**NFL Database Application**

**Team Members:**

-Chuck Zumbaugh

-James Chapman

-Vishnu Bondalakunta

# Introduction:

The National Football League (NFL) is one of the major professional sports leagues in America and the highest professional level of football in the world. Sports databases, especially those for the NFL, store comprehensive collections of structured data related to various aspects of football. These databases serve multiple purposes, including statistical analysis, historical reference, betting odds calculation, fantasy sports management and more. They are commonly employed in sports apps to provide users with information related to various statistics. For this project, we designed a simple NFL database application that allows users to search for different information pertaining to the NFL.

The NFL Database Application manages and reports information for NFL games during the past decade (2013 - 2023). This system allows the user to easily retrieve information about players, teams, games, and even plays within each game (where available). With over 1.5 million records, users can obtain information down to each play made by a specific player in a game. The application itself is a lightweight, command line program that can be used once a user registers for an account. Target users are statisticians and eager NFL fans who want to have specific NFL game information at their fingertips! In addition to being able to view a data snapshot, users can export the snapshot in a filetype of their choice (ex. markdown or CSV) for their own use.

# SQL Implementation:

## Description:

Users are identified by a uid and have attributes username, password, first name, last name, and created on.

• Users can favorite zero or one team

• Users can favorite zero or one athlete

• Teams will be identified by a team name and have attributes location, abbreviation, primary color, and secondary color.

• Athletes will be identified by an athlete ID and have attributes date of birth, birth city, birth state, first name, last name, height, and weight.

• A player can play for many teams in their career, and a team can have many players.

• Each position will be identified by a position name and have attributes abbreviation and platoon(offense, defense, or special teams).

• A player is contracted to play for a certain team in a specific position, and the contract lasts from the start date to the end date.

• Venues will be identified by a name and have attributes capacity, city, state, grass (boolean), and indoor(boolean).

• A venue can have many home teams (ex. MetLife Stadium), but a team must have exactly one venue.

• Games will be identified by a game ID and have attributes attendance, date, and utc\_time.

• Many games can be played at a given venue, but each game must be played at exactly one venue.

• The NFL schedule will be identified by the date and have attributes season\_year, season\_type, and week.

• Many games may be played on a given date, but each game must occur on exactly one date.

• Each game must have exactly one home team and one away team. Each team can play in many games.

• Scores in each quarter (linescores) will be identified by the game its played in, the team it corresponds to, and quarter and have attributes score.

• Games and teams can have many linescores.

• Each play will be identified by a play ID and have attributes play type, play text, quarter, seconds remaining in quarter, score value, start down, and end down.

• A game can have many plays, and each player can be associated with many plays in each game.

Additionally, a play can involve many players.

## ER Diagram and Relational Schema:

A diagram of a network

Description automatically generated

Users(uid, username, password, first\_name, last\_name, created\_on, favorite\_team\_name, favorite\_athlete\_id)

• favorite\_team\_name is a foreign key referencing Teams.team\_name

• favorite\_athlete\_id is a foreign key referencing Athletes.athlete\_id

• username is a unique key

Teams(team\_name, location, abbreviation, venue\_name, primary\_color, secondary\_color)

• venue\_name is a foreign key referencing Venues.venue\_name

Venues(venue\_name, capacity, city, state, grass, indoor)

Games(game\_id, date, attendance, home\_team\_name, away\_team\_name, venue\_name, utc\_time)

• date is a foreign key referencing Season\_dates.date

• home\_team\_name is a foreign key referencing Teams.team\_name

• away\_team\_name is a foreign key referencing Teams.team\_name

• venue\_name is a foreign key referencing Venues.venue\_name

• date, home\_team\_name, away\_team\_name is a unique key

Season\_dates(date, season\_year, season\_type, week)

Athletes(athlete\_id, first\_name, last\_name, dob, height, weight, birth\_city, birth\_state)

Positions(position\_name, abbreviation, platoon)

• abbreviation is a unique key

Rosters(team\_name, athlete\_id, position\_name, start\_date, end\_date)

