**Software Implementation and Testing Document**

**For**

**Group <8>**

Version 2.0

**Authors**:

Megan Achurra

Dylan Giesler

Daniel Lukish

Miguel A. Perez Ojito

Thomas Rudolph

1. **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 and for the tooltips. The main reason for the use of Javascript is due to how closely integrated Javascript is with any modern site.

Python: Python allows us to call the API easily and update our database to reflect current scores and stats. We can use our Python scripts with Windows Task Scheduler to automatically run the script everyday and update our database.

1. **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 - API used: sportsop-soccer-sports-open-data-v1.p.rapidapi.com

We receive information on each teams: Wins, Losses, Draws, Points, Scores(Goals), Conceded(Goals), and number of matches played. Our API is limited to 100 requests a day, which should be no problem for our work.

1. **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.

League Creation - This was an easy check, I checked the database after requesting a league creation to make sure that it was added correctly with the appropriate captain (current user). I did this multiple times under different test users, making sure you build more than one league and that the captains were correctly set.

Invites - The invites use a random char, put in an 8 size array. It uses the combination of this and the league name to create an invite link. Each link is different for every user. I checked multiple links to verify that they worked correctly. I also checked to make an error was thrown when a used invitation was attempted to be used a second time. If a user is not logged in when the link it click, it should forward user to login, without using up the invite. I tried this multiple times to make sure the forwarding worked. The last case I built in was to check what happens if you get a second invite to the same league, which it does not allow.

1. **Execution-based Non-Functional Testing**

Hashing - After adding the hashing in incrementation 1 I had to test it by retesting the login and signup. Because all passwords are hashed after this point, I had to empty the table, then all the users have to re-sign up. Essentially if the correct username and password let you signup and login, it was working. I also checked it with wrong users and passwords, including different caps. Note: Because it is hashing, there is always a VERY small chance the right username and password combination could allow a false login to occur.

Loading the website on computers with different resolutions to make sure things don’t break at different sizes. We did this by resizing the page to extremes and used different monitor resolutions (1920x1080, 3440x1440, 1440x900) so that we could make sure everything stayed where it was supposed to be no matter what screen you were on.

1. **Non-Execution-based Testing**

While again we didn’t do any specific code review or inspection, we would regularly meet in Dirac to show each other what we had done and to get feedback on progress. And still sticking to our promise of maintaining style consistency when it came to actually coding anything, this has proven to be a tremendous benefit for when someone else needs help it is easy to see what the intent behind a piece of code is.