Application Description

For our Course Project, we created a Fantasy Ultimate Frisbee application, modelled off of the extremely popular ESPN Fantasy Football platform. Four of our five team members play on Duke's Ultimate Frisbee team, so the motivation for the project came naturally to us. Believe it or not, the American Ultimate Disc League (AUDL) is a fully functioning professional Ultimate Frisbee league which launched in 2012 and has been growing ever since. We pulled the data needed for our application from UltiAnalytics, a website where the AUDL stat-keepers upload data from each game.

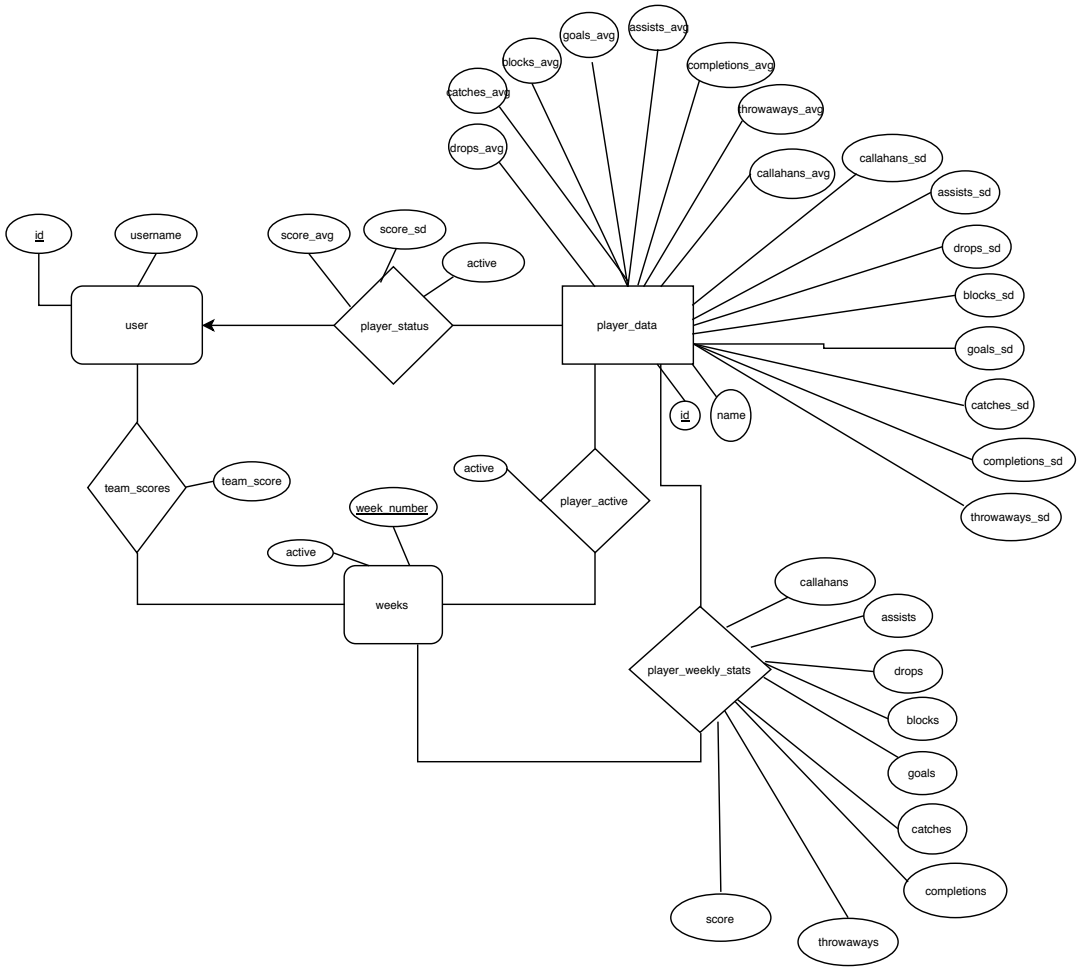
Our application allows users to create their own teams, drafting real players and setting their rosters each week. Players score points based off of various statistics (goals, assists, blocks, catches, etc.), and the objective is to finish with the highest cumulative score across all active players during the 16-week season.

The season begins with users creating accounts in the application system, which prevents one user from maliciously (or accidentally) updating another user's roster. Before Week 1, a draft occurs, where all of the users pick players for their teams. There are no constraints on the number of users, draft order, or roster size, so users can decide these variables as they see fit. The only constraint is that once a player is picked, he or she is off limits to the other users.

Once the draft is complete, users manually iterate through each week. Each iteration runs a SQL query that selects the current week's scores for each active player and aggregates them by team. Users can click on a given week to see player statistics from that week (ordered by player score), as well as each team's results (the sum of the team's active player scores). Users can view other users' rosters and those players' statistics, but they can only modify their own rosters. Another built-in constraint is that only 7 players can be active at a time, which means users must choose wisely about who to start or put on their bench. Users can also view the leaderboard, which shows the cumulative score for each team, summed from the start of the season to the current week. At the end of Week 16, the season ends, and the leaderboard becomes the final standings. At this point, users can choose to run another season (re-drafting teams but maintaining their accounts), or reset everything, deleting all of the teams and their associated accounts.

Note that the crux of this application lies in scraping real-world data from UltiAnalytics, based off of the most recent (2018) AUDL season. This data is analyzed in Python, and summary statistics (averages and standard deviations) for each player's stats (goals, assists, blocks, etc.) are outputted. At the beginning of the season, SQL queries are run that randomly sample from these distributions to simulate week-by-week performances. The idea is that player performance is quite variable, similar to how athletes perform in the real world. All of the sampled data is generated at the start of the draft, which means that the results of the season are immediately known to the database under the hood. However, weekly data is only revealed to the users as they move through the season, which gives the impression of a real-time season unfolding.