• team\_name is a foreign key referencing teams.team\_name

• athlete\_id is a foreign key referencing players.player\_id

• position\_name is a foreign key referencing positions.position\_name

Linescores(team\_name, game\_id, quarter, score)

• team\_name is a foreign key referencing teams.team\_name

• game\_id is a foreign key referencing games.game\_id

Plays(play\_id, quarter, yards, score\_value, play\_type, text, seconds\_remaining, start\_down, end\_down)

Player\_Plays(play\_id, player\_id, game\_id, type)

• play\_id is a foreign key referencing plays.play\_id

• player\_id is a foreign key referencing Athletes.athlete\_id

• game\_id is a foreign key referencing Games.game\_id

## Views:

All\_final\_game\_scores(game\_id, home\_team\_name, home\_team\_score, away\_team\_name, away\_team\_score, home\_winner\_bool)

All\_third\_quarter\_scores(game\_id, home\_team\_name, home\_team\_score, away\_team\_name, away\_team\_score, away\_team\_winning\_bool)

## SQL Queries:

We implemented several interesting queries in our application. We have several queries implemented which are expected for such an app (For Ex: Finding specific games, players). We also have additional queries which return some interesting statistical information. We list the queries below:

1. Weekly Receiving stats: For a given week, list the players (with receiving stats) and the number of receiving yards.
2. Athlete Receiving stats: For a given athlete, list the number of receiving yards over all season weeks.
3. Post season game count: For each team, list the number of post season games and the type (Wildcard, Divisional, Conference, Superbowl).
4. Avg points grass indoor: How impactful (w.r.t total points) is playing on grass field versus turf and playing indoors versus playing outdoors?
5. Team rivals: Find all the games and information which involves two particular teams playing each other.
6. Top comeback wins: Which teams have the greatest number of 4th quarter comeback wins?
7. Win probability: What is the probability that a given team will win given their 3rd quarter scores?
8. First quarter greatest: Find all games where the total number of points scored in the first quarter is greater than the total number of points scored in any other quarter.
9. Game recap: Find all the information about a requested game including scores and winner.

10.) Most home wins: Which stadium(s) has the most home wins in a given season year (has the greatest home-field advantage)?

11.) Passing Statistics: Find passer statistics such as Passing yards, Pass Attempts, Touchdown Passes, pass completions and passer rating for a given season year.

12.) Stadium occupancy: How full was a stadium for a given game?

13.) Player plays: Find the plays made by a player in a particular game.

14.) Player roster: What team and position does a given athlete play for currently?

15.) Game scores: Find the teams and scores for each game in a given season year and week.

16.) Win loss: What is the number of games won and lost by a team in a given season?

17.) Popular teams: Which teams have the highest average attendance per game in a season? And what is the percentage change from the previous season’s attendance?

