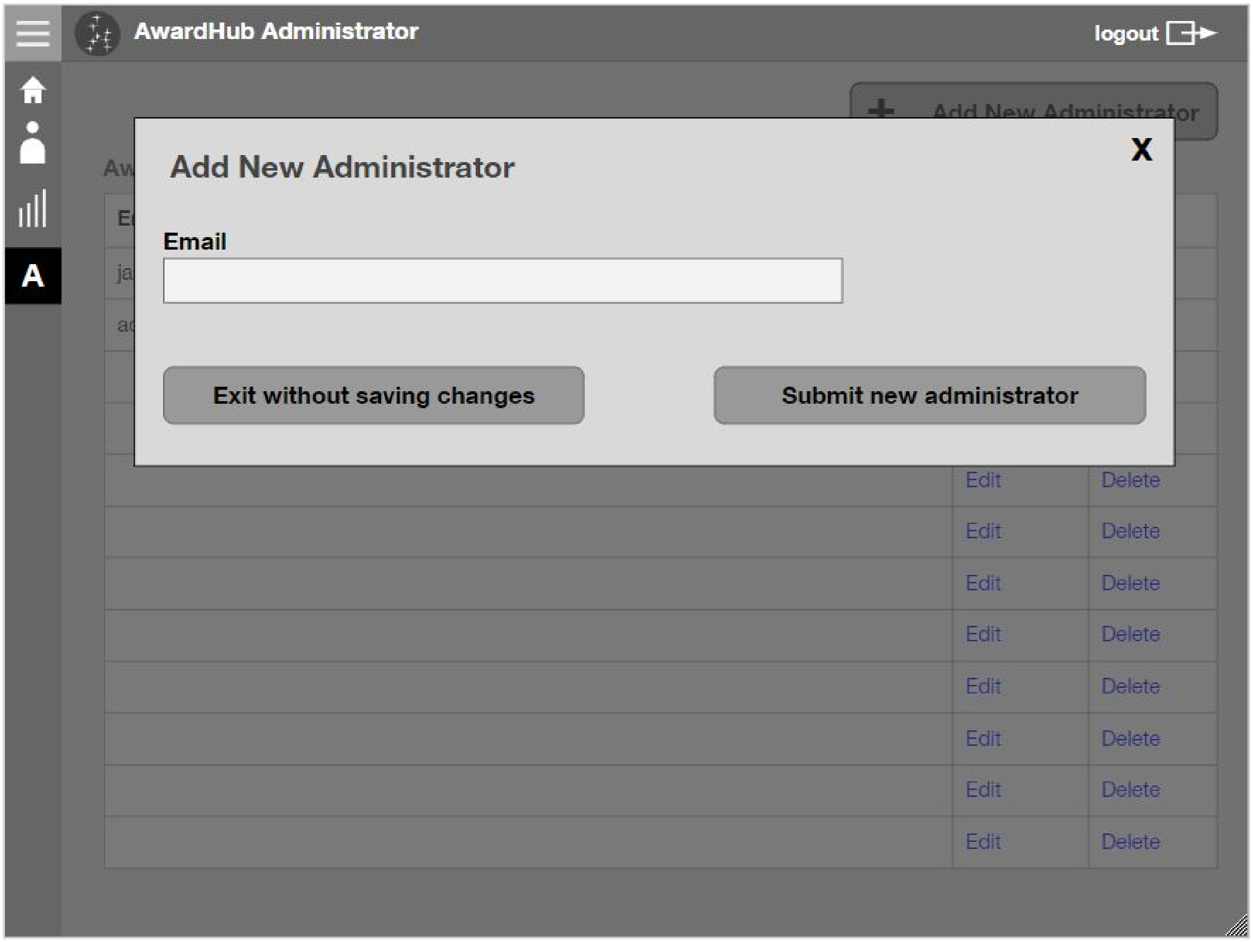
III. Manage Admin - Home



AwardHub Administrators

Email		
janetheadmin@gmail.com	Edit	Delete
adminman12@yahoo.com	Edit	Delete
	Edit	Delete
	Edit	Delete
	Edit	Delete
	Edit	Delete
	Edit	Delete
	Edit	Delete
	Edit	Delete
	Edit	Delete
	Edit	Delete
	Edit	Delete

