# Pocket Fantasy Football Analytics

CS 411 Project Proposal - Fall 2022

Team # 032 - MangoDB

Adish Patil, Sid Karnam, Rohit Chalamala, Neehar Sawant

TA: Lu, Yicheng

## 1. Original Project Summary

Fantasy Football is a game that allows users to be the owner, GM, and coach of your very own football team. Users draft a team made up of real NFL players; based on their on-field performance in a given game, users score points. Points are awarded to users based on various events in a game for a player: amount of rushing/receiving yards, touchdowns or field-goals scored, defensive turnovers and so forth. The flexibility of a user's lineup makes fantasy sports so captivating. Fantasy Football allows users to trade players with one another, and also pick up free-agents of the waivers (players that don't belong to any team). With the game of football being so violent, these tools are crucial when a users team suffers injuries or aren't performing up to standard. Our project aims to help fantasy users take advantage of players of this fluid system, and make the best player choices for their teams.

Pocket Fantasy Football is a web application that aims to aid users' fantasy football team performance with personalized recommendations. Users can enter their current teams, receive guidance on their strengths/weaknesses with positional recommendations, and also view key statistics and other information relevant to helping users achieve their goals.

## 2. Project Direction

**Please list out changes in directions of your project if the final project is different from your original proposal (based on your stage 1 proposal submission).**

We never used the NFL Prospects table. This was due to current fantasy football not taking into account college prospects. Thus, for that reason it wouldn't really make much sense for our application to take these players into account. We did have numerical ratings for current user teams and gave recommendations for new additions based on these rankings, but we did mention we would give key information and stats about these players through means of a drop down, but we decided against it as we thought it would be too much to output to the screen. We also didn't include the player of the week for our creative component, but instead pivoted to a hometown best players section which features the best players from the hometown football team the Chicago Bears.

**Discuss if you changed the schema or source of the data for your application**

The schema of our application was largely not changed by the end of the project asides from the extra tables we had made for our positional recommendations and team evaluation features. Our source of data for this project was the Sportsradar NFL v7 API. There was ample documentation from this API, in which we felt comfortable understanding and coding up what was necessary for the application: https://developer.sportradar.com/docs/read/american_football/NFL_v7.

**Discuss what you changed to your ER diagram and/or your table implementations. What are some differences between the original design and the final design? Why? What do you think is a more suitable design?**

As we continued doing the project, we realized that our proposed functionality wasn't going to be met by the original proposed ER diagram and table implementation. Specifically, to implement features such as the positional recommendations or team evaluations, we needed relations that housed this information to be viewed by the front end UI. Hence, we added additional tables such as User_Team_Rating, and positional recommendation tables as part of our stored procedure. This was a more suitable design than our original one because of how these tables were automated by our stored procedures and trigger implementation in Stage 5 of the project.

Additionally, one last change we implemented from our original DDL commands was the variable selection for some of the relation attributes. Each position table (WR, RB, TE…) originally was created with a player_id INT primary key. After further testing with the API, we realized this id was a VARCHAR, and that we needed to add the player names to the table. This would allow us to perform the team rating evaluations triggers and query's a lot more efficiently.

**Discuss what functionalities you added or removed. Why?**

We removed the specific attributes for each recommended player due to too much information needing to be outputted to the screen. Instead, we thought it easier and more useful for the user to simply use individual player stats and calculate a rating, compare this rating to current players on a user's team, and recommend players accordingly. Other than this our other functionality is present on the screen.

**Are there other things that changed comparing the final application with the original proposal?**

We completely changed our UI from the design. We went with a more simple approach for the UI compared to our proposal due to time constraints.

## 3. Project Outcome

**Discuss what you think your application achieved or failed to achieve regarding its usefulness.**

I believe our application did a great job of being able to input the user team, update the team, and search for the team no matter how many users there are based on a username. Additionally, we developed an algorithm to compute a standardized player rating that works across positions. This allows us to compare the players on the team to other players in the league and give recommendations for specific positions which makes it very easy for the user to target other players that can improve their team. Even though we were unable to implement the creative component to have a drop down for the player recommendations, we were able to add the functionality of it.

**Explain how you think your advanced database programs complement your application.**
The advanced database program was perfect for our application as the trigger and stored procedure work together with our algorithm to give the user specific player recommendations. The trigger is called every time the user inputs or updates their team and creates ratings based on their current team. Then, the stored procedure compares the ratings of the player on the user team to other NFL players of the same position.

## 3. Team Reflection

**Each team member should describe one technical challenge that the team encountered. This should be sufficiently detailed such that another future team could use this as helpful advice if they were to start a similar project or where to maintain your project.**

Neehar - One issue that was very challenging was when we were trying to use the output of a query as a parameter of our stored procedure. We later found out that we could not do these one after the other as the queries occur in parallel. Therefore, we would have to run one query, get the output, and run the stored procedure within it to get our correct result.
Rohit - One issue that took us a long time to actually figure out was where to actually put the stored procedure and trigger code so that we could use it in our server.js file. Eventually after many hours and going to office hours, we figured out that we had to go on the GCP SQL shell and then input our procedures and triggers there after selecting our specific database.
Sid - We dealt with the problem of using multiple cursors in one single stored procedure. While this is possible to do, it is difficult to set exit loop variables separately, so what we found was easier was simply using multiple stored procedures in order to store separate positional recommendations.
Adish - One challenge as a team ran into was uploading the SQL database file we created into the GCP SQL instance. The SQL database file we'd created had our relation DDL commands, and hundreds and thousands of INSERTs for each table. When uploading it to the database, we ran into various syntax and relation errors that we didn't know we had. Tracking down these errors was quite a hassle, but made us ensure that our schema was problem free for the later

stages of the project. An example of a bug we ran into was non-existent players and duplicates from our tables.

**Describe the final division of labor and how well you managed teamwork.**

The labor for this project was spread fairly evenly between us. For checkpoint 3, we all worked on developing the database and making sure that everything from our ER diagram was accounted for. When we got to checkpoint 4, we were pretty set to start getting everything implemented on the front end and connecting it to a backend. We followed the workshop video posted on Canvas to do this, and we all worked on it together to make sure everything was done correctly and we didn't make many errors to limit the amount of debugging that we had to do. As for the last checkpoint, we split up the work such that Sid and Adish worked on the stored procedure and trigger code and Neehar and Rohit worked on moving that code into our application so it can actually be used in the backend as well as updating the front end to reflect these changes. As a whole, our team didn't have any issues with communication, division of labor, or teamwork.
.

## 4. Future Work

**Describe future work that you think, other than the interface, that the application can improve on**

As far as future objectives, there are a few different things we could potentially implement such as:

- A user login system so each user can use the tool with their specific team.
- A more interactive and colorful UI with multiple pages versus everything being stored on a single page along with not having everything centered in the page.
- We could add some type of scheduler to fetch from the API so that all the information is updated automatically as opposed to manually by us.
- We could add more statistics so that the user is able to fetch all the data about a player by clicking on them.
- We could do additional research and testing to see if there are better formulas/calculations that can be done to determine the strength of a player/team.