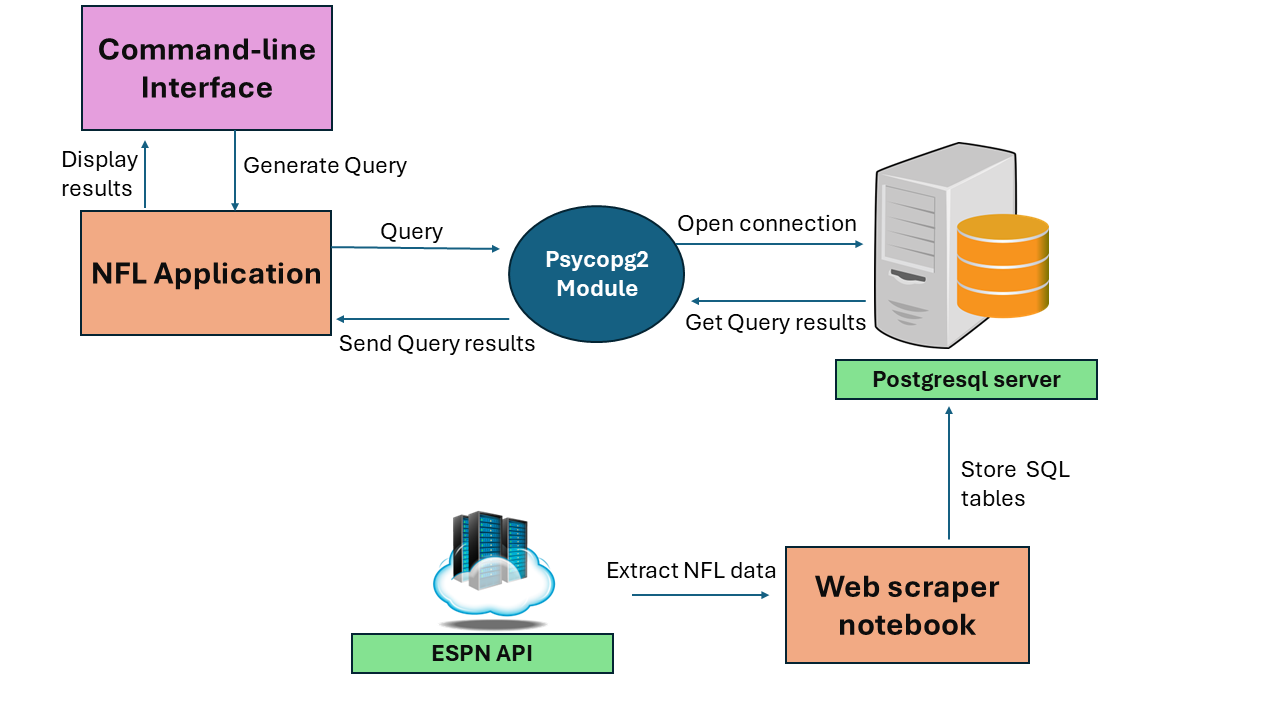
18.) Venues: What is the venue(s) along with the home team that match with the searched keywords?

19.) Users: Find the given user’s information.

# System Implementation:

Our application is implemented entirely using the Python3 programming language. For database operations we use PLpgSQL. We also implemented a web scraper which extracts and processes information from an external ESPN api to populate our database. We used a juptyer notebook for the webscraper implementation. After we have extracted the data as csv files from the ESPN api, we store them as tables in a postgresql server(postgres.cs.ksu.edu). Our application makes a connection to this database using the psycopg2 python module when it is run.

Users can connect to our NFL application, register for an account and run queries on the database and save any results in either csv or md files.



# Usage:

(Latest screenshots showcasing user interface and results needed)

The NFL Database Application is a command line program, that once launched, allows the user to issue commands to retrieve data. The commands are discussed in detail below, and each command can be controlled by the use of flags.

### Top Level Commands

There are 7 top level commands that can be executed:

1. Team
2. Athlete
3. Venue
4. Game
5. Top\_Comeback\_Wins
6. Win\_probability
7. Save
8. Login
9. Register

Each of these commands requires additional arguments and flags to retrieve specific data.

### The Team Command

The Team command is used to retrieve information related to a team. The team command can be executed without any arguments or flags to retrieve information on all 32 teams in the NFL. Optionally, users can specify a team name to retrieve information for that team only.

#### Required Arguments

None

#### Flags and Optional Arguments

#####-y or --year <year> This is used when retrieving the records for teams in a given season. When only this flag is used, the records of all teams during that season will be returned.

#####-t or --team <team\_name> This is used in conjunction with the -y or --year flag. When provided, only the record for the provided team name in the given season is returned.

#### Usage

> Team Returns all teams in the NFL  
> Team Chiefs Returns information on the Chiefs  
> Team -y 2020 Returns the records for all teams in the NFL during the 2020 season  
> Team -y 2020 -t Chiefs Returns the record for the Chiefs in the 2020 season

### The Athlete Command

The Athlete command is used to retrieve information for a given athlete. This command requires at least one argument, either the first or last name of the athlete.

#### Required Arguments

Either the first or last name of the athlete must directly follow the Athlete command. The default strategy is to search by first name.