IV. Manage Admin - Add New Admin



Add New Administrator

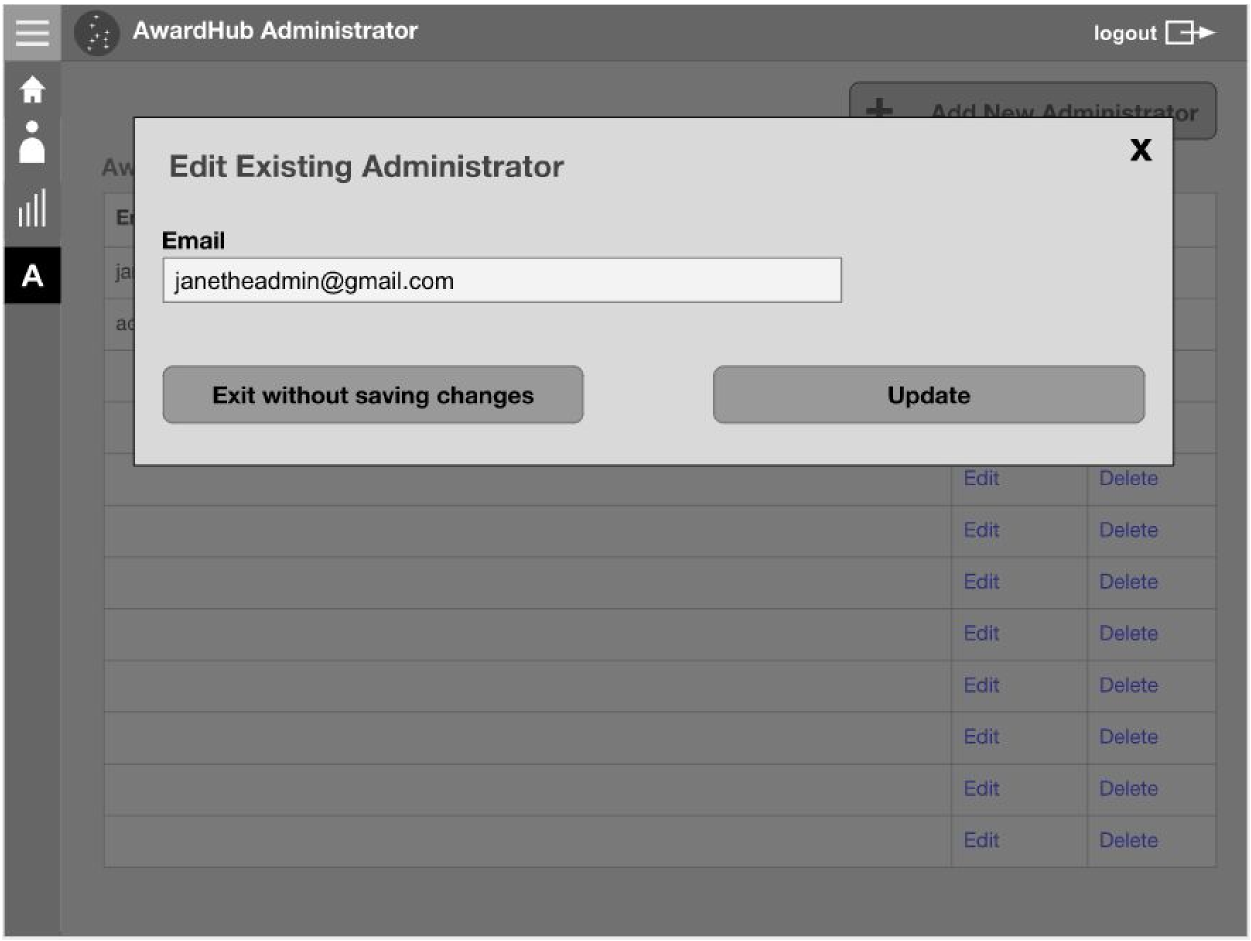
X

Email

Exit without saving changes

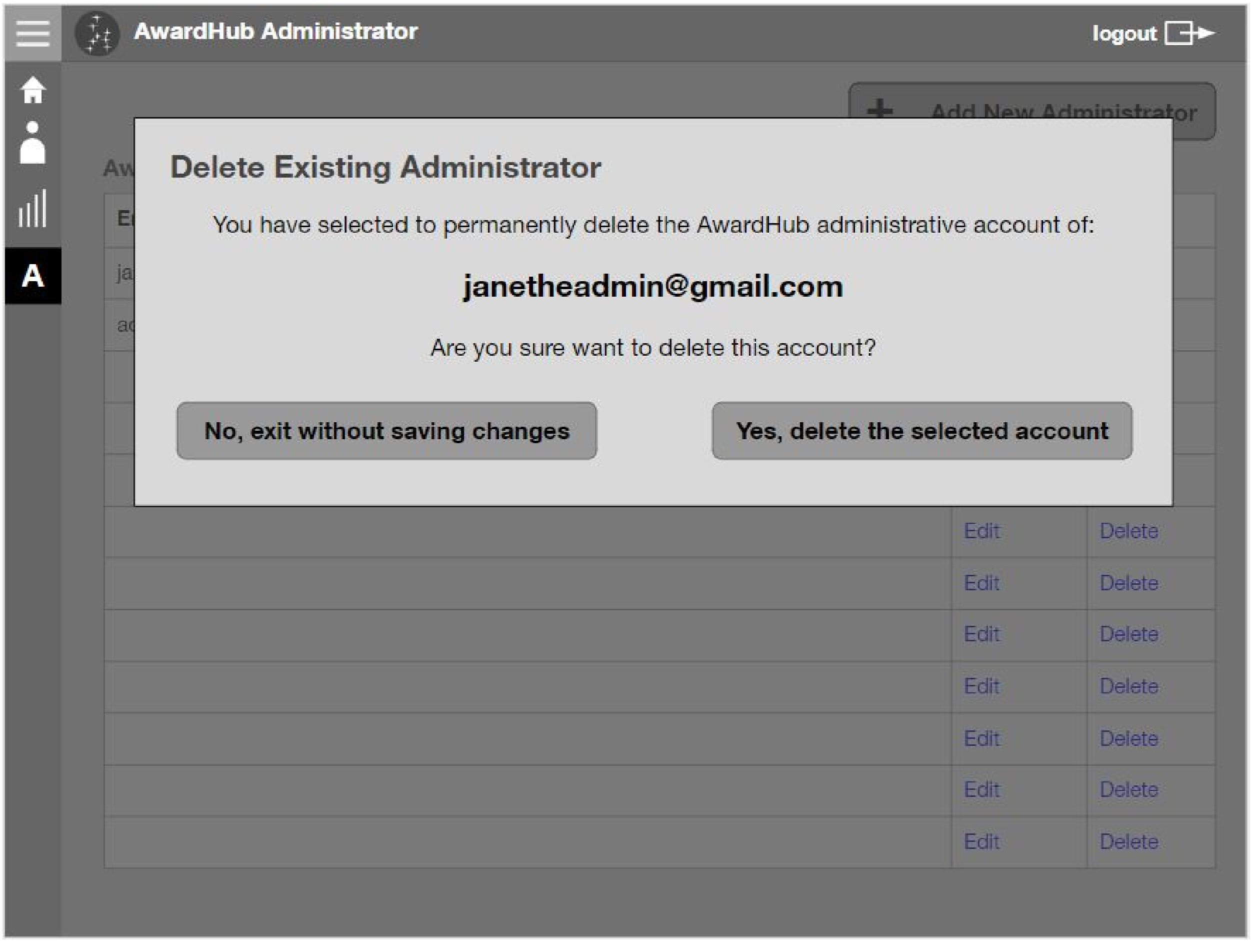
Submit new administrator

V. Manage Admin - Edit Admin



Email
janetheadmin@gmail.com

