Homework 7: Implementation 2

Project Group 20: SportSponsor

James Doocy
doocyj@oregonstate.edu

Carissa Foster
fostecar@oregonstate.edu

Trang Lam
lamtr@oregonstate.edu

Allan Ngei
ngeia@oregonstate.edu

Brian Phair
phairb@oregonstate.edu

Oregon State University
Corvallis, Oregon

KEYWORDS

application, sports, league, charity, children, youth, sponsor, physical health, mental health, academic achievement, income gap, low-income families, financial barriers

1. VIEWING THE APPLICATION

1.1. URL

flip1.engr.oregonstate.edu:6544

1.2 Special Instructions

If you are logging to view the website on flip, please ensure you are on the OSU server or using VPN to access flip.

To VPN into the OSU server, open Cisco AnyConnect and connect to sds.oregonstate.edu. Next enter your onid username and password. If successful, the VPN connection should be complete. Full instructions can be found at: https://oregonstate.teamdynamix.com/TDClient/KB/ArticleDet?ID=51154.

The website does not adjust for different size screens, so the user may need to zoom out to view the full pages.

2. USER STORIES IMPLEMENTED

2.1. User Log In

2.1.1. Teammates Who Worked The Story

Brian

- Merged the code written by James into the handlebars files.

Allan

- Added the login view as the redirect destination for the sign-in option on the homepage

James

- Added the login HTML with a username and password form submitted by post, and the accompanying CSS

2.1.2. Unit Tests Completed

To set up for testing, an account was set up in the database with a username of “Capitals” and a password of “best”.

To test if the link to the login page worked properly, the link was clicked. Both the Chrome and Firefox browsers were used in these tests. Each time, the login page was loaded.

The next test was to see if the user could log in to the website. This was done by entering "Capitals" into the username box and "best" into the password box. Then the submit button was clicked, and the account home page was then loaded with the account information displayed in multiple tables. This means that the user is able to access their account if they enter the matching username and password.

The next test was to see if an account could be accessed if the wrong password was entered. This was done by entering "Capitals" into the username box and "best1" into the password box. When the submit button was clicked, we were redirected to the account page; however, no information was displayed in the tables. This means that the account will only be accessible if the correct username and password are entered.

2.1.3. Problems Encountered

An issue we encountered had to do with the configuration of our database tables. The username field for instance in the login page form would cause a duplicate entry error upon submission if there existed another entry in the database with the same username.

2.1.4. Completion Time

The HTML and CSS development time took around two hours.

2.1.5. Current Status

Currently, the user is able to login and access their information store in the database.

2.1.6. Remaining Tasks

One consideration is how the page looks. Some thought will be given to the CSS not only for this page but for all pages within the website so that a "brand" can emerge that reflects the the goals and mission of SportSponsor. See Section 2.2.6 on some thoughts on Branding and usability.

2.2. Family Home Page

2.2.1. Teammates Who Worked The Story

Brian

- Updated some of the JavaScript that Allan wrote

Allan

- Added the login page Brian created to the SportSponsor website
- Set up the login process to include a query to the SportSponsor database for user verification. Upon successful login, the family user is redirected to the family account page (created by Brian) where user-specific sign-on and contact information is displayed

2.2.2. Unit Tests Completed

The user is now able to view their account information. Other user stories involve the user creating an account and logging into the site. The user will create an account by creating a unique username and then a password. After the account creation, the user will then proceed to login (see Section 2.1). If the user enters the matching username and password combination, all of the user's information [they had entered previously upon registering] will be displayed on a new page.

The first test was to test if the user enters the right username and password combination. An account was created with a username of "Capitals" and a password of "best". When the username and password was entered into the login form, a new page was loaded with the appropriate account information.

Another test was to see what would happen if the wrong password was used. The account with the username "Capitals" was used for this testing. In the login form, "Capitals" was entered into the username section and "best1" was entered into the password section. Upon hitting the submit button, a new page was loaded however no information from the account belong to "Capitals" account was loaded to the screen. Therefore, only when the correct

username and password combination is entered will the account then be displayed on the screen.

2.2.3. Problems Encountered

During the completion of this task, we encountered challenges stemming from the small amount of feedback given by the database in the cases where queries did not produce errors. Whether form-submissions had updated the database properly was difficult to say during testing as a result.

Another database related issue was the long duration between some queries and the database responses to them. This sometimes caused a sort of connection time-out error that prevented us from determining the efficacy of our queries.

As currently constructed, the website suffers from a few bugs we believe might be the result of improperly accessing database response object fields.

2.2.4. Completion Time

The development time took around five hours. Since Handlebars is being used, most of the web page layout was already established. Therefore, most of the time was devoted to developing the forms and writing queries.

The time spent on testing took about one hour.

2.2.5. Current Status

Currently, the user is able to see their account information displayed in several tables. The user will only be able to see the account information of the matching username and password.

