17/02/2020
Nuriel Leve, Amanda Patterson, Dominic Smorra, Lucy Lawrence

Introduction

Description

For this project we're targeting users who are interested in playing Fantasy Soccer within the Premier League. By interacting with the command-line interface, this user will be able to register in an open league or create a new one. Once the league is closed, in the 24 hour period after, there will be a draft. Users will form a team, or teams, of players composed of one goalkeeper, four defenders, four midfielders, and two attackers. They may choose players from any team within the Premier league; however, if there are multiple teams, a player may only be on one team at a time. The order teams may draft players in is set randomly, or can be chosen explicitly by the league manager. After the draft, they can then schedule games between members of that league. It is only the 'league manager,' or user who created the league, who is able to close the league and schedule the games between teams. This user will also have the option to opt for a randomized schedule, which would pair teams at random. Matches will be 'played' each Sunday according to the created schedule. A team will win the match if their total score is higher than their opponents. A score is tallied by adding up the points each player on their team has earned. In our system, player statistics will be provided manually by the league manager. Players are either on a team or in the pool of free players, and for 24 hours after the last game is played on Sunday, users can swap their players for players in the pool to improve their team. They can also ask for trades with other teams. When a team receives a trade request they can accept, deny, or send an alternative request. Users will also be able to see the current ranking of the teams in their league.

We got inspiration from https://fantasy.premierleague.com/. We also looked at https://en.wikipedia.org/wiki/Fantasy_football_(association) for background knowledge.

Features

Functional Features

When defining the functional requirements we attempted to be as complete as possible. We went through all possible actions every type of user should be able to complete. We then brainstormed the other aspects the system must have in order for those actions to be completed.

ID	Short name	Description
F1	League Commands	The user can create and open/close the league to teams, set the schedule, and add statistics by issuing command-line commands with the following syntax: **Note that these are only open to the league manager** `command-name [target-objects]*`. The available `command-names` are the following: - *create: ...* - *close: ...* - *open: ...* - *setDraftOrder: ...* - *setSchedule: ...* - *randomSchedule: ...* - *addStatistics: ...*
F2	Team Commands	The user can create a team, and add and remove players to their team by issuing command-line commands following this syntax: `command-name [target-objects]*`. The available `command-names` are the following: - *create: ...* - *trade: ...* - *draft*: ... - *drop*: ...
F3	Statistic Viewing Commands	It is highly important that everyone be able to see the statistics of the teams and rankings. We will have multiple ways of viewing stats. - *viewRanking* - This gives the overall ranking of the teams in the league.

		- *viewSchedule* - This gives the regular season schedule for the rest of the season. Teams must know who is playing who. - *viewOpenPlayers* - This gives a list of all players currently undrafted - *viewTeamPlayers [team name]** - This will print all players by team. If given a team name it will only print out the roster of that one team. - *getTeam [player name]* - This will return the team name that the specified player is on, or 'free' if the player is undrafted. - *getRecord [team name] [player name]** - This will return the record of a given team: who they played, the points they earned and whether it was a win or lose. If given a player name as well, the stats for that particular player are printed per game.
F4	Scoring	The points should be tallied every Sunday after the last game (and after the league manager inputs players statistics) and added via the scoring system to create individual team scores.
F5	Manage Ranking	The points for each team will be listed and parsed through to create the ranking of the week, updating after every game (on Sunday).
F6	Permission Status	Each team will have a unique id, their team name, but the league manager will have an additional password that they must use to access league commands. The league manager should be the only user allowed to make changes to their league.
F7	Player Changes	The system should not allow any changes to teams except for the 24 hour window after the last game on Sunday.

F8	Team Management	The system will only keep track of players and their positions, only allowing teams of one goalkeeper, four defenders, four midfielders, and two attackers, and ensuring players are only on one team at a time.
F9	Player Trades	When a trade is sent from team to another in the form of the command: *trade [sending team] [receiving team] [sending player] [receiving player]* The receiving team may accept, *y*, deny, *n,* or deny and send an alternative trade in the same form.
F10	Drafting	The system will automatically set a random order that drafting may take place in. Players are picked in a turn by turn basis and the order reverses each round. Only the team whose turn it is may draft a player. However, the league manager may override this with the *setDraftOrder* command found above.

Quality Requirements

ID	Short name	Quality attribute	Description
QR1	Commands sanity checks	Reliability	When the player issues a command, the syntax of the command shall always get validated against the format specified in F1,F2, and F3.

QR2	Extensible Team	Maintainability	The fantasy game shall be easily extendable in terms of drafting trading, calculating scores for players.
QR3	Instantaneous results	Responsiveness	Once the scores of all soccer players are provided by the user, the results of the virtual matches shall be available within one second.
QR4	Easy to use	Usability	The command-line interface commands are easy to understand and intuitive for players to use.
QR5	Extensible Functions	Maintainability	The fantasy game shall be easily extendable in terms of what users can do, for example creating new leagues at any point.
QR6	Seamless functionality	Availability	The system will not crash, such that the game can be played seamlessly by the user.
QR7	Information security	Security	Any information provided by users can not be accessed by other users.
QR8	State saving	Reliability	The state of the game will be maintained when the program is exited, such that the user can return to their game later.

Java Libraries

Name (with link)	Description
opencsv	Csv parser that we will use for parsing a csv file of players in the premier league. We chose this among others because we wanted a very straightforward and simple parser. This seems to be a good balance between fulfilling all of our requirements and not being exceedingly complicated.
JUnit	We will use it for testing various components of the application to ensure everything works smoothly. We chose this as it is a common testing framework. Some team members already have familiarity with this library, which will help as we aim to test rapidly and efficiently. We know the importance of testing and for this reason chose to include a framework like this one.
Time4J	We will use it for keeping track of real time throughout the season. Scheduling is very important in fantasy soccer. Drafts occur for only 24 hours, points are calculated every Sunday, and trading can only happen in a set period of time. For all of these reasons we knew we needed a way of keeping real time in our system. This seems to be a robust, but straightforward library, and as such, is ideal for our system.
ASCII Table	This may be helpful if we want to draw a table of the schedule of games. Fantasy Soccer is extremely statistically heavy. We want to give users the best experience possible, while still using a command line interface. We believe that utilizing this library may help us convey statistics to users in a readable manner.
args4j	This may help us parse command line arguments. As we are using a command-line interface, our program will clearly utilize many command line arguments. As with many of the other libraries we attempted to find a good balance between robustness and simplicity. We are open to the fact that as our project progresses we may need alternative command line

	processing power and may add additional libraries. However from where we stand now we believe this will be a suitable choice.