#### Flags and Optional Arguments

#####-l or --last Search by last name instead of first

#### Usage

> Athlete Patrick Returns all athletes with the first name of Patrick > Athlete Mahomes -l Returns all athletes with the last name of Mahomes

### The Venue Command

The Venue command is used to retrieve information related to venues (or stadiums). This can be executed with no arguments, in which case it returns all venues used by the NFL. Optionally, the user can specify the venue name (or a substring of the name).

#### Required Arguments

None

#### Flags and Optional Arguments

-y or --year <year> If a year is specified, the venues with the greatest number of home wins, their teams, and the number of wins in that season will be returned.

#### Usage

> Venue Returns all venues  
> Venue GEHA Returns all venues with 'GEHA' in the name  
> Venue Field Returns all venues with 'Field' in the name  
> Venue -y 2023 Returns venues with the greatest number of home wins in 2023

### The Game Command

The Game command is one of the more flexible and powerful commands in the program. It is used to retrieve all information related to a given game or games. A large number of flags and arguments can be given to the Game command to control it's execution.

#### Required Arguments and Flags

At the base level, at least one of the following flags and arguments must be provided. However, these may or may not be required if additional flags are used.

* -y <year> Returns information for all games in the given season
* -g <game\_id> Returns information related to the game with the specified ID

#### Flags and Optional Arguments

The following flags and their arguments can be provided.

##### -s or --score

This specified that you would like to retrieve only the scores for the games. This flag requires no additional arguments, but requires the user to specify what year and week of the season to search for scores in. This is done through the -y <year> and -w <week> flags.

##### -w or --week <week>

Specify which week of the season to search in.

##### -p or --plays

Tell the program that you wish to find plays in a given game. This flag accepts no arguments, but requires the use of the -g --game <game\_id> and -a or --athlete <athlete\_id> flag. When used, the program will return all plays made by the given athlete in that game.

##### -a or --athlete <athlete\_id>

Specify which athlete to find plays for. This flag accepts an athlete ID following the flag.

##### -t or --team <team\_name>

Specify the name of the team to search for. This flag is used when retrieving the information related to games between two teams. If this flag is used, the opposing team must be specified using the -op or --opponent <op\_team\_name>.

##### -pf or --percent\_filled

Specify that you would like to know how full the stadium was for a given game. This flag takes no arguments, but requires you to specify which game to compute the statistic for through the -g or --game <game\_id> flag.

##### -S or --statistics

Specify that you would like to retrieve the leaders for passing, rushing, and receiving yards in the given game. This flag takes no arguments, but requires you to specify which game to compute the statistic for through the -g or --game <game\_id> flag.

#### Usage

The general use pattern for each of the features is listed below.

> Game -y 2020 Return information for all games in the 2020 season  
> Game -g 401437927 Return information for the game with the id of '401437927'  
> Game -s -y 2020 -w 10 Return the teams and their scores for all games in week 10 of the 2020 season  
> Game -p -a 3139477 -g 401547235 Return all plays made by athlete with ID of '3139477' in the game with an ID of '401547235'  
> Game -g 401547235 -pf Return the percent fill of the stadium for the game with ID of '401547235'  
> Game -t Chiefs -op Raiders Return all games played by the Chiefs and Raiders in the database  
> Game -t Chiefs -op Raiders -y 2020 Return all games played by the Chiefs and Raiders in the 2020 season  
> Game -t Chiefs -op Chiefs Return all games played by the Chiefs against any opponent in the database  
> Game -t Chiefs -op Chiefs -y 2020 Return all games played by the Chiefs against any opponent in the 2020 season  
> Game -g 401547235 -S Return the leaders for passing, receiving, and rushing yards in the game with an ID of 401547235

### The Save Command

In some cases, a user may wish to save the results returned by the program. This can be accomplished with the Save command. The Save command requires a filetype to be specified, and optionally a filename. The results of the most recently executed query will be saved in the user's Downloads folder.