2.2.6. Remaining Tasks

One of the areas that will need the most work is developing how the forms look on the web page. A minimal amount of CSS was used when working on this user story. A border was added to the table to be able to identify certain positions. Currently, if no information is contained in the database, the table has an empty box (in CSS file: border: 1px solid black). Empty boxes will be need to be *not displayed* or "N/A". Other possible CSS considerations is centering the tables in the middle of the page and having bold characters for table headers. Branding (which includes color, font, borders and so much more) was a topic that the team had started to consider. If this project was taken to completion, the appearance, user-interface, and Branding would be a major consideration that would visually promote the goals of SportSponsor and would be part of the Remaining Tasks to be completed.

Another task that needs solving is how to handle the situation when a user enters the wrong credentials. Currently, an error message is sent to the server about a bad input. However, a message (alert window or a new web page) will need to be sent to the user to inform them of the situation.

3. SPIKES AND UML DIAGRAMS

3.1. Spikes

As stated last week, our Spike Testing including using forms, GET, and POST. Please see below for details.

To explore potential solutions for SportSponsor, we used two simple programs using forms and web pages. To summarize, our application will need a front-end, a form for the Users to input data, and a place to collect and store the date. To that end, we used 2 methods [GET and POST] that included forms, and then we outputted that data onto a webpage. Our goal was to see if we could quickly put a basic form together, pass the input from the form into a variable or database, and then output the data to a webpage. We were able to successfully use the spike/product-testing method to gain insights into how we would build SportSponsor, and we then used these insights to work on the prototype for SportSponsor. For example, some of the insights gained is that (1) creating a form that has 2 input-fields helped us to better understand forms, and (2) each form took about 4 hours; since SportSponsor will use 10+ input-fields, we know that it will probably take us another 12+ hours to create the form that the User (e.g. Family) will use. Therefore, using spike as a product-testing method will help us to

understand the level of effort needed and the time needed to complete certain tasks.

Note: The code for the Spike Testing was used by the team to estimate the schedule and research possible solutions. Please see Section 7 Integration Tests for more details. The original code was only kept for study-purposes [since this project is part of a college/school program] and can be found on a public GitHub account here:

https://github.com/lamtr/Get-Post-Testing

3.2. UML Diagrams for HW7

3.2.1. Log In Page

The Log In page shows the process of a user logging into the SportSponsor website and then being directed to a page specifically for their user type. The username and password entered will be cross referenced with registered users to determine the user's user type.

This diagram is useful to visually see the log in process including what actions and information are needed from the user and what actions are triggered after the user initiates the log in.

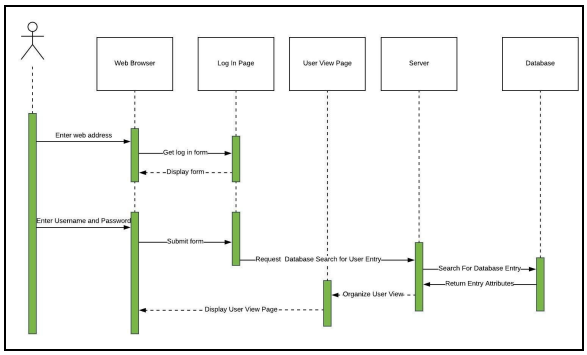


Figure 3: UML Diagram - Account Creation

*See Appendix E for larger version

3.2.2. Family View - Sponsorship Request

The Family View is the page family users are shown upon entering their login information. On the Family View page, family users will be able to search for leagues to register for and select any required equipment they need help purchasing. All of this is recorded in the SQL database.

This diagram is useful to have during implementation to view the overall vision for the Family View Page. However, having a simplified version modeling anticipated progress on this page would have been more helpful. The simplified version for this week would have showed any username and password would have resulted in a login with a table being displayed for any login attempt. If a family user match is found in the database the table will be populated with information provided during account creation, otherwise the table would be empty.

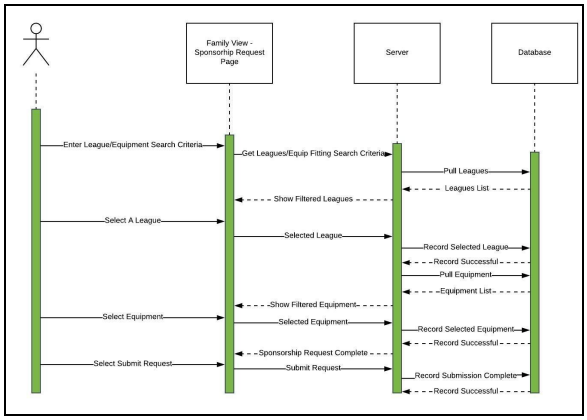
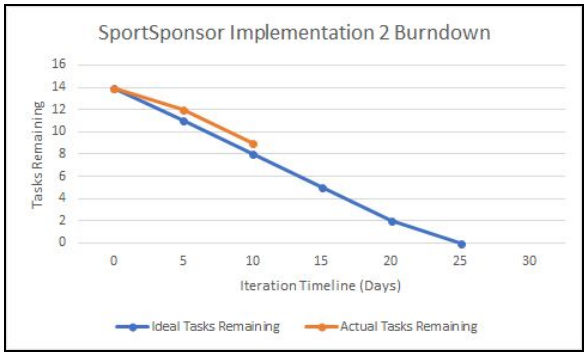


Figure 4: UML Diagram - Account Creation

*See Appendix F for larger version

4. BURNDOWN DIAGRAM



*See Appendix K for larger version

As seen on the chart, we have continued to be on track, though slightly behind the ideal implementation timeline. (**Note:** 'slightly behind' is due to implementing a database. Now

that the database is set up, this work will support the completion of future tasks since all the forms require database access.)

