# Project -1- Report and Developer Notes

## I. Getting Started

### I.1 Running the Virtual Environment:

1. Type 'pip install virtualenv' to install virtualenvironment
2. Navigate to the directory in which you want to place your environment
3. DO NOT PUSH the venv into the github. Put it in a .gitignore.
4. To create a venv, type 'python -m venv name_of_venv'
5. to activate the venv, type 'name_of_venv\Scripts\activate.bat' for windows, and 'source name_of_venv/bin/activate' for MacOS
6. Once the venv is activated, you should notice that the terminal should have a ("name_of_venv") at the front of the terminal line.
7. While inside the venv, type 'python install -r requirements.txt'

### I.2 Running the Web App

1. Make sure the virtual environment is running
2. Navigate into the folder containing the project (in this case, 'sports_gui')
3. Type into the command line 'python manage.py runserver'
4. If the compilation is successful, follow this link: http://127.0.0.1:8000

## II. How to Use the Web App

### II.1 Home Page:

The first page you should land on is the homepage. There should be a navbar at the top with links to pages such as "Leagues, Date, Queries" etc.

All fields are mandatory unless stated otherwise.

Navigate to whichever tab you would like to get started.

### II.2 League Page

This page creates new leagues. When creating an entirely new league, you must have created teams first before inserting them into the league, otherwise, the league cannot be created.

Scroll down on the page to insert teams. Here, you'll have to input fields such as name, field, state, etc. to create the team.

If there are teams that you know you want to insert into a league, go ahead and create the league. Each league will have a commissioner, and you will have to put the name of the commissioner and their SSN into the system.

Notice that creating a league will initialize the first season. You have the option to either automatically insert games, manually insert them, or to not insert any games at all for that season. You also get to create the rules for the season, meaning how many points a win is, a loss is, and how many points a draw is. Note that win > draw > loss, otherwise the input will not be accepted.

Manually creating the teams will mean that you will have to insert the date and the field that each and every possible combination of game is played at.

### II.3 Date

Here, you can change the current date. Notice that this will affect how you put in game results, as you can only enter game results for that day only. Additionally, you cannot change the date to go backwards (despite what Tame Impala sings, while it may feel like you only go backwards, you can only go forwards).

### II.4 Season

Create a Season for a League. The league in question has to exist in the database, otherwise it will return that there is no such league. Additionally, the new season cannot be in conflict with a season that the league is playing already. You must also place the number of games per day and the maximum number of games total. Finally, there is the option that you can manually create the games in the season.

### II.5 Move Teams

On this page, you can change teams from league to league. The team must exist, and it must exist in the league that you indicate. The league you wish to move this team to must also exist as well.

### II.6 Games

You can insert games into a season. If at the beginning of the league creation you chose not to insert any games, here's your chance to do so! Just like on all the other pages, you can insert the games into whichever league you desire. You must also know when the start and end dates of the season you want to insert games into. Just like in both the season and league menus where you have to insert games, you once again have the option to insert games into the season automatically or manually.

### II.7 Queries

You have the option to submit several queries to the database. You can find information about:

1. Leagues
   For leagues, you can find general information about the leagues, or you can find the champion of the league
2. Teams
   You can also find general team info and also the records for that team
3. Games
   You can find games played between two teams, and it will return which league these games were played in, and the game, the game date, and the score
4. Seasons
   Find information about a season of a league. You must know the exact start and end date about the season, otherwise it won't find it
5. Ratings
   Here, you can find the ratings for a league. Again, you must know the exact start and end date of the season.

## III. Developer Notes

### III.1 Backend:

All the backend functions that we used in our project can be found in the back_end\Project7330_Fct.py.

A naming convention was respected so that variables should start with a lower case, and constants should start with an upper case to avoid conflicts between different parts of the program.

In the first part of the program, the database connection/location is set by default to be on **localhost,** this setting can be modified easily by inserting the proper URL into the mongo client method.
In addition, all the collections and indexes are defined in one group before starting with the functions definition.

The functions were ordered in a way that each category is grouped together, starting from the insertion up to get/set functions and ending with the queries part.

Instead of writing simple MongoDB queries methods, we used aggregate features whenever possible to increase the speed of execution and the efficiency of the methods.

Some of the functions had added features, presented as optional calling parameters to offer more flexibility to the user to implement the same function in different scenarios.

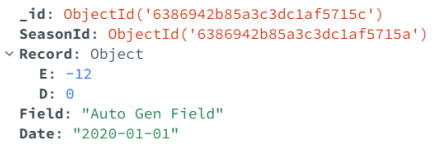
Finally, a ___res variable was added at each return situation, so that functions can return formalized strings or dictionaries to ease the process of displaying them to the user.

The naming of ___res was intentional so that if the functions files was to be implemented in a different project the variable can be found/access and modified easily.

As final data architecture we settled on 5 collections: Dates, Games, Leagues, Seasons, Teams with the following architecture:

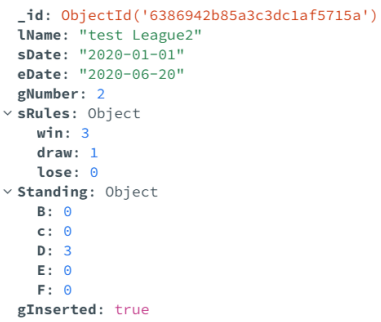
### III.1.1 Dates:


### III.1.2 Games:

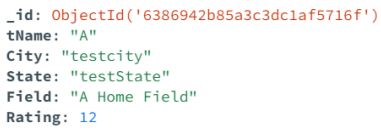



### III.1.3 Leagues:


### III.1.4 Seasons:




### III.1.5 Teams:



### III.2 Front and back-end connection

Connecting the front end with the back end involved getting the input values entered by the user on the webpage and feeding it to the function defined in the backend.

After the values have been fed to its respective functions, we would retrieve the result and display it on a newly created webpage

If the user decides to manually (dynamically) insert the date and the team field in leagues, seasons or a game, the user will be taken to another webpage where the user can dynamically insert the values to be eventuality transmitted as parameters to the function. After processing the input by the logic implemented into the backend functions, the user will be taken to another webpage where the results will be displayed.

We have used the Django template language (DTL) to embed Django logic codes into HTML. A Django template is a text document or a Python string marked-up using the Django template language and it helps us to provide a bridge between HTML and python.

We also use *Render* to combine a given template with a given context dictionary and have an HTTP response as a return with that rendered text. The render uses the argument '*request*' to generate a response.

In our code, we have used 3 variables to render an HTTP response. First is the 'request', second is the name of the html webpage where it will be rendered and third is a variable named '*params*'. 'Params' variable is introduced to pass a dictionary to the template, which can be accessed directly through html.

## III.3 Future Changes

There are several future changes that we would like to implement in. These changes mainly revolve around changing the type of inputs that are available

1. When entering game results, instead of a text field for team entry, replace it with a dropdown menu that contains all the available teams. When a team is selected, the dropdown menu options for team 2 will change based on the value for team1.
2. When moving teams from one league to another, change the text inputs to dropdown selection menus. For example, the team's dropdown menu will show the list of teams available, and the leagues entry should be changed to dropdown menus as well.

3. On the insert games page, instead of relying on the user to input the exact dates when a season starts for a league, create a dropdown menu that shows which seasons have been created for the league entered.
4. For all of the queries, except the 'League info' and 'Team info' queries, print the output in a neat format.