NBA App User Manual

Quinn Butcher and Viswa Rathnakumar

CMPSC 431W

Professor Dong

Pennsylvania State University

Table of Contents

Introduction

Welcome to the NBA app, Quinn Butcher and Viswa Rathnakumar created this app for the final project of 431W. In this project, we aim to show a basketball statistical app that has many SQL features that we have learned over the course of the semester.

Features

In this app, you will be able to:

- View players, teams, and games
- View statistical counts for each
- Search for players, teams, and games based statistical counts
- View an advanced statistical table based on teams
- Enact trades between teams
- Obtain video links for each game
- Insert players into the database
- Update players' information in the database
- Delete players in the database
- View award counts of teams

We hope you enjoy the nba-app!

Getting Started

MacOs

Backend Installation

Install PostgreSQL via Homebrew

Open your terminal and run:

```
brew install postgresql
```

After installation, start the PostgreSQL service:

```
brew services start postgresql
```

Initialize the PostgreSQL database cluster if it isn't already:

```
initdb /usr/local/var/postgres
```

Connect to PostgreSQL:

```
psql postgres
```

Set up your database and user:

```
CREATE DATABASE nba_db;
ALTER USER postgres WITH PASSWORD 'Sup3rP@ss45';
```

Exit PostgreSQL:

```
\q
```

Create a Python virtual environment:

```
python3 -m venv env_name
```

Activate the virtual environment:

```
source env_name/bin/activate
```

Install packages from requirements.txt:

```
pip install -r requirements.txt
```

In the backend/ folder, create a .env file:

```
touch backend/.env
```

Add the following lines to .env:

```
export DB_HOST=localhost
export DB_PORT=5432
export DB_USERNAME=postgres
export DB_PASSWORD=Sup3rP@ss45
export DB_NAME=nba_db
```

If you can't edit .env directly, create a .txt file first, edit it, and then rename it to .env.

Running the Backend

Run the backend server:

```
uvicorn app.main:app --reload --workers 1 --host 0.0.0.0
--port 8000
```

Installing the Frontend

Open your terminal and run the following command to install Node.js:

```
brew install node
```

After the installation is complete, verify the installation:

```
node --version
```

```
npm --version
```

After installing Node.js, you can install Yarn using Homebrew as well:

```
brew install yarn
```

Verify Yarn installation:

```
yarn --version
```

Running the Front end

Navigate to the nba-stats folder and run these commands

```
yarn build
```

```
yarn start
```

You should now see the site of the application!

Windows

Backend installation

Install Postgres from https://www.postgresql.org/download/windows/

The installer will ask for some information

We only need the PostGreSQL Server and Command Line tools installed, the rest can be unchecked

> The password should be Sup3rP@ss45
>
> The port is 5432

Press next until the installer installs the application, this may take a few minutes

Add Postgres' bin folder to your PATH variable, the directory of the bin is something like

```
C:\Program Files\PostgreSQL\17\bin
```

Where 17 is whatever version os PostgreSQL you have installed.

Use this command to test if Postgres is installed:

```
psql --version
```

Create a virtual environment in python to put all of our packages in:

```
python -m venv env_nam
```

Run the environment:

```
env_name\Scripts\activate
```

Now install the packages in the requirements.txt:

```
pip install -r requirements.txt
```

Create a file .env in backend/ and put these lines into the file

```
export DB_HOST=localhost
export DB_PORT=5432
export DB_USERNAME=postgres
export DB_PASSWORD=Sup3rP@ss45
export DB_NAME=nba_db
```

If you cannot edit a en.env file, edit a .txt file first and then change the extension to .env

Running the Backend

Run the backend app with this command:

```
uvicorn app.main:app --reload --workers 1 --host 0.0.0.0
--port 8000
```

Frontend installation

Instal node.js from the official website, and include npm in the installation settings