We had 4 tasks to complete in the first week of our implementation, and we had finished 2 by 5 days in. At 7 days in, we had completed 4. We currently have 7 tasks completed. By the end of this second week, at 14 days, we would have 8 total tasks completed. If we were to continue implementing all the tasks we considered, we estimate a total of 24 days to finish them all. We are on track for our ideal tasks completion rate.

5. REFACTORING

Currently, we do not have any items that are refactored yet since we have only started working on the code for our application. This week for HW7, the login page and an initial family view page were implemented. Our work has not been redundant yet.

However, based on the design of SportSponsor, we can theorize that we will be refactoring such items as the form that asks the User for his contact information. All users (e.g. Family, Sponsor, League/Vendor) will need to input their contact information including:

- Name
- Address
- Phone Number
- Email Address
- User-type (e.g. Family, Sponsor, or League/Vendor)
- Username/Password

Once the form for the Family is created, we know that we can re-use this same code, and at that point in our process, we believe that refactoring of the code will be useful in creating this app.

This was mentioned in the past week and proved to be true...the refactoring of the database. In this case, how we will structure the tables/ schema in our database for each user-type: Family, Sponsor, League/Vendor. The process of setting up our database will be similar in that we will want to have security settings in place (e.g. login, passwords) for each table and user-type. The structure of our database and the way the SQL queries in our database are written may require refactoring. From the past week, we had set up the schema [of the database] for three additional users in addition to the Family: Sponsor, League, and Vendor. The refactoring for the database was started and may need further planning. As stated in Section 2.2.6, further thought must be given to solving how to handle the wrong credentials.

A third refactoring of the code is the code that creates the capability that tells the User when he has been approved. (Note: This is scheduled for future deployments.) This capability will let the User know when his approval has been processed for approval [or denied]. For this capability, only the Family, League, and Vendor will be notified when he is approved. The Sponsor, who wants to donate and sponsor someone, will automatically be approved when his credit card is approved.

In summary, certain capabilities in the tool will be structured or refactored since there are commonalities in the way the tools are used in SportSponsor. As stated in the first paragraph of this section, the refactoring will occur when we continue to create more pages (e.g. code for Forms will be re-used). The refactoring will not occur until we find instances to reuse the code.

6. QUESTIONS FOR THE CUSTOMER

This week's discussion with the customer was focused on status of implementation of the SportSponsor application, what our group expects to be able to complete by Friday June 8th, and receiving customer thoughts and feedback on the current implementation.

The customer was informed that our implementation goals now appear to be unrealistic given the time restraints of the class. The customer was in agreement that the group should make as much progress this week and

discuss in this document what made various sections out of scope for this assignment. He also recommended to mark incomplete items as to be completed in the projected future schedule.

There have been no changes in requirements and changes to the SportSponsor application implementation progress for this week was the only surprise.

Possible questions to ask the customer in the future involve styling of web pages and how to handle certain errors. In section 2.2.6, one of the things that needs work is how to handle if the user enters the wrong information when logging in to the website. Should the user be redirected to a new web page or some type of alert pop up on the current page? If an alert pop up is desired, a new technology such as AJAX may be required; thus, prompting further discussions of possible implementations.

The customer reviewed the application and provided the following feedback.

7. INTEGRATION TESTS

Priorities for integration testing is the same as what was discovered from prior integration testing thoughts. After some research, discussion among the team, and Spike testing, the decision was use technology that would promote success when it came to Integration Testing:

- **Accessible**
- **Templatable**
- **Ease of Use**
- **Compatible**

We used HTML, CSS, JavaScript (JS) for the front end and use JavaScript and Node.js to access the back-end and access the SQL database. The database is currently hosted on Amazon Web Services (AWS). Integration of these various technologies was done in several ways.

For the Front-end, HTML, CSS, and JavaScript is compatible and easily integrated in the same document and viewed in a browser (e.g. Internet Explorer, Firefox, Safari, and Chrome.) The forms for each user was created first created in HTML and JavaScript [during Spike testing]. Once the SQL database was added, it was decided that based on the team's skills, we would use Handlebars JavaScript (JS) as a templating engine in order to create the forms on the webpages [since we knew that we had 3 major types of User and we would need forms for each User]. Handlebars JS allows the integration of HTML in that parts of the HTML could be easily rewritten in Handlebars. Handlebars also has its own set of formats/structure, similar to the result of CSS. Since Handlebars is a JavaScript library, that makes it compatible with any JavaScript that we need to write. In addition to Handlebars, using Node.js allows us to write JavaScript on the back-end, which then allows us to work with our SQL database, which is where our data is stored. In addition, Amazon's AWS is an Infrastructure-as-a-Service which allows compatibility among diverse technologies, which includes the backend and frontend.