VI. Manage Admin - Delete Admin



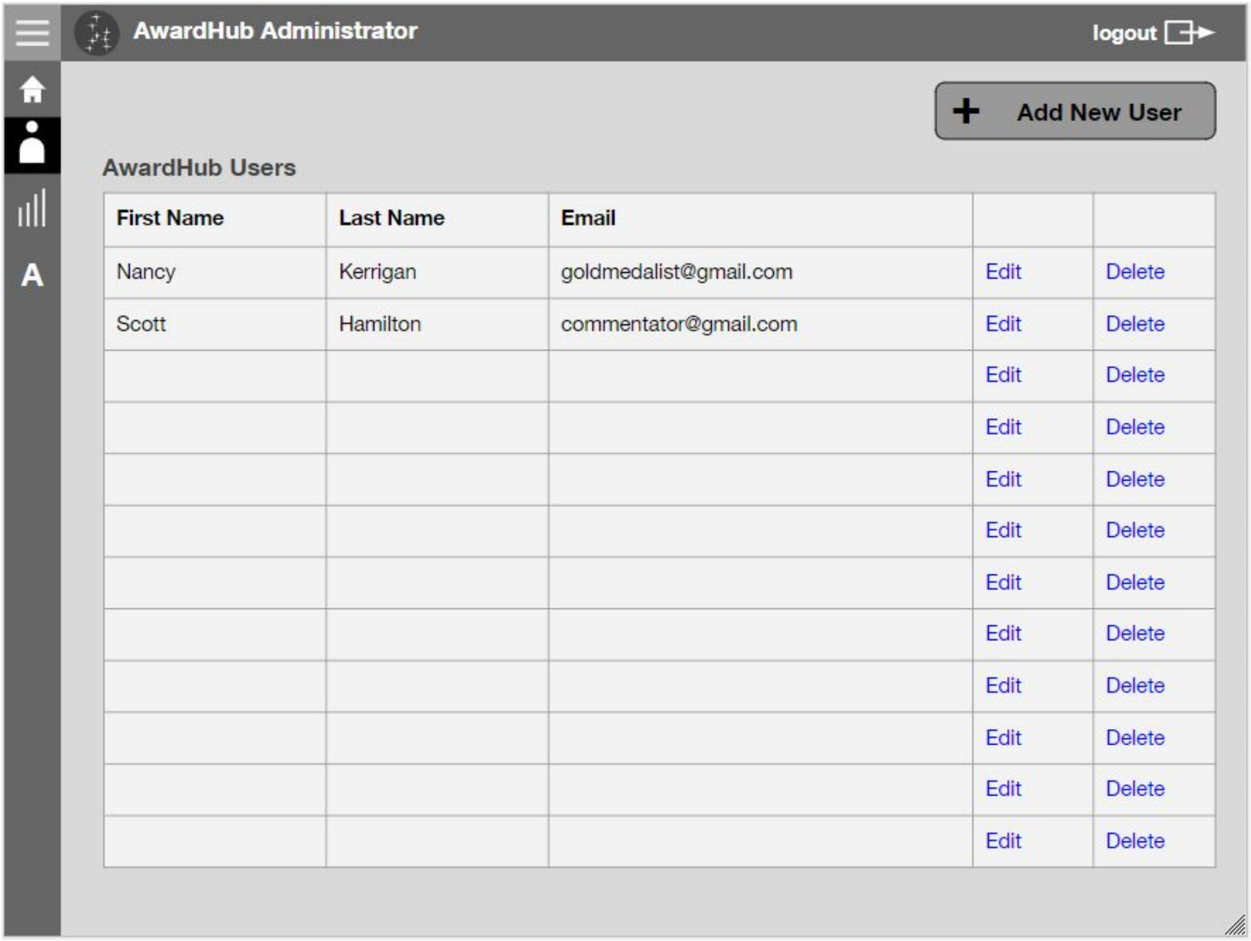
Delete Existing Administrator

You have selected to permanently delete the AwardHub administrative account of:

janetheadmin@gmail.com

Are you sure want to delete this account?