#### Required Arguments

The filetype must immediately follow the Save command. Supported filetypes are markdown (specified as md) or comma separated value (specified as csv). The program will write the file in the appropriate format.

#### Flags and Optional Arguments

#####-o or --output <filename> The name of the file, without the extension. If this is not provided, the default filename is NFL\_last\_data.

#### Usage

> Save md Save the results of the last executed query as a markdown file  
> Save csv Save the results of the last executed query as a CSV file  
> Save md -o my\_data Save the results of the last executed query as a markdown file with the filename of 'my\_data'

### The Login Command

The login command is used to authenticate and gain access to the program.

#### Required Arguments and Flags

You must specify the username and password to use. This is done as follows.

-u or --username <username> Provide the username  
-p or --password <password> Provide the password

#### Usage

> NFLapp Login -u <username> -p <password>

### The Register Command

The Register command is used to register an account. The system will prompt you to enter various details.

#### Required Arguments

None

#### Usage

> NFLapp Register  
Prompts to enter information will follow and you will need to provide a username, password, and name.

# Evaluation:

Performance:

Easiness:

No, we did not use indexes in our application.

# Other features:

1. PuTTY/Terminal color interface: Our application supports colored font display for terminal outputs. We use the python module ‘rich’ for this implementation. This is why certain queries are designed to also return jersey colors of teams so that we can display team-based results in a color code matching their actual jersey colors.

We opted for this approach as this would be easier than implementing a web interface but at the same time being more interactive than the usual terminal interface.

1. Logistic Regression: For certain queries such as finding the winning probability of teams, we also train a simple logistic regression model based on the collected data. We used ‘sklearn’ module for this implementation. We have implemented the example query: Given the scores for the third quarter in a game, what is the probability that a given team would win?

During a live game, spectators are frequently curious about the chances of a team winning the game given the current situation. Also, live betting sites recalculate odds as the game progresses. Calculating live win-loss probability is a very common feature in sports which motivated us to include this functionality.

# Summary:

We implemented a user-friendly application that retrieves interesting statistics from NFL games. These statistics are of interest to various groups of people using sports apps, betting sites, fantasy sports etc. Our design is mostly back-end focused and deals with querying a database to generate statistics for some commonly expected information on games, although we do offer a basic interactive command line user interface.

# Experiences:

We learnt a lot from working on this project. We encountered many unexpected situations when dealing with real data obtained from the web.

1. For starters, we had to do additional processing such as adding and deleting exceptional entries to ensure the integrity of our data.
2. We were also surprised when we discovered that certain expected functional dependencies did not hold due to some technicalities. For example, there were few players who had contracts with two different teams on the same day. This occurred because their old contract with one team ended on a particular day and their new contract started on the next day but was stored in the database using the previous day due to time zone differences.
3. The data for older years such as 2013 was stored slightly differently. For example, the older data used ‘pass’ as a single play type while the new data used ‘pass attempt’, ‘touchdown passes’, ‘pass interceptions’.
4. The project enabled meeting and working with other people. Designing and discussing various ideas was an enjoyable experience.

# Possible future work:

1. Currently, the data stored in our database is static data over the past decade of NFL games. It would be interesting to have an application that regularly updates the database by automatically scraping the latest NFL data. However, there are challenges to face as the new data stored in ESPN website may not follow the same method and structure of storage(ESPN Data is stored as json files in big data platforms such as NoSQL databases).
2. It would also be interesting to provide an interface to users with higher levels of access(‘admins’) to add more queries or to modify existing queries to the application.
3. Naturally, an interactive web interface is the next step and possibly deploying the application as an app on Android or iPhone platforms.

# Teamwork experience:

We did not rigorously partition work and distributed amongst ourselves. We had regular meetings where we discussed things to accomplish for the next meeting or next assignment (Project proposal/SQL Queries).

Everyone equally contributed to design and discussion of various stages of implementation (Web scraping, Raw data processing, Database design, SQL queries, python implementation, user interface). However, Chuck Zumbaugh was the significant contributor of the codebase of our application.