Big Sky Shark Hunt Web Application

Authors:

Spencer DeBuf

Brady Cornett

(Group 14)

Project Link:

www.bigskysharkhunt.com

Objective:

The objective of this project was to create a website to aid in the organization of the Big Sky Shark Hunt pool tournament. Previously, all matches had to be registered manually into an Excel file which made management a nightmare. By using this web application, there could be a drastic change in the amount of time needed to manage the league while also providing the players access to data about themselves and other players within the league.

Summary:

The web application we created consists of several PHP pages that query a MySQL database hosted on the web server. We use HTML form tags to create an easy to use solution for logging matches to the database via PHP. We also have JavaScript functionality for form validation and some other small UI additions. The HTML form submits a POST request which the PHP files then read, parse, and submit as a query to the SQL database. The PHP API has several variants of the way it sorts and displays the data, based on the specific request.

Brady Member Page (Front End):

I contributed to the front end development of the Big Sky Shark Hunt project for the Web Development class. Spencer set up the foundation of the database and also created the PHP page setup he wanted to use. From there, I customized how the pages looked and interacted by using HTML and CSS. While sprucing up how the application looked, we decided that it wouldn't make much sense to have it specifically designed for laptop/PC use and would be better suited for mobile. With this in mind, I accomplished this task by assuring media queries/breakpoints were being used within the CSS file to scale the website appropriately for mobile devices. I used a minimal amount of JavaScript to have some UI functionality.

I used one basic styles.css file that would span across all of the files in the project to set a standard for the design of each page. I then created a specific style sheet for each individual page to style the specific elements that were contained within that page. I wanted to take a RWD (Responsive Web Design) approach to the layout so that the page was served to scale appropriately efficiently, but settled with the adaptive design strategy using breakpoints and media queries.

Spencer DeBuf Member Page (Back End):

I used HTML5 form inputs to gather the info from the user. Javascript validates all input data before before submitting anything to the php back end. Simple alerts notify the user if they have erroneously selected the same player for both players, didn't have anyone set as winning the match, or have both players set as winning the match. The design of the inputs ensures that null values can never be submitted (every input has a default value).

Once data is submitted from the html form to the php file, the data is parsed from the POST request, organized into appropriate variables, and added as rows into the tables of the SQL database. It is worth noting that I chose to store some common values such as current points (which could be calculated when needed from the standings) in the tables, and update those values for every insertion. I did this to reduce the load on the database for a situation where there are hundreds of matches logged, and a user would like to see overall standings (where every player's score would have to be calculated from scratch). This required extra care to ensure that the attributes are always correctly calculated and stored, with the benefit of extremely cheap queries.

I used three tables in the database: matches, players, and locations. The number of rows in players and locations theoretically should remain static throughout the season (the exception being the case where players and/or locations are added/removed during

the season). The matches table consist of the following columns: match_id (primary key), p1_id(foreign key), p2_id (foreign key), p1_points_wagered, p2_points_wagered, p1_games_needed, p2_games_needed, p1_games_won, p2_games_won, p1_ero, p2_ero, date_and_time, location_played (foreign key).

I also created an iOS application to work alongside the web application for players who preferred a more user-friendly and native way to interact with the league. The app has all of the same functionality for registering matches, looking at match history, and looking at the current standings. As added functionality, the user can choose a player and view all matches that the chosen player has played. The app relies on the online database and php API on www.sharkhunt.com. This is important because if the app stored its own data, there could potentially be issues where the app's model and the website's model are out of sync. This architecture (where both the app and the website draw from and add to the same datasource) greatly simplifies the design.

Conclusion:

In conclusion, I learned a ton from this assignment! I had to research the different types of form inputs in html (and html5), learned all about DOM manipulation in javascript, form validation in javascript, and submitting POST requests. On the PHP side, I learned how to parse POST requests, generate json output (for the iOS app to read/parse), and query SQL databases. On the SQL side, I learned how to create

databases, add users and user privileges, export the database to then import into another host, query the database using php, and organize a database which has a table containing primary keys to two other tables. On the iOS side of things, I learned how to generate POST requests that the app will send to the url, generate json data to send to the url, parse JSON data from a url, Table Views, Tab Bars, Segues, Pickers, etc. Looking back on the project, I think I could have made the php API a bit more organized and user friendly. As it is now, it is fully functional, but not the cleanest of designs to try to update. The next steps in the continued development of this project will include a cleaned up and refactored php API.