We hosted our code on GitHub so that we could view it among our team members and comment on it. We also shared code via Google Docs.

In summary, we used technologies that were:

- **Accessible** to our team (.e.g on the Cloud, GitHub, and Google Docs)
- **Templatable**/easy to create templates since we recognized that with 3 Forms for the Users that were similar, it makes sense to create a template so we can reuse code and processes
- **Ease of Use** in that it was easy to work with (e.g. code that our team had experience in)
- **Compatible** among themselves (e.g. JavaScript is compatible with both

backend and frontend, SQL is highly compatible)

These choices in turn allowed for Integration Testing among the team.

Besides the Integration Tests and Research done above, using Handlebars, the JavaScript (JS) Library, and its templating engine has allowed the team to quickly created new web pages without having to refactor much of the code. Handlebars in JavaScript lends itself to quickly creating new pages, and this has proven true. For example, Handlebars JS can create views of the 4 Users (Family, Sponsor, League and Vendor); see **Appendix I**. The code is similar in nature since Handlebars takes care of integration and templating. And since Handlebars is in JavaScript, it is compatible with Node.js, which connects to the database.

Additional Testing for Week 7 was done to ensure that the views the Handlebars created was compatible with the forms [of each user]. Please see the results from the status of both the User Log In 2.1.5 and Family Home Page 2.2.5.

Note: Please see Section 2 for additional details of the testing of User Stories.

8. SCHEDULE FOR NEXT WEEK

If implementation of the SportSponsor application were to continue into next week, the following schedule was devised.

- Continue to modify implemented application pages including the Landing Page and Account Creating Pages started during the first week of implementation and the Log In Page and Family View Page implemented during the second week of implementation. Continuing to update these pages each week rather than waiting to revisit them once initial implementation of all pages is complete will make it easier to work on these pages since they are fresh in our mind and changes made may impact new pages - so rework time is potentially decreased. Anticipated time to make changes on existing pages is between 2 to 4 hours depending on what changes are needed and if any unexpected issues are encountered.
- User View Pages for all user types. Initial landing pages for various user types will be created. They will initially be very simple and just give indication that the right user was selected from the database and the user type identified. This lay the groundwork for forming the various user view pages and provide user specific actions to be performed. The anticipated 4 to 8 hours depending on the difficulty of looking up user types in the database and designing the various user view pages.
- Make Login Page more robust. Initial implementation of the Log In Page will be very simple and include limited examination of inputted username and password values in comparison to values saved in the SportSponsor database. The more robust Log In Page should be able to indicate whether the username or password are incorrect. Also, successful log in will result in the appropriate user view page being displayed. Anticipated completion time is 4 to 8 hours including time to research the best way to search the database for the matching username and password.
- Further functionality added to the Family User View. This new functionality will include entering league search criteria, returning database entries meeting the entered search criteria, and being presented to the family user so that one may be selected. This implementation is anticipated to

take 6 to 10 hours including researching how to use user input as database search criteria and presenting search results to the user in a format where a search result can be selected by the user.

- Additional testing of page and database prototypes will need to occur to ensure that new pages are working as expected and flow with pages and database interfaces created in the previous week(s). Full, detailed testing of the application during the third week of implementation is expected to take from 3 to 5 hours. Issues found will be addressed in the following week of implementation.

Planned tasks not completed will be pushed to the following week.

9. CUSTOMER INTERACTION

As stated previously, the Customer was responsive and timely in answering questions and interacting with the team.

10. GROUP CONTRIBUTIONS

Trang - Spikes Update, Refactoring Update, Integration Tests Update, Appendix I [on Templating], edited document

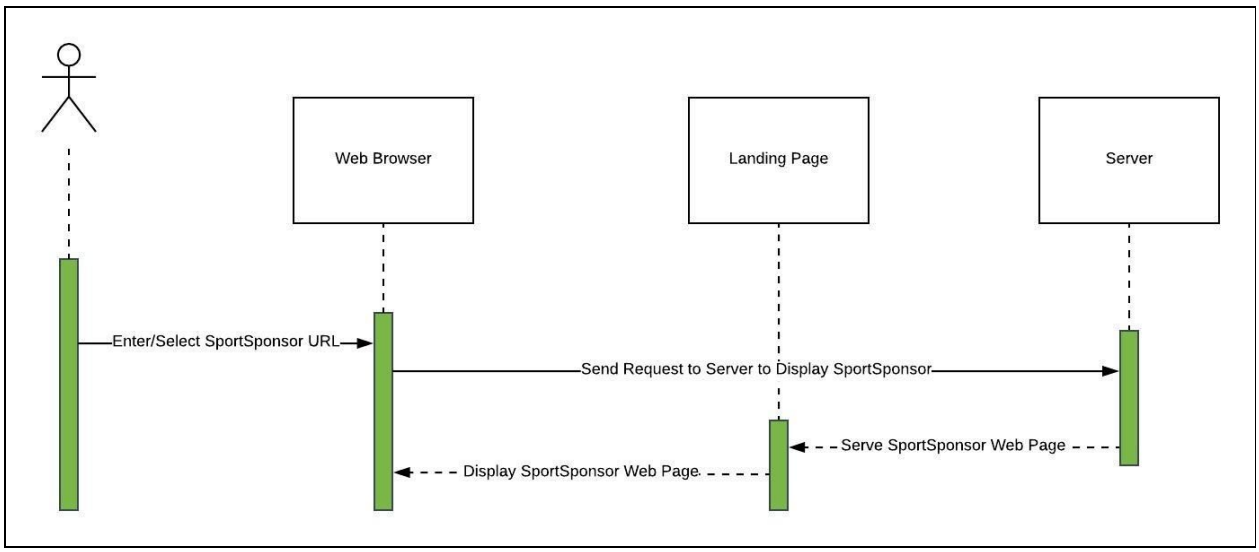
Brian - User Stories (All users have account registrations; User Login - Section 2.1; User Account Page - Section 2.2), edited document