Verify by running these commands:

```
node –version
```

```
npm –version
```

Now we need to install yarn:

```
npm install –global yarn
```

Running the Frontend

Navigate to the nba-stats folder and run these commands

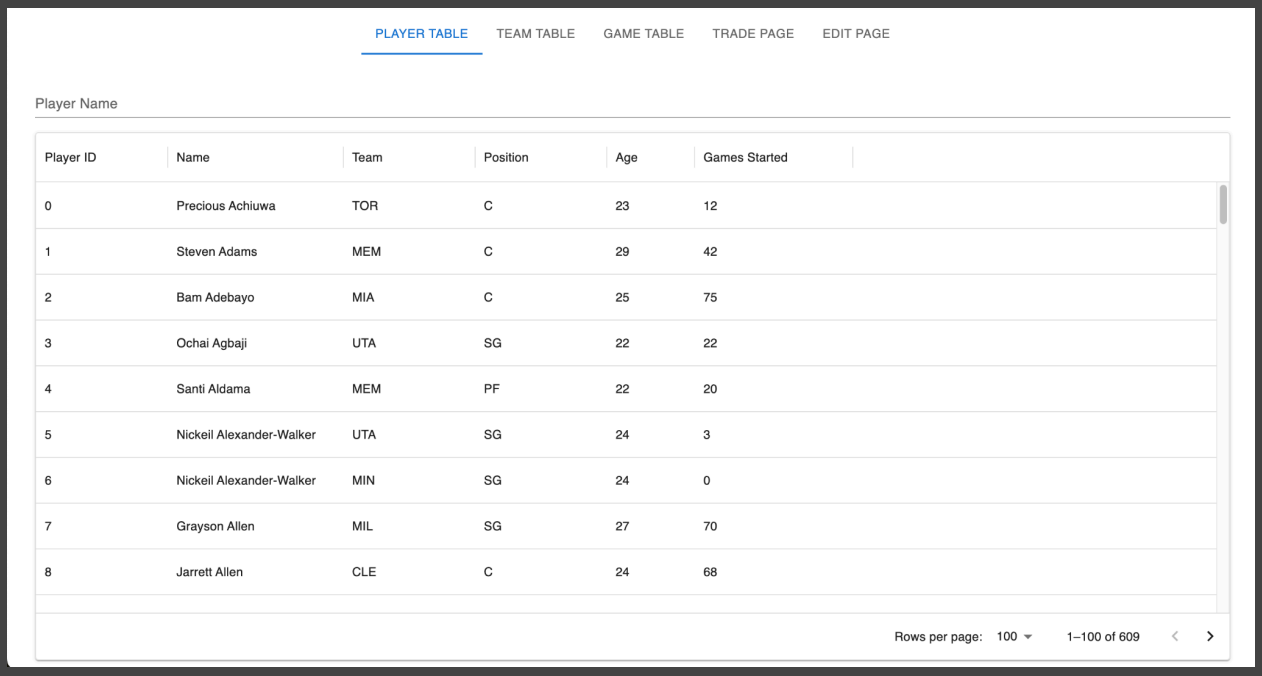
```
yarn build
```

```
yarn start