References:

Assignment 7, Demolamp

https://w3schools.com

developer.apple.com

raywenderlich.com

docs.swift.org

codewithchris.com

appcoda.com

Learn.zybooks.com

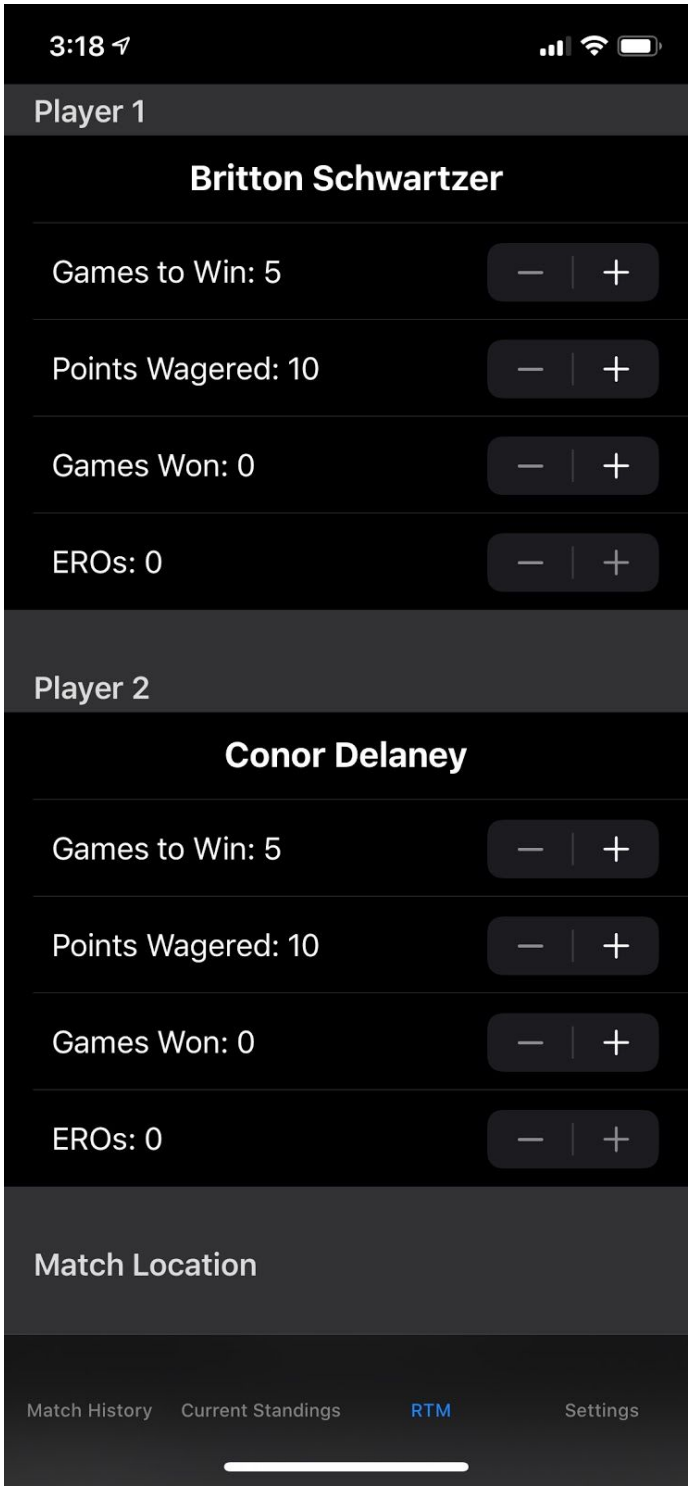
Screenshots:

Dave Hebert +25: 7/7 ERO: 3
Davey Labrie -25: 5/9 ERO: 1

Leo Hatch +20: 6/6 ERO: 1
Tyler Blair -20: 3/5

Britton Schwartzer +25: 9/9 ERO: 3
Davey Labrie -25: 5/9 ERO: 1

Alex Peterson +15: 6/6 ERO: 1
Andy Feistner -15: 3/5


3:18
Player 1
Britton Schwartzer
Games to Win: 5
Points Wagered: 10
Games Won: 0
EROs: 0
Player 2
Conor Delaney
Games to Win: 5
Points Wagered: 10
Games Won: 0
EROs: 0
Match Location
Match History
Current Standings
RTM
Settings

1 275 Britton Schwartzer
Matches: 100.0% (1/1)
Games: 64.3% (9/14) - ERO: 3

2 275 Dave Hebert
Matches: 100.0% (1/1)
Games: 58.3% (7/12) - ERO: 3

3 270 Leo Hatch
Matches: 100.0% (1/1)
Games: 66.7% (6/9) - ERO: 1

4 265 Alex Peterson
Matches: 100.0% (1/1)
Games: 66.7% (6/9) - ERO: 1

5 250 Tyler Hoopes
Matches: NA% (0/0)
Games: NA% (0/0) - ERO: 0

6 250 Cameron Lucero
Matches: NA% (0/0)
Games: NA% (0/0) - ERO: 0