Allan - User Stories (User Login - Section 2.1; User Account Page - Section 2.2)

James - Login User Story, Burndown Diagram

Carissa - Expanded URL Special Instructions, UML Diagrams Section Update, Questions for Customer, Schedule for Next Week, edited document

APPENDIX A: UML Diagram - Landing Page



APPENDIX B: UML Diagram - Account Creation

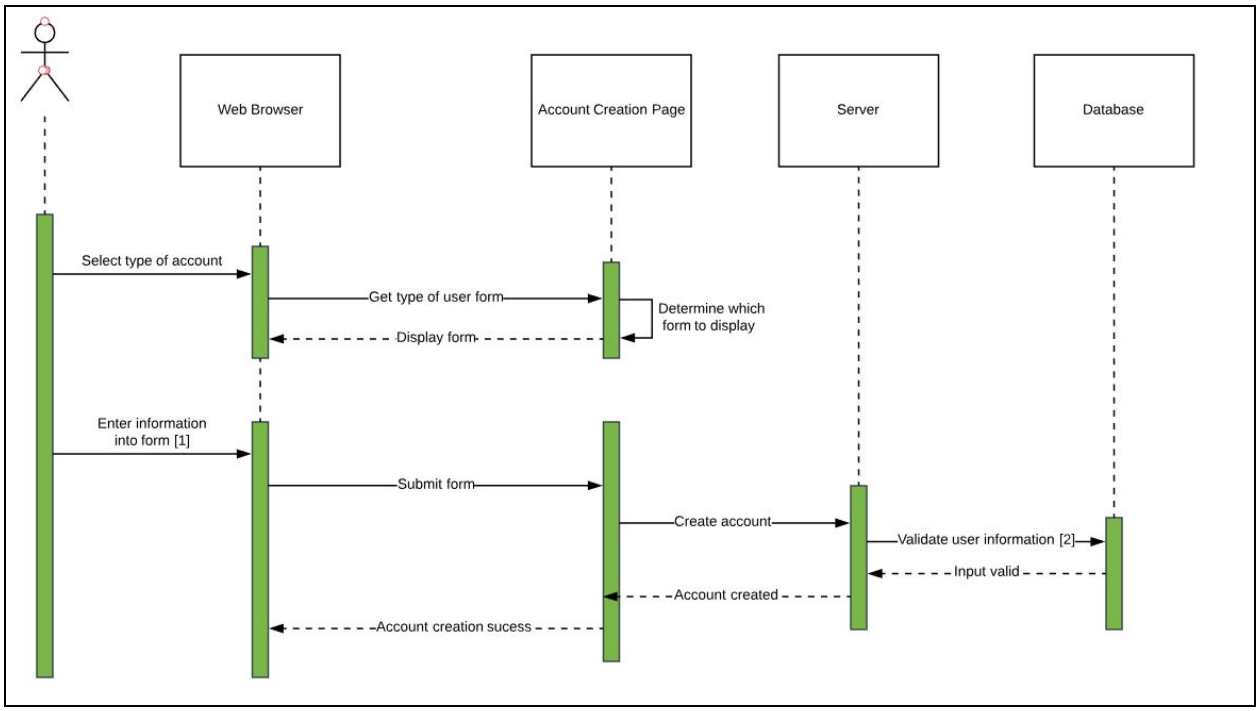


Figure 1: For specific information about user input into the form, see Appendix C Table 1. For specific information about input validation, see Appendix D Table 2.