```

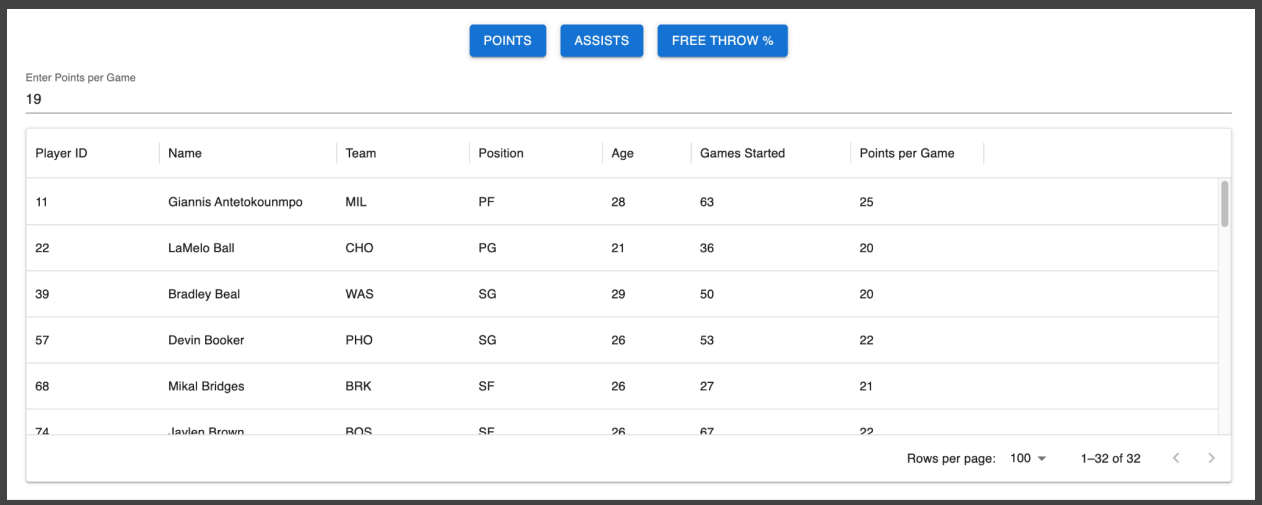
You should now see the site of the application!

Navigating the Site

Player Data Table

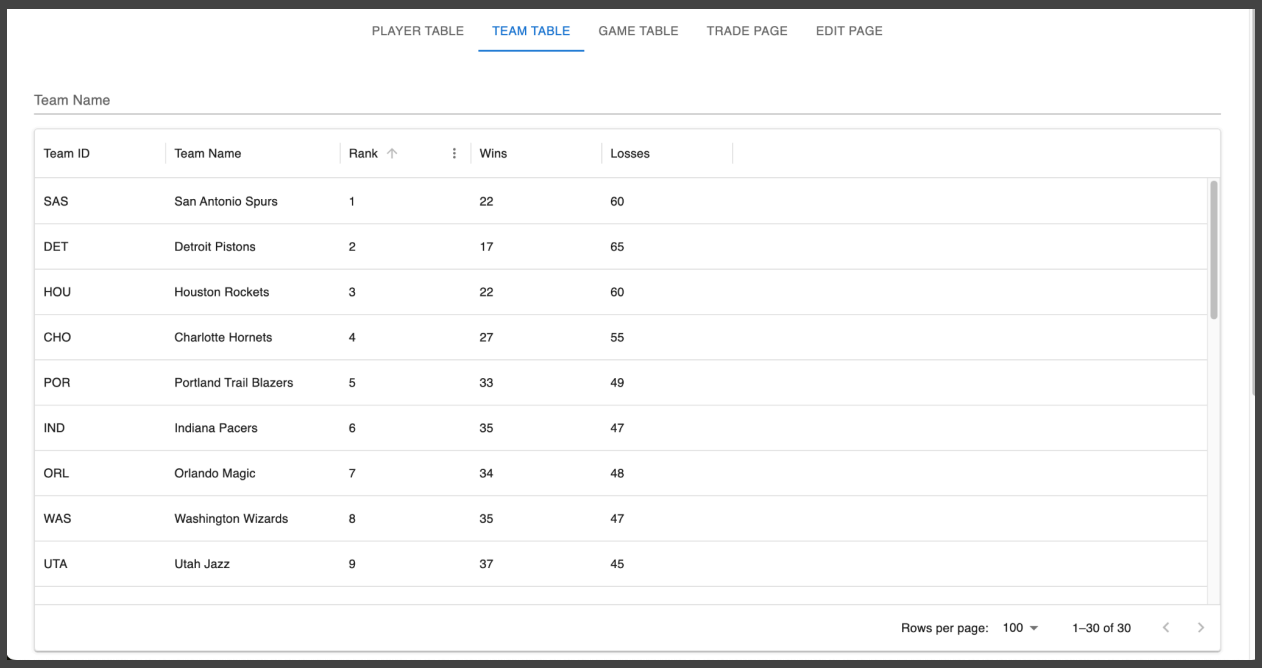


The search bar on the top allows you to search the player by name, and results will show up in the dropdown as you type.