E/R Diagram


goals_avg
assists_avg
blocks_avg
completions_avg
catches_avg
throwaways_avg
drops_avg
callahans_avg
callahans_sd
assists_sd
id
username
score_avg
score_sd
active
drops_sd
blocks_sd
user
player_status
player_data
goals_sd
catches_sd
id
name
completions_sd
team_scores
team_score
active
player_active
throwaways_sd
week_number
active
weeks
callahans
assists
player_weekly_stats
drops
blocks
goals
catches
completions
score
throwaways

Assumptions

There are a few important assumptions we made about our data, which are outlined below:

- We assume that individual statistics from UltiAnalytics are mostly correct. It's entirely possible that the stat-takers made many mistakes while recording data. However, it's not crucial that every single piece of raw player data is perfect: the goal is to get a good approximation for how each player performs on average across the course of the season, from which we can randomly sample our own simulation data.

- We assume that each player in the UltiAnalytics website can be mapped to a player on the team's roster. Unfortunately, some players are given nicknames by the stat-takers. For example, Dallas's Brandon Malecek is ubiquitously known to fans as "Muffin." Some of these instances are well known to us (Muffin is one of the top players in the country), so those nicknames were able to be modified manually, but there were many mysterious nicknames and that didn't appear on rosters which we weren't able to correct.

- We assume that each player has his or her own unique set of raw statistics. For example, we assume that a player from Atlanta named Parker Bray won't have his stats recorded under "Parker Bray" during some games, and "P. Bray" during others. Unfortunately, while sifting through the raw data, we did notice several instances of this, causing our database to think one player is actually two, hurting that player's average weekly performance.

- Finally, we assume that individual players' statistics are accurately represented by a normal distribution. It's entirely possible that statistics tend to fit Poisson, binomial, or exponential distributions better than normal distributions, or even that different players' data fit different types of distributions. For the sake of simplicity, we decided to model performances as normal distributions, because a) the mean and standard deviation of a data set are easy to calculate in Python, and b) normal distributions are hopefully intuitive to most users, who can use the mean score (a general prediction about how well a player performs week-to-week) and the score standard deviation (a measure of player consistency) to evaluate which of their players to make active.

Database Tables

- Name: **player_data**
- Attributes: *id, name, goals_avg, goals_sd, assists_avg, assists_sd, blocks_avg, blocks_sd, catches_avg, catches_sd, completions_avg, completions_sd, throwaways_avg, throwaways_sd, drops_avg, drops_sd, callahans_avg, callahans_sd*
- Description: This table contains statistics from the 2018 AUDL season. All of the raw statistics from CSV files provided by UltiAnalytics were scraped and analyzed in Python. For each player, an ID was created (there are a few players in the league with identical names), and the average and standard deviation for each statistic (goals, assists, blocks, etc.) were calculated. This player_data table is loaded into the database as the source for simulating different players' performances across the season.

- Name: **player_weekly_stats**
- Attributes: *player_id, week, goals, assists, blocks, catches, completions, throwaways, drops, callahans, score*
- Description: This table contains randomly generated statistics for each player during each week of the season (Weeks 1-16). A given week's statistics were calculated using the Box-Muller transform (which converts a uniformly distributed random variable into a normally distributed one). The uniform variable came from SQL's **random()** function, and the mean and standard deviation for the normal distribution came from **player_data**. Each value was rounded to the nearest integer (3.42 goals is in a week is unrealistic), and negative values were converted to 0 (-4 drops is unrealistic as well).

- Name: **player_status**
- Attributes: *player_id, name, score_avg, score_sd, user_id, active*
- Description: This table keeps track of all the players selected during the draft, their average score and its standard deviation, the user that owns them, and whether or not the player is currently active. The purpose of this table is that it allows a user to click another user's profile (or their own) and view all of that user's players, how much they score on average, how consistent they are, and whether or not they are currently active.

- Name: **player_active**
- Attributes: *player_id, week, active*
- Description: This table keeps track of all of the players and whether or not they were active in a given week. In other words, it's the historical record of the *active* attribute from **player_status** (which shows whether a player is currently active or not). This allows users to click on a specific week of the season and see what points were scored that week and by whom. Only active players score points, so the *active* attribute is crucial when querying to calculate player scores during a given week in the season.

- Name: **user**
- Attributes: *id, username*
- Description: This table stores all of the users currently playing together in the app, and their ids. Theoretically, two users could create the same username, so the *id* attribute unambiguously differentiates between users.

- Name: **team_scores**
- Attributes: *user_id, week, team_score*
- Description: This table tracks the total score for each team (user) across the season. It allows users to click on a specific week and see how many points each of their competitors' teams scored.

- Name: **weeks**
- Attributes: *week_number*
- Description: This table stores all of the weeks in the season (Weeks 1-16).

- Name: **week**
- Attributes: *week_number*
- Description: This table acts as a global variable, storing the current week of the season (Weeks 1-16). Each time the season iterates to the next week, the single tuple in this table is incremented.

- Name: **settings**
- Attributes: *active*
- Description: This table acts as another global variable, simply storing the number 0 or 1, to indicate whether or not the season has begun yet. After the initial draft is completed and Week 1 begins, the *active* attribute is changed from 0 to 1.