APPENDIX C: User Input Table

Family	Sponsor	Vendor	League
User name	User name	User name	User name
Password	Password	Password	Password
First/last name	First/last name	Business Name	League Name
Profile photo	Home address	Business photo	League photo
Income level	Sporting interests	Business address	League sport type
Income documents (e.g. W2, pay stubs)	Remain anonymous	Equipment offered	League location/radius
Home address	-	URL link to website	League fees
Sport played	-	-	Program(s) available
Sporting equipment needs	-	-	Equipment requirements
League fees needs	-	-	-

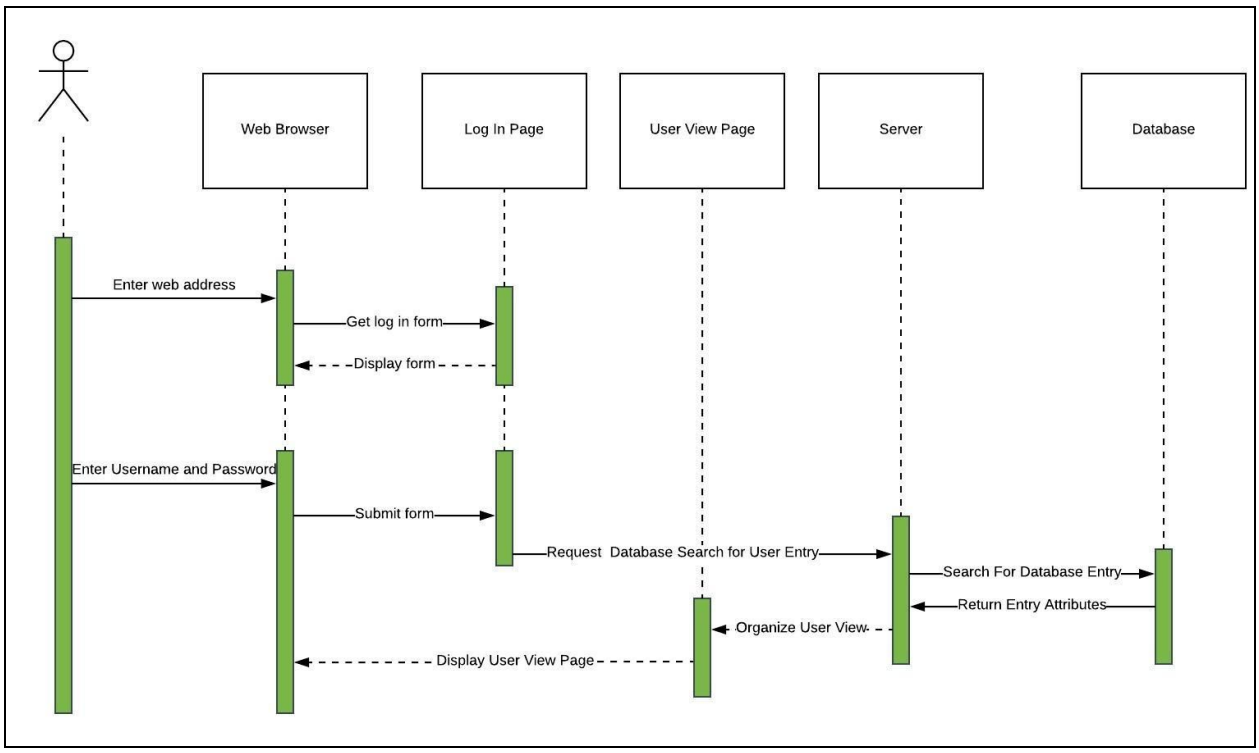
Table 1: Specific user input per user type.

APPENDIX D: Possible Validation Parameters

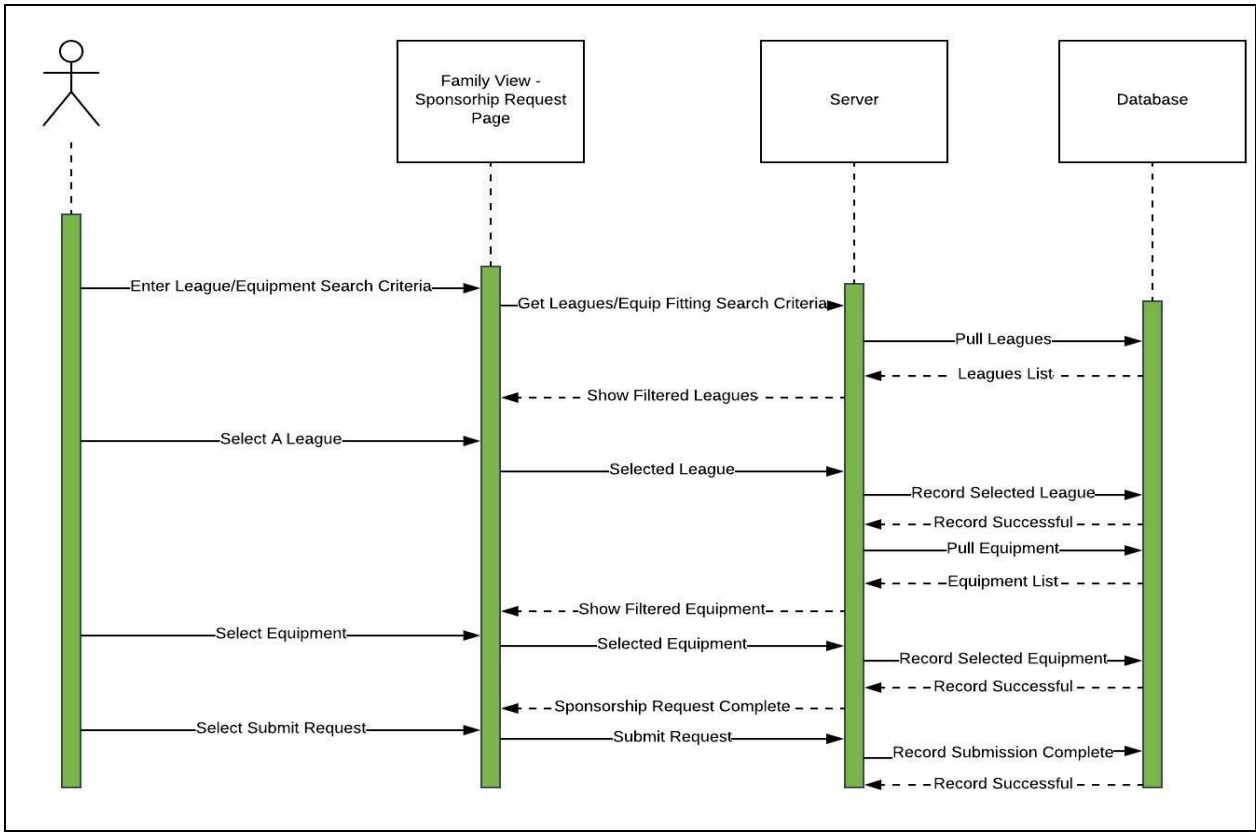
Family	Sponsor	Vendor	League
Unique username	Unique username	Unique username	Unique username
Password meets minimum requirements	Password meets minimum requirements	Password meets minimum requirements	Password meets minimum requirements
First/last name is not associated with another account	First/last name is not associated with another account	Business name not associated with another account	League name not associated with another account
Photo is appropriate [1]	-	Photo is appropriate [1]	Photo is appropriate [1]
-	-	Check if up-to-date business address	Appropriate equipment is requested (e.g. if league is a tennis league and the equipment required is a basketball, there's an input error)
-	-	Check link is not a broken link (see link (1))	-