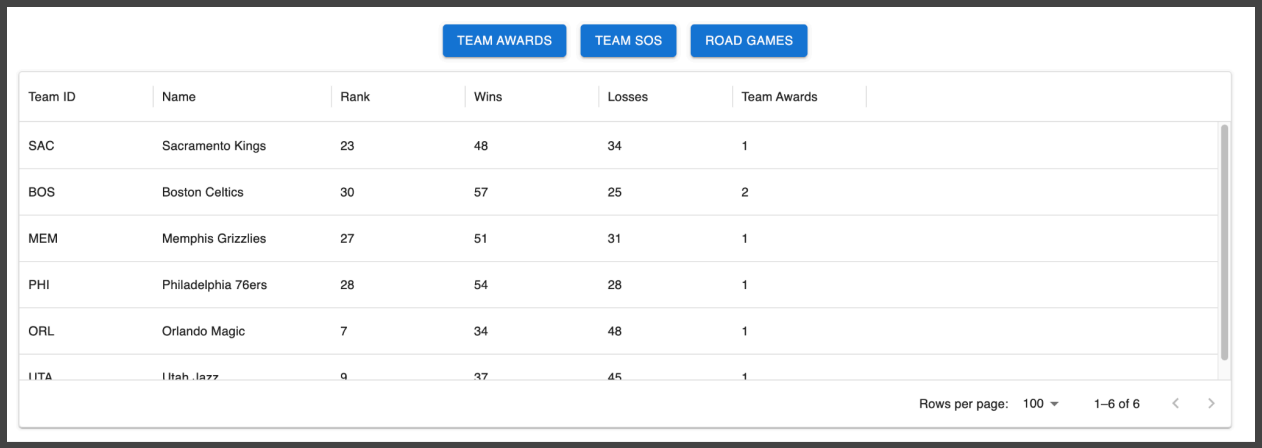


Scrolling below the initial table will show the statistical selection table. Here you can click on any one of the points, assists or free throw % button to show a table based on those stats. In the search bar, enter the value of the statistic you wish to see players of. For example, typing in 19 points after clicking the points button yields this list of players that have above 19 points.