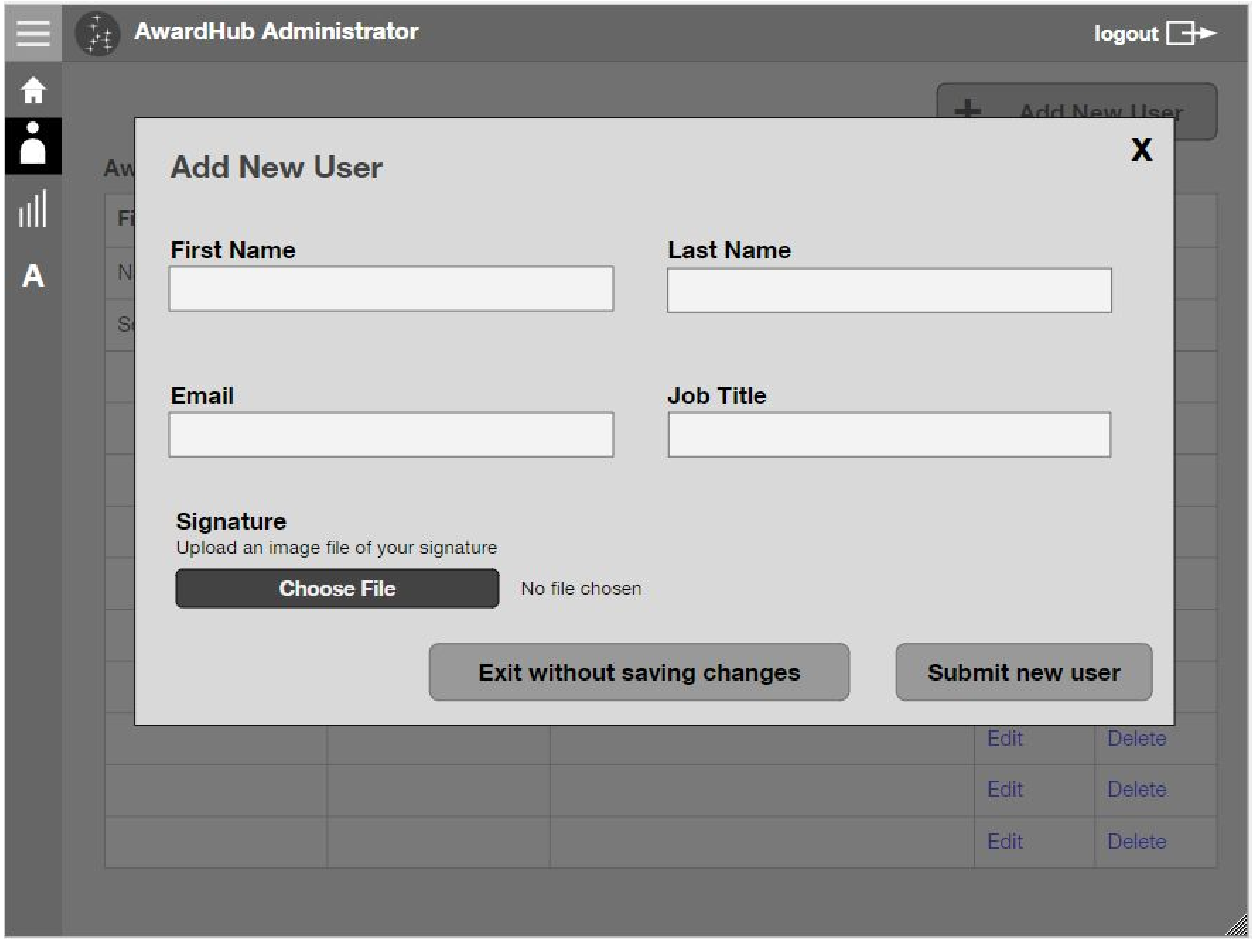
VII. Manage User - Home



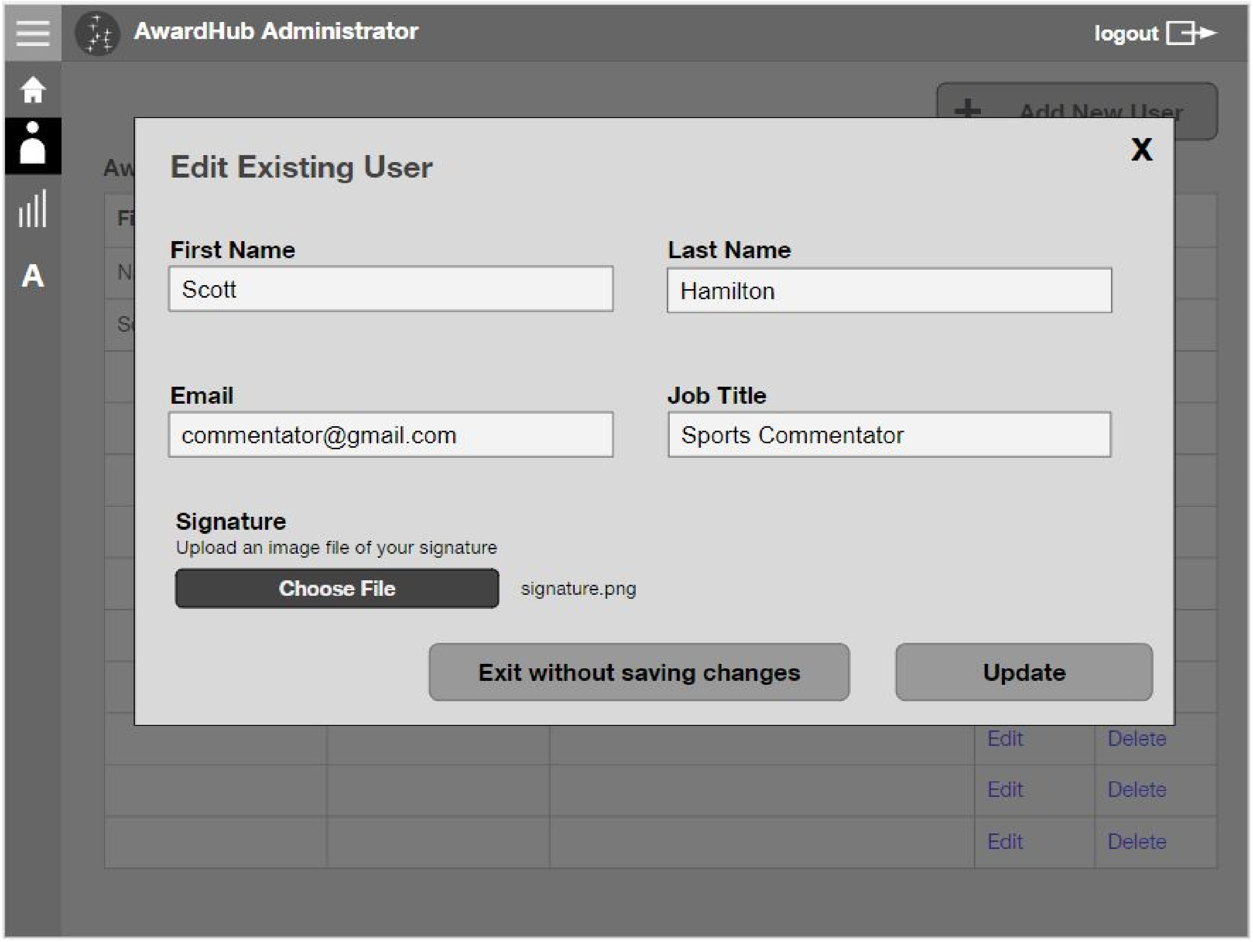
AwardHub Users

First Name	Last Name	Email		
Nancy	Kerrigan	goldmedalist@gmail.com	Edit	Delete
Scott	Hamilton	commentator@gmail.com	Edit	Delete
			Edit	Delete
			Edit	Delete
			Edit	Delete
			Edit	Delete
			Edit	Delete
			Edit	Delete
			Edit	Delete
			Edit	Delete
			Edit	Delete
			Edit	Delete

