**Software Implementation and Testing Document**

**For**

**Group 8**

Version 1.0

**Authors**:

Megan Achurra

Dylan Giesler

Daniel Lukish

Miguel Perez Ojito

Thomas Rudolph

# Programming Languages

PHP: PHP, along with MySql, allows us to add a ton of functionality while being secure. Using this as a combo gives use the ability to store and use any data we choose. It will give us the ability to login, signup, create leagues, store login attempts, and even create a custom chat. PHP also gives use the ability to create a session, thus keeping us logged in from page to page. Also, because it is a server side language, the code is run before it is sent to the client. This means the users can’t see the PHP code, adding an extra layer of security.

*HTML/CSS*: These are being used for designing the different sites and layouts. So they determine what the colors are, what font we use, and what the overall site looks like.

*Javascript*: Being used sparingly and only for some basic scripting, like on the home page where you can select between the standings and your current league scores. The main reason for the use of Javascript is due to how closely integrated Javascript is with any modern site.

# Platforms, APIs, Databases, and other technologies used

MYSQL: Using MYSQL was a complete freebie for us. The fundamental reasons for choosing MYSQL for our database are: Two of us just took the course this past summer and have it fresh in our minds, and PHP works perfectly with MYSQL out of the box so there is little to no configuration that we needed to do on our end.

API: Sports Open Data API

This free RESTful API (hosted on Mashape) is an impressive service provided by http://sportsopendata.net/. Introduced in early 2016 it is very promising. The data is under a Creative Commons license. Limited to 10,000 requests a month which should be more than enough for our project.

Apache: Apache is a web server that allows us to run PHP. Essentially the request is made by the client, the page then runs the php code, and the results (along with the html/css/javascript) is sent back to the client. Without a web server, only front end languages can be tested (through a web browser).

# Execution-based Functional Testing

Login – I created a session that stores the username that comes from the database. This is easy to test, all I have to do is output the session to make sure that it is giving the right username. I also test the inputs by making sure wrong combinations of usernames and passwords do not grant the session, thus blocking the login.

Signup – I tested the signup by creating random test account and logging into the MySql server to see if they were inputted correctly. I wrote code to block same usernames. I tested this by trying to sign up with the same usernames and also checked it against different Cap letters.

# Execution-based Non-Functional Testing

The primary testing we have done for this section has been loading our website onto different browsers. This just checks to ensure that the HTML features we are using works as intended across the spectrum of browsers*.* After making changes to one of the pages we reload it in a browser to make sure it looks the way we want it to and make adjustments as needed. In addition to that we have also shown the pages to our peers in order to collect feedback about the design, layout, and usability of the pages thus far. We have also tried doing things such as resizing the page, opening the site on different monitors (1920x1080, 3440x1440, 1440x900) to ensure everything still remains in place and looks good.

# Non-Execution-based Testing

We never held any kind of formal code inspection or review, but at different points in our progress we would sit down with one or more other group members and go over what we’ve done and get their feedback. We will likely do more of this in the future as we move towards heavier coding work and less design.