Table 2. Validation parameters per user type. [1] The software to validate appropriate photo (e.g. no nudity/profanity) is at the minimal priority and thus may not be developed until main features are functional.

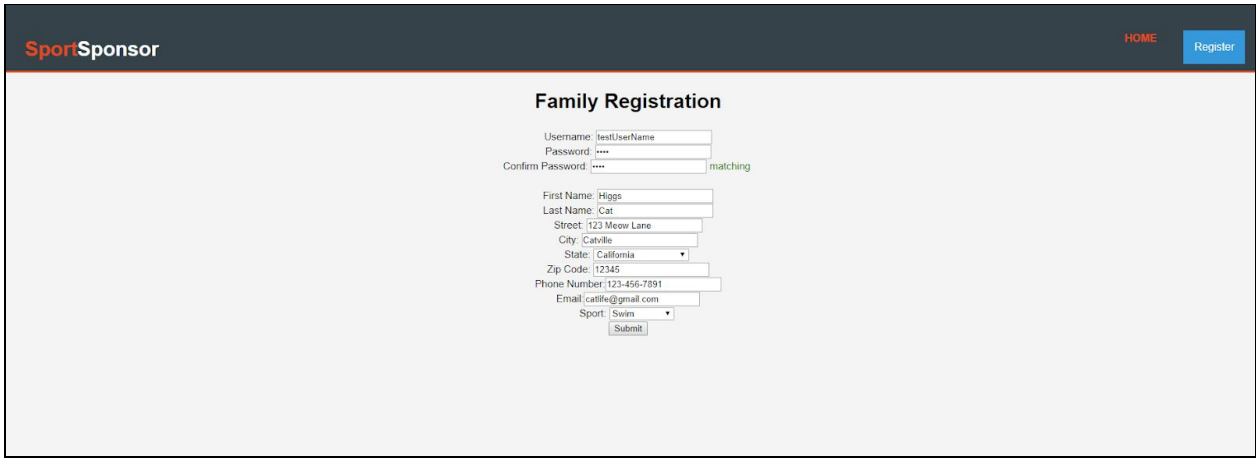
APPENDIX E: UML Diagram - Site Log In



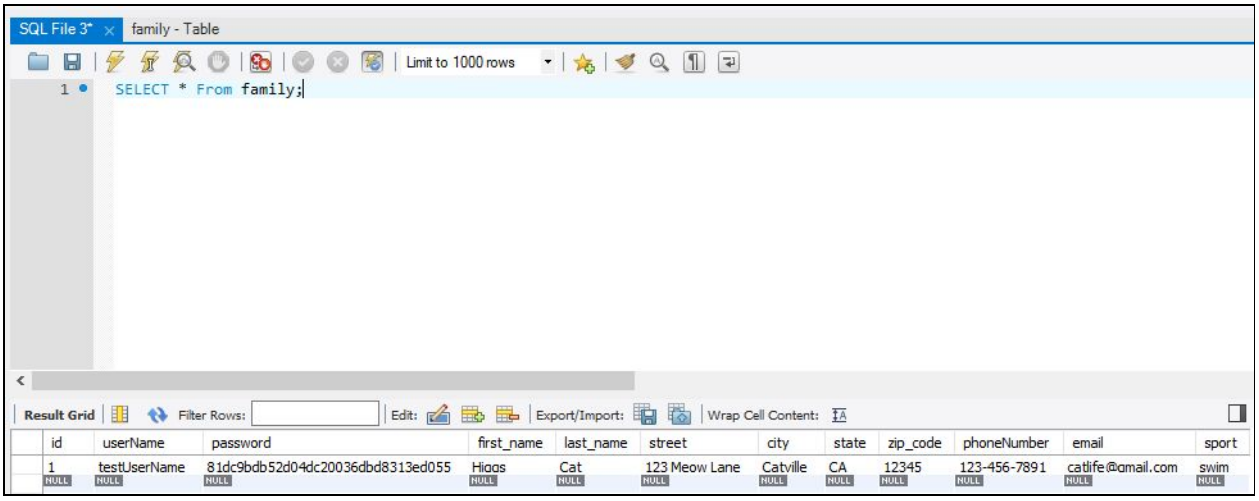
APPENDIX F: UML Diagram - Family View (Sponsorship Request)



APPENDIX G: Form Accepting User Input:



APPENDIX H: Database Showing Input from the Form:



APPENDIX I: Sample Code to show Templating and Integration Tests (and Results)

As stated, the code shows the templating engine at work. Please note the similarity in the structure of the Code in the 4 examples highlighted in blue.

Example 1:

```
app.get('/registerFamily', function(req, res){
  res.render('registerFamily', {layout: 'register.handlebars'});
});

app.get('/registerSponsor', function(req, res){
  res.render('registerSponsor', {layout: 'register.handlebars'});
});

app.get('/registerVendor', function(req, res){
  res.render('registerVendor', {layout: 'register.handlebars'});
});

app.get('/registerLeague', function(req, res){