VIII. Manage User - Add New User



IX. Manage User - Edit User



Edit Existing User

First Name: Scott

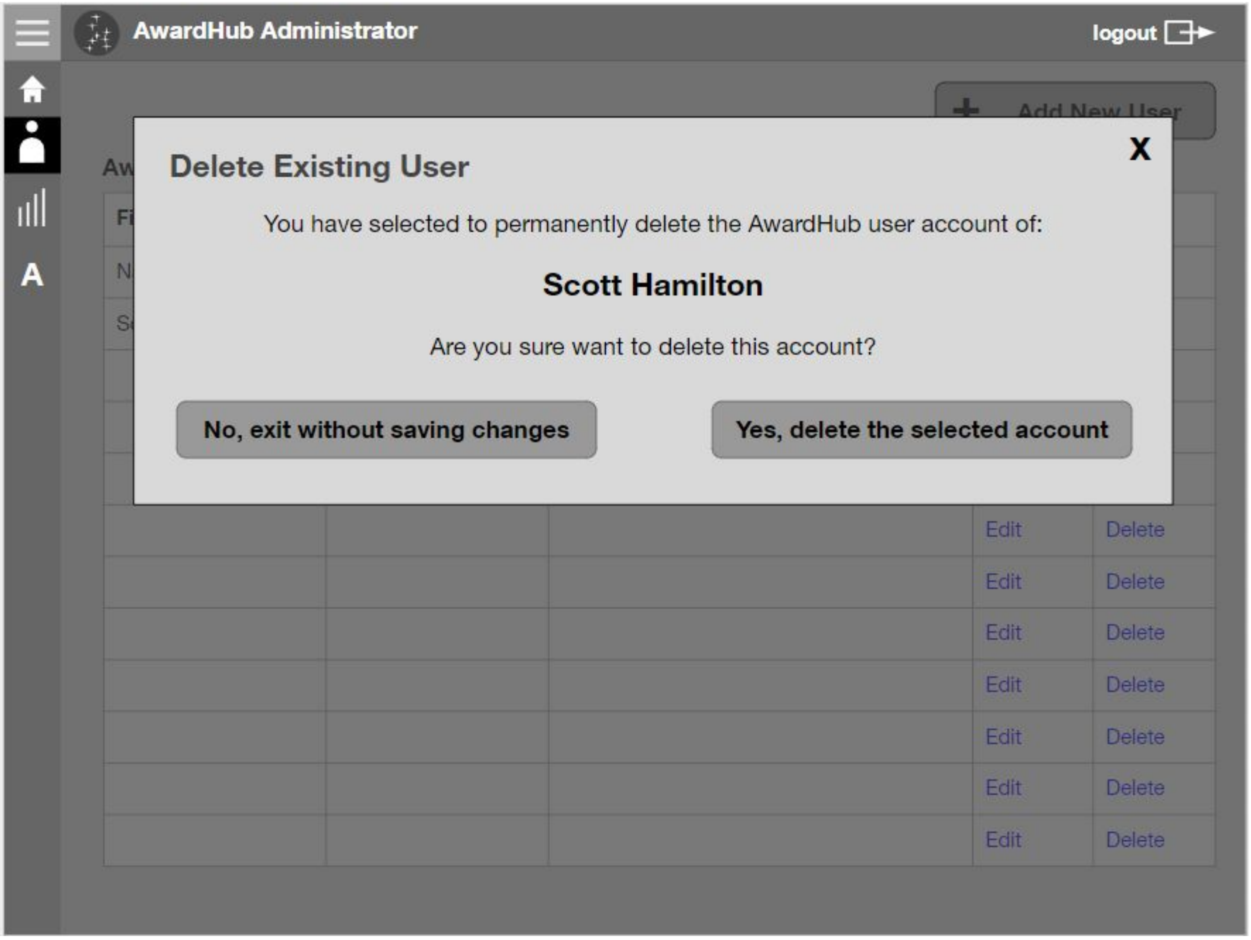
Last Name: Hamilton

Email: commentator@gmail.com

Job Title: Sports Commentator

Signature
Upload an image file of your signature

X. Manage User - Delete User



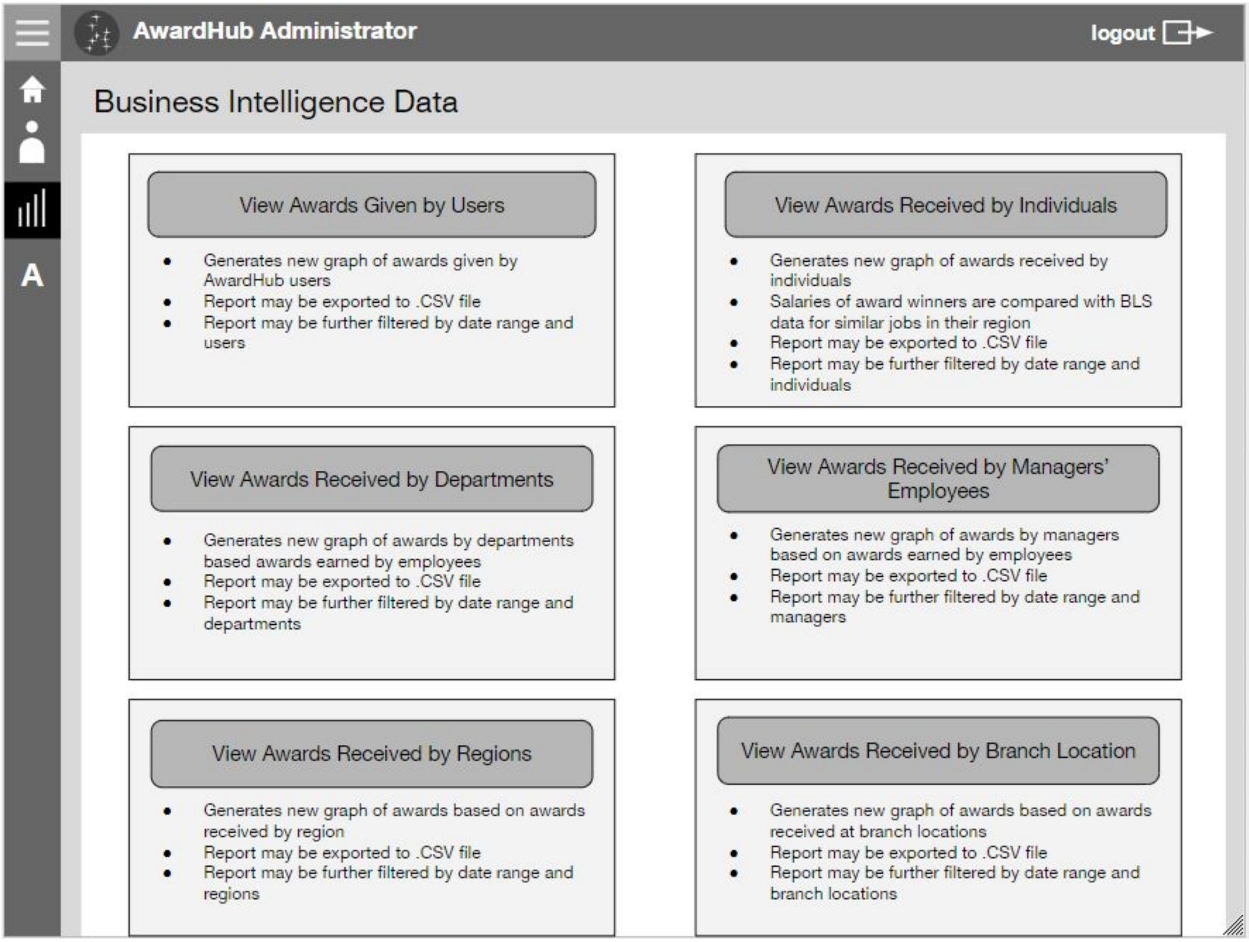
Delete Existing User

You have selected to permanently delete the AwardHub user account of:

Scott Hamilton

Are you sure want to delete this account?

