CPSC 490 Final Project Report: Platform for Custom Fantasy Soccer League Management

Edward Kimo Hon
Spring 2015

Part I

Abstract

Fantasy sports are an increasing popular hobby for millions of fans worldwide. Fans create their own teams of real life sports stars, accumulating points when players do well in real life games. Though an official league Fantasy football is of course the most ubiquitous game in the genre, but official leagues exist for many sports. By far and large these leagues are only the most popular - there is a dearth of options for fans of lesser known leagues.

In this project I developed Fantasy Sunday League (FSL), which provides a platform and framework for dedicated fans to create, manager, and play with custom leagues. The goal was to create an easy to use web app that would be an improvement on the logistical challenges of coordinating a custom league via spreadsheets, which is the dominant method currently. FSL first and foremost allows creation of these leagues, which are then populated with user supplied .csv data for both available players for selection and league schedule. Users then build their teams by selecting a squad of players, which can be edited by making transfers due to player performance, preference, etc. League administrators report results from the games in a week, noting goalscorers, etc. which in turn leads to teams scoring points. A leaderboard for each league tracks the competition and allows users to see how they are stacking up against each other.

This app was written using Ruby on Rails, with a PostgreSQL database in production. The basic functionalities are mostly complete, but there are some additional key features which need to be introduced before it can stand as a good alternative to playing with more established and official leagues. There are many additional areas of improvement that can and will be expanded upon after beyond the CS490 deadline.

Part II

Features

All of the basic functionalities that are required for a fantasy soccer league were implemented. From my expectations in the proposal,
• Setup - Users create an account and either join an existing league or create one of their own. The creator of the league is the sole administrator. Administrators have the power to add real-life players to the league, add games to the fixture list, and distribute an invite code for the league.

![Figure 1: Administrative Options for a League and Player Index](image)

• Team Building - Users can scroll through a list of players sorted by position and add players to their team to begin the season. There are restrictions on how many players must be added to the team (ie: 2 Goalkeepers, 5 Defenders, 5 Midfielders, 3 Forwards). 1 of each position is benched at all times, making players use the 4-4-2 formation. There are no limitations on how many players you can select from an individual club.

• Transfers - Users have a text based interface for adding and removing players, where transfers are queued up before being confirmed. There are no prices in transfers, and no penalties for multiple transfers. Transfers are locked for a gameweek once results begin being reported.
Figure 2: Transfer Interface

- Scoring - At the individual team level, scores for each player are reported next to their names in the squad layout view. In addition, on the league level, a league table displays user and team’s name and in the other the number of points scored overall. Individual gameweek scores are not tracked other than the most recent gameweek.

- Reporting results - Administrators have an interface to report results. Goalscorers are added, followed by assists. No support for disciplinary measures has been added at this time.
Part III
Implementaion & Data Models

The data model supporting this app is a fairly complicated but intuitive hierarchy. Leagues are comprised of Users and possess Fixtures, which record information about individual matches. Users have one team, which owns players through two relationship tables. Ownerships describe players that are actually on a team, whereas queues are used to facilitate the transfer methods.

Figure 3: Match Report Interface

Figure 4: Model Structure of Databases
1 Leagues

Leagues are the highest class in this model. Methods within the league show upcoming fixtures, match reports, and overall scoreboard for the league. Administrators have access to special methods to add critical data.

1.1 Fixtures

Fixtures record the game result along with goal scorers and assisters. The league extrapolates game results from information stored in fixtures, using this data to assign points to players. Each set of fixtures corresponds to a gameweek, which may or may not be a week. The ability to advance gameweeks was given to administrators to accommodate a variety of schedules that may occur, especially in an amateur league.

2 Users

Creation of a secure user database and account creation process was very important. Cookies allowed for persistent sessions, with hash digests being stored in the database much in the same way as passwords were implemented and kept. Not much was done with users beyond creation, as most of the important information takes place on the team level rather than the user. Administrators are a special class of user that have the responsibility of uploading players, fixtures, and match reports, along with advancing gameweeks.

3 Teams

Teams are comprised of players. The most important methods for the team are adding and removing players via the transfers interface. Teams also hold a user's overall and weekly score.

3.1 Relationships

Because each team owns many players, and each player is owned by many teams, two relationship tables were created to manage the team-player interaction. Ownership tables are updated after queue tables are processed in the make transfer action.

4 Players

Players are the smallest unit, and the entire reason for the structure. Players hold information about their names, position, club, and scoring history. Teams use this information to build a user's scores and determine where they rank in the league.
Part IV

Difficulties and Future Directions

This was my first time using Rails to build anything, so that came with a lot of challenges. A lot of time was spent just learning how both Ruby and Rails worked so that I had the knowledge base to accomplish what I wanted. Another major difficulty was related to unfamiliarity with the way Rails works. Many of my models and controllers grew rapidly and were difficult to reign in because I let the structure develop organically step by step. A more clear vision of how all the models would interact with each other would have really made my life easier in the long-term, though painful in the short-term. In addition, a lot of exceptional behavior is not being caught right now, again due to my unfamiliarity with Rails. Things work perfectly if users are perfect and don’t stray far off the signposted paths - but of course users are the developer’s worst nightmare.

Future directions were outlined in my project proposal, and they still apply because I only met my minimum expectations. This is mainly because I did not have a good understanding of how difficult it would be to debug and create something in a whole new framework! The most obvious improvement is allowing a crowdsourced match reporting tool. Users could report on matches and administrators could select accurate match reports, rather than having to fill each one out on their own. This could be especially helpful when more statistics and ways of gaining / losing points are available in the match reports. The transfer system is also a big area for improvement. I initially set out to emulate the official Fantasy Premier League model of players having prices for transfers, but realized this was difficult with the scale of most custom leagues being no more than a dozen. It is definitely a space that could use design improvements in the way it functions, as well as how it looks.