Justin Chen, Daniel Schnoll

Introduction to ITWS

**The Proposal**

Currently, IMLeagues is the go-to sports application for intramural sports. It’s purpose is to provide statistics and scheduling for club athletics, and provides detailed insight into personal and team performance. It is used at many schools, including RPI, as well as fraternities and sororities. However, the IMLeagues website is extremely flawed: It is littered with ads, has an old, unathestic interface, and at times it is very unresponsive. The majority of users have a severe distaste for the service, and is proven via the scathing reviews on Google Play and the App Store. However, people are forced to use it because there are no competitors with similar functionality. Our solution to this is to make a new service named, Swish, that will rival IMLeagues’ functionality, but without all of the clutter.

**The User Interface**

Our goal when designing the user interface was to be intuitive, and simultaneously, aesthetically pleasing. In order for the website to be intuitive, navigating the website should be easy to understand. We achieved this with a tabbed header and a user interface that implements Materialize CSS cards, which color code and separate the information being displayed. There are only 3 tabs in the header, one indicating the home page, one indicating the teams page, and one indicating the logout action. This allows the user to understand what will be within each page upon clicking the tab. After reaching a page, the cards are displayed with the user-appropriate information. Each card has a title, indicating what information is on the card, and since each card is color coded, it’s easy to tell how the webpage is divided up.

**The Features**

Swish is still in its infancy, so functionality wise it’s pretty limited in comparison to IMLeagues. However, it has the core features that make it a useful tool. It has a login page where you have the option to sign into your Swish account, or if you don’t have an account, you can create a new one. Creating an account is extremely simple, as all you need to do is supply an email, username, and password. Upon a successful login, you are redirected to the homepage dashboard. This dashboard contains the most important parts of your profile, which are the teams you are a part of, all of your upcoming events, and your personal statistics such as your win loss ratio and total games played. All of these are divided up with the cards user interface, making navigation through these features extremely easy.

Clicking on the teams tab brings you to the teams page, and this page allows for simple team functions. You can create a new team through this page, assuming that it has a unique name that isn’t already in the database. Upon creating a team, you automatically join it. You can also join a team through this page, as long as you know the name of the team you want to join. The last block of this page is the “Create an Event” card, which allows you to schedule events between two teams. Inputs include the home team, an away team, as well as the location and time of the match.

**The Implementation (Focus 1)**

Our primary area of focus was Area 4, which is pulling real data from a database. The importance of this area is clear, due to the fact that our project is built around recording and displaying information about your profile. We went with a LAMP (Linux, Apache, MySQL, PHP) stack for this project, so this area of focus was done via PHP and MySQL. The login page utilizes data pulling, as logins are done via inputting an email and password. Upon submission of the form, the site runs a query on the database, finding the entry where the email matches. This also guarantees one entry returned, as the system only allows for an email to appear in the database once. We then hashed the inputted password and compared it to the hashed password in the database. If both of these conditions are met, the user is then logged in and taken to the dashboard.

Upon reaching the homepage, all of the information regarding the user’s teams, upcoming events, and personal statistics, is displayed. The data is all stored in a MySQL database, which also falls under the data pulling umbrella. We came across a challenge regarding the teams that a user is part of, since SQL databases can’t hold lists as variables. As a workaround, we chose to include a relational table that contains all the relationships between users and teams. We would display the list of teams by doing a query of the relationship table where the user ID was equivalent to the current user. This returned a set of database rows that each contain a unique team ID. Displaying the teams was then done by using a query to pull the team from the teams table based on the ID. Once getting the team, we displayed all of its variables, which included their name, wins, and losses.

Displaying the upcoming events was considerably more complicated. Events are recorded in the SQL table with a datetime variable, so we only took a query of events that take place in the future. This ensured past events would not be displayed. The events table contained the IDs of the participating teams, so in order to determine whether our user was part of the event, we had to pull their teams from the relationship table again and compare it to the IDs in the event. If they were part of a team that was in the event, that event was displayed. A home and away team is shown, using bolded and italicized text styling if the user is part of said team, and the location and date of the event is also displayed.

Finally, we realized that we could compute personal statistics by adding up all of the wins and losses of the individual teams. As a result, this saved us some unnecessary table space and calculation complexity, as personal statistics are never individually recorded.

**The Implementation (Focus 2)**

Our secondary area of focus was Area 1, which was usage of HTML and CSS for page layout and design. We decided to go with the Materialize CSS library, which is based off of Google’s Material Design. We chose this library because we agreed that this flat and minimalistic look was perfect for creating a visually pleasing modern design. The majority of the CSS came from the provided library, although we did have our own CSS when fine tuning the look of the website, such as adjusting font sizes and adding padding for various aspects of the site.

**The Challenges**

The biggest challenge by far was getting used to using PHP with the MySQL database. Since PHP and MySQL are the two most recent things we learned in class, we didn’t have a lot of experience when dealing with it. As such, we had to figure out a lot of things on our own and go through much trial and error since we were not used to the language yet. The code from Lab 10 was hugely beneficial, as was Google and Stackoverflow. Both of us are programmers, so the logic behind most of the programming wasn’t too hard, it was mostly the syntax that tripped us up. One extremely frustrating error we had was that SQL frequently uses the ` character, which looks very similar to the ‘ character, especially on the text editor we were using. We would repeatedly run the code through an SQL validator, incapable of understanding why two seemingly identical lines resulted in only one having an error. We eventually figured it out, but having such a trivial error give us problems for hours was pretty disheartening.

Another challenge we had was that our third group member dropped the class a couple months into the semester. Because of this, we operated as a two person group. Being down a person was definitely an impact, as each one of us had significantly more responsibilities, but we persevered.

**The Future**

Like we said, there is a huge market for Swish, as demonstrated by the popularity of IMLeagues and the overall negative perception of IMLeagues. If people saw that they had a superior alternative to IMLeagues, they could be inclined to switch over. Obviously, Swish is not fully complete yet, and it will require quite a bit of work to make it a competitor to IMLeagues functionality wise, but with the right tools and proper development, we believe that Swish could turn into a web application used by college campuses across the country. If we choose to monetize Swish in the future, we could do so via either carefully placed advertisements, or via a licensing/annual subscription service where we would sell a campus wide license to a school.