XI. Business Stats - Home



Business Intelligence Data

View Awards Given by Users

- Generates new graph of awards given by AwardHub users
- Report may be exported to .CSV file
- Report may be further filtered by date range and users

View Awards Received by Individuals

- Generates new graph of awards received by individuals
- Salaries of award winners are compared with BLS data for similar jobs in their region
- Report may be exported to .CSV file
- Report may be further filtered by date range and individuals

View Awards Received by Departments

- Generates new graph of awards by departments based awards earned by employees
- Report may be exported to .CSV file
- Report may be further filtered by date range and departments

View Awards Received by Managers' Employees

- Generates new graph of awards by managers based on awards earned by employees
- Report may be exported to .CSV file
- Report may be further filtered by date range and managers

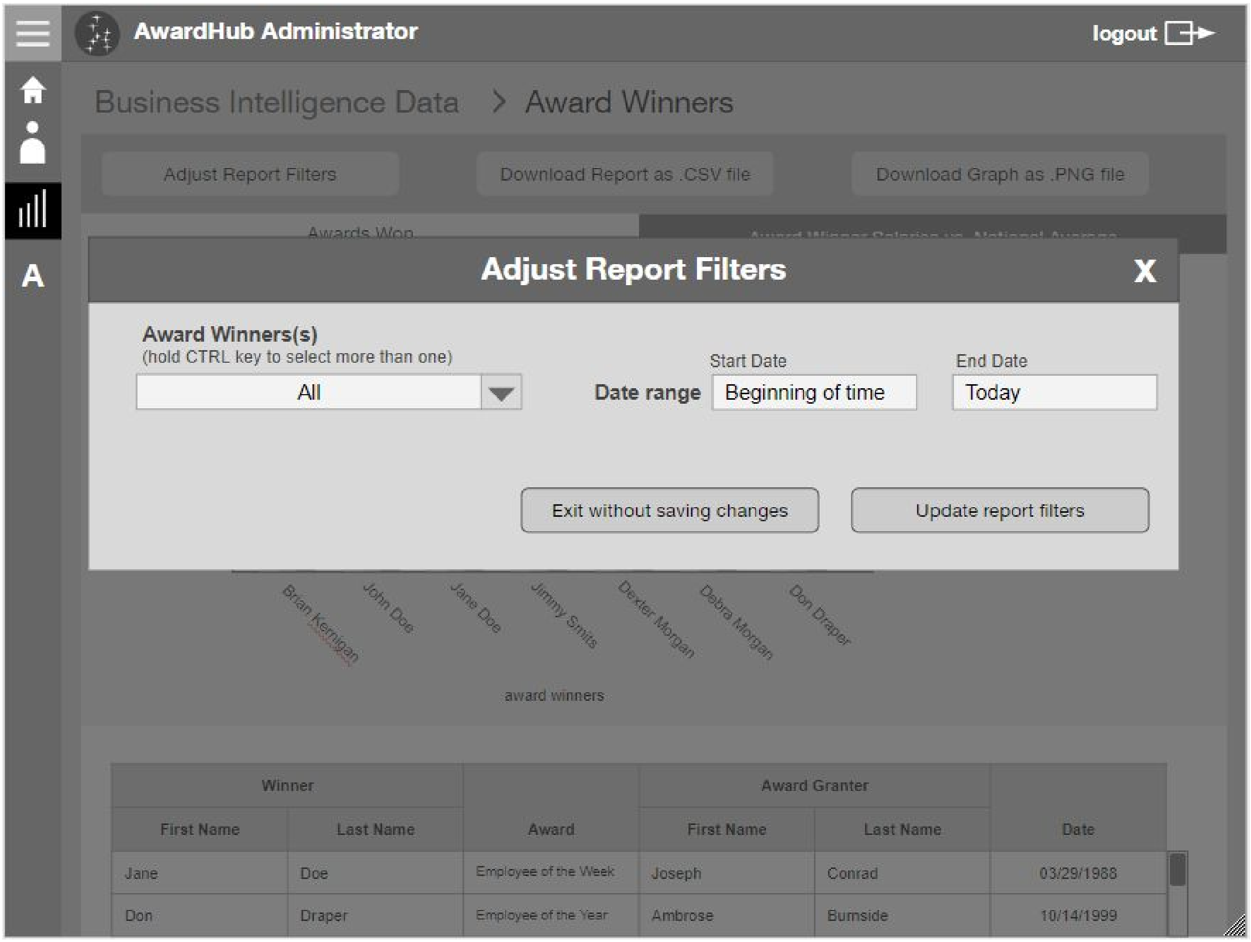
View Awards Received by Regions

- Generates new graph of awards based on awards received by region
- Report may be exported to .CSV file
- Report may be further filtered by date range and regions

View Awards Received by Branch Location

- Generates new graph of awards based on awards received at branch locations
- Report may be exported to .CSV file
- Report may be further filtered by date range and branch locations