  res.render('registerLeague', {layout: 'register.handlebars'});
});
```

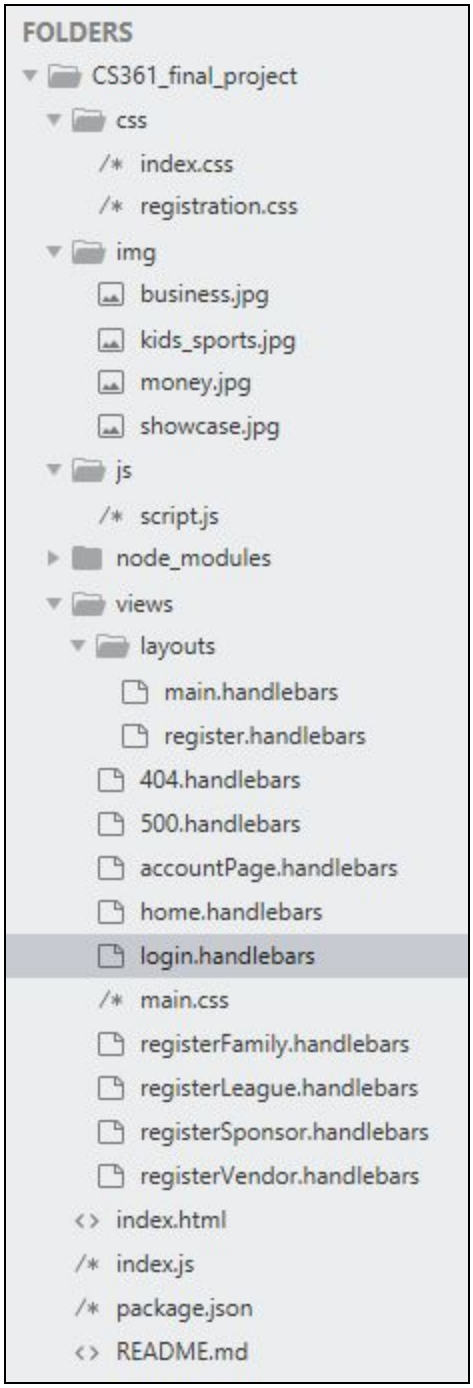
As stated, the code shows the templating engine at work. Please note the similarity in the structure of the Code in the 2 examples highlighted in blue. The first set of code is from the Family page; the second from the League page.

Example 2:

```
<h1 class="pageTitle">Family Registration</h1>
<div id="parent">
   <form id="familyForm" action="/family" method="post">
         <h3>Account Details</h3>
         <label>Username:</label> <input type="text" name="username" placeholder="username"><br>
         <label>Password:</label> <input type="password" name="password" id="password"
placeholder="password" onkeyup='check();'><br>

<h1 class="pageTitle">League Registration</h1>
<div id="parent">
   <form id="familyForm" action="/league" method="post">
         <h3>Account Details</h3>
         <label>Username:</label> <input type="text" name="user_name" placeholder="username"><br>
         <label>Password:</label> <input type="password" name="password" id="password"
placeholder="password" onkeyup='check();'><br>
```

APPENDIX J: Folder Structure



Appendix K: Burn Down Diagram