XII. Business Stats - Filter 1



Business Intelligence Data > Award Winners

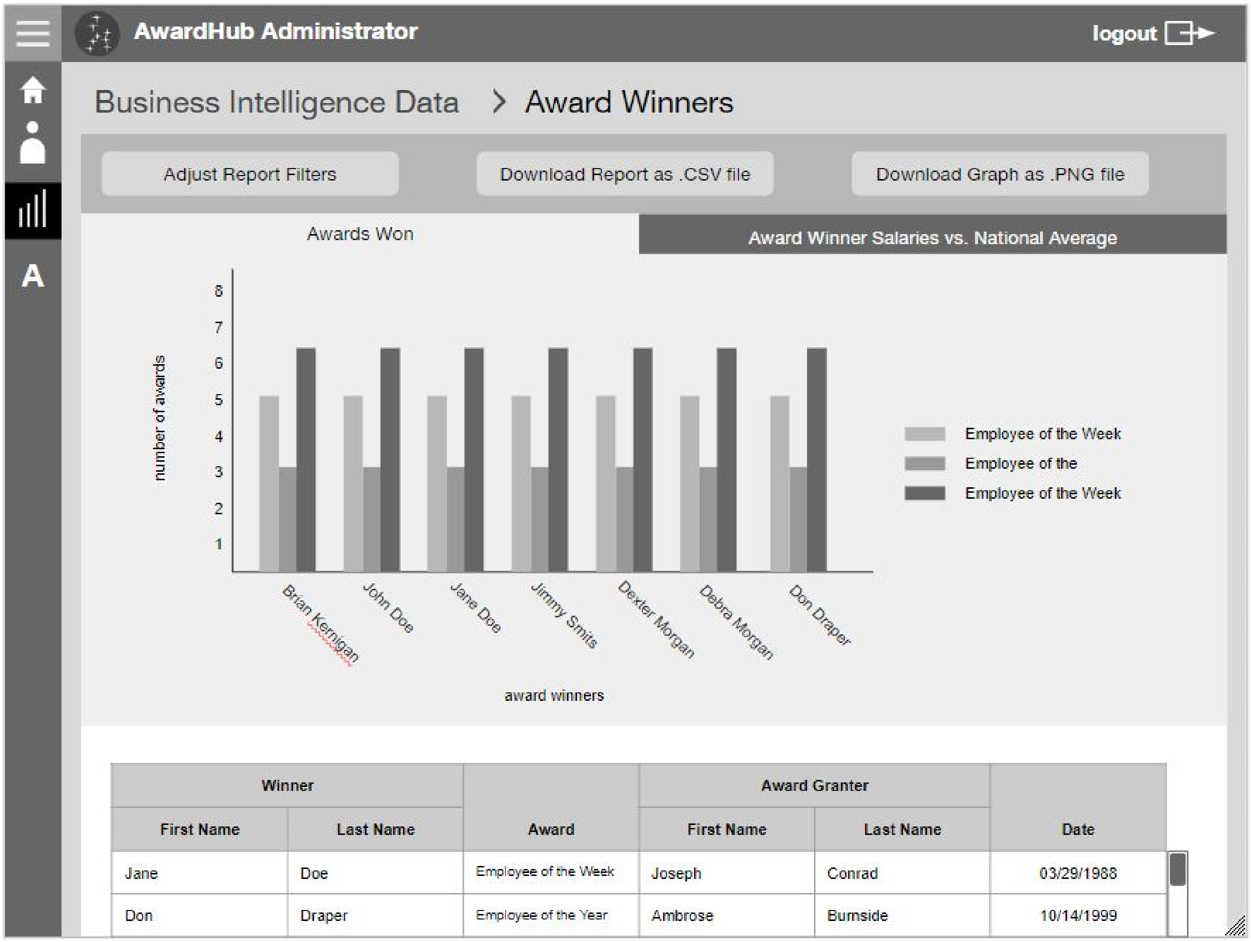
Adjust Report Filters

Award Winners(s)
(hold CTRL key to select more than one)
All

Date range
Start Date: Beginning of time
End Date: Today

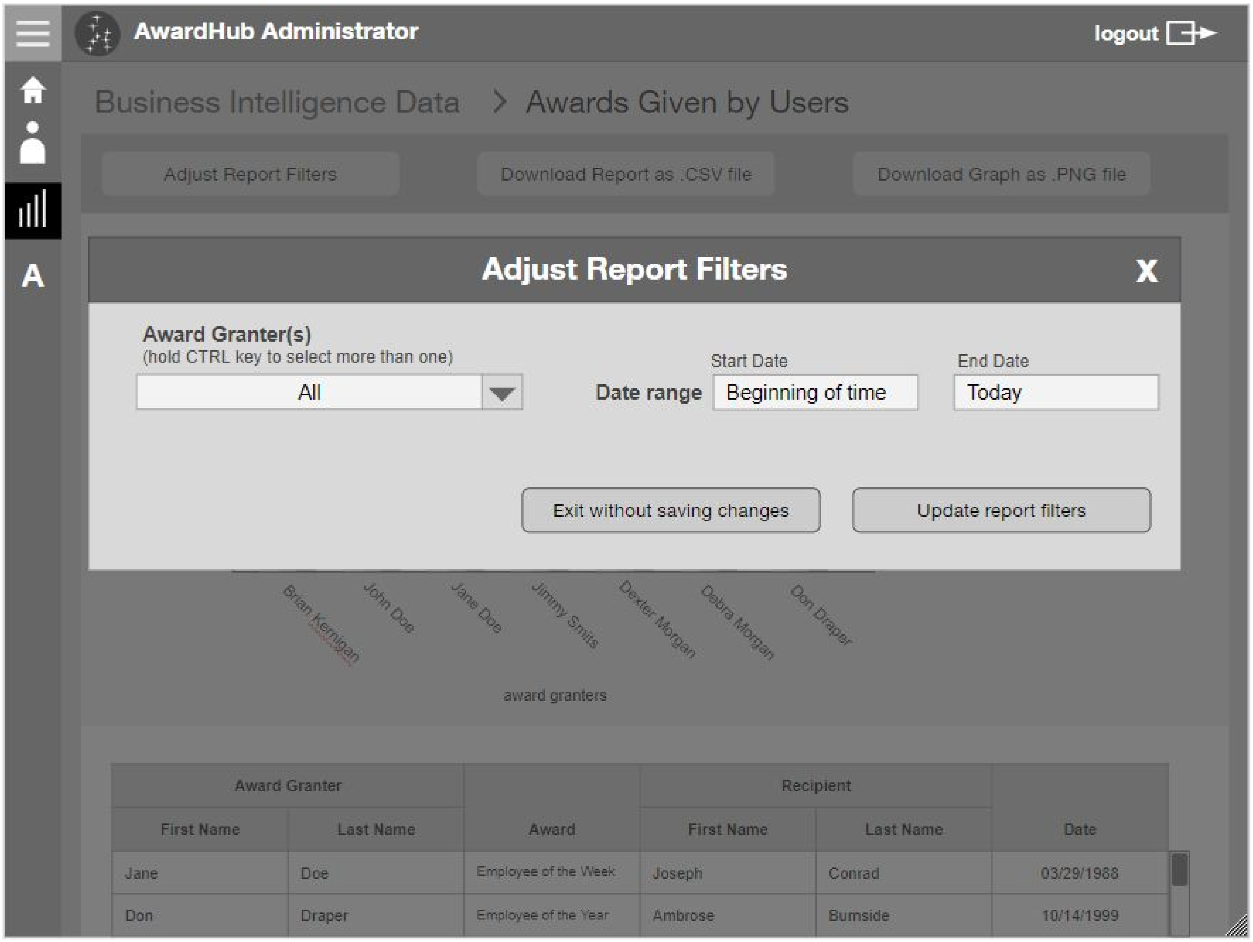
Winner			Award Granter		
First Name	Last Name	Award	First Name	Last Name	Date
Jane	Doe	Employee of the Week	Joseph	Conrad	03/29/1988
Don	Draper	Employee of the Year	Ambrose	Burnside	10/14/1999

XIII. Business Stats - Dashboard 1



Winner		Award	Award Granter		Date
First Name	Last Name		First Name	Last Name	
Jane	Doe	Employee of the Week	Joseph	Conrad	03/29/1988
Don	Draper	Employee of the Year	Ambrose	Burnside	10/14/1999

XIV. Business Stats - Filter 2



Business Intelligence Data > Awards Given by Users

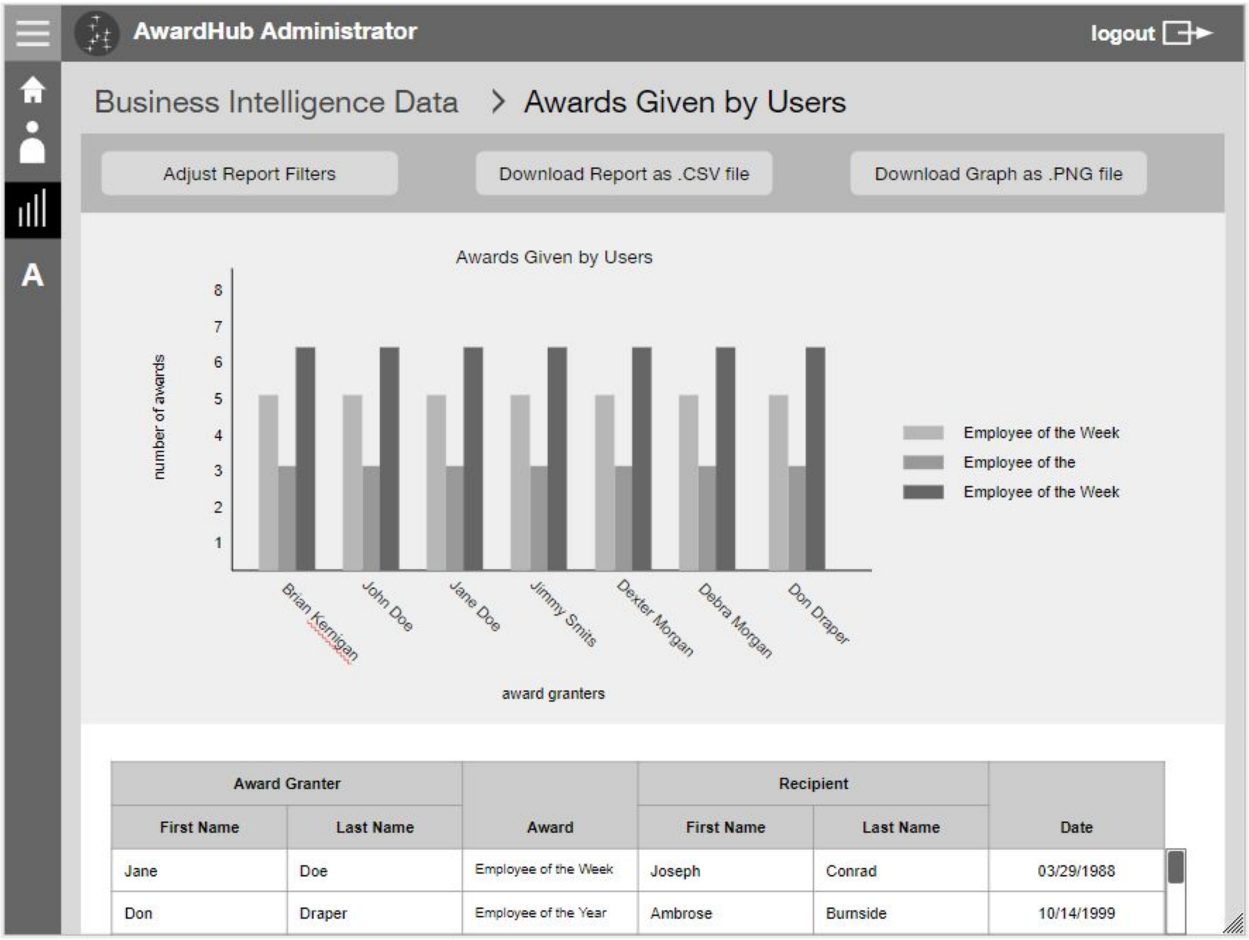
Adjust Report Filters

Award Granter(s)
(hold CTRL key to select more than one)
All

Date range
Start Date: Beginning of time
End Date: Today

Award Granter			Recipient		
First Name	Last Name	Award	First Name	Last Name	Date
Jane	Doe	Employee of the Week	Joseph	Conrad	03/29/1988
Don	Draper	Employee of the Year	Ambrose	Burnside	10/14/1999

XV. Business Stats - Dashboard 2



Award Granter			Recipient		
First Name	Last Name	Award	First Name	Last Name	Date
Jane	Doe	Employee of the Week	Joseph	Conrad	03/29/1988
Don	Draper	Employee of the Year	Ambrose	Burnside	10/14/1999