Team Data Table

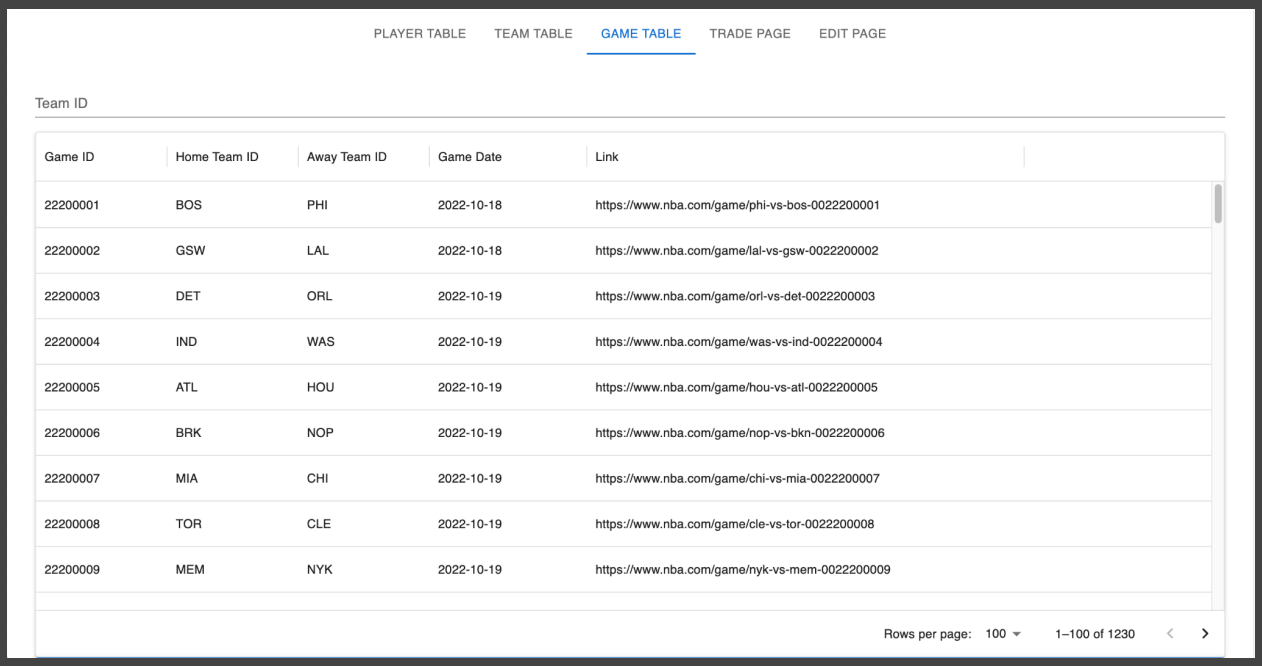


Similar to the player table, this table yields simply results of teams. You can search through by team names and teams will appear in the drop down.

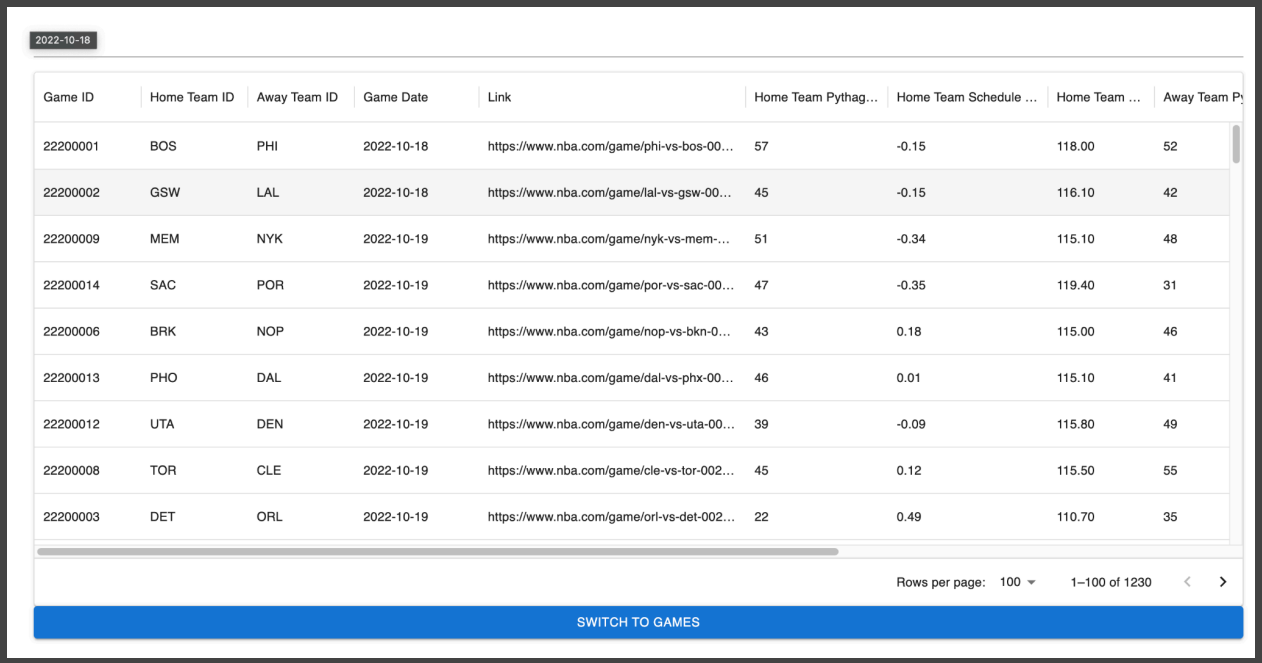


Scrolling down yields the team statistical selection. Selecting the Team SOS button will show a input text field where you can input a percentage of SOS to find teams above that SOS. Clicking either Team Awards or TEAM SOS will show the column with that statistic for all teams.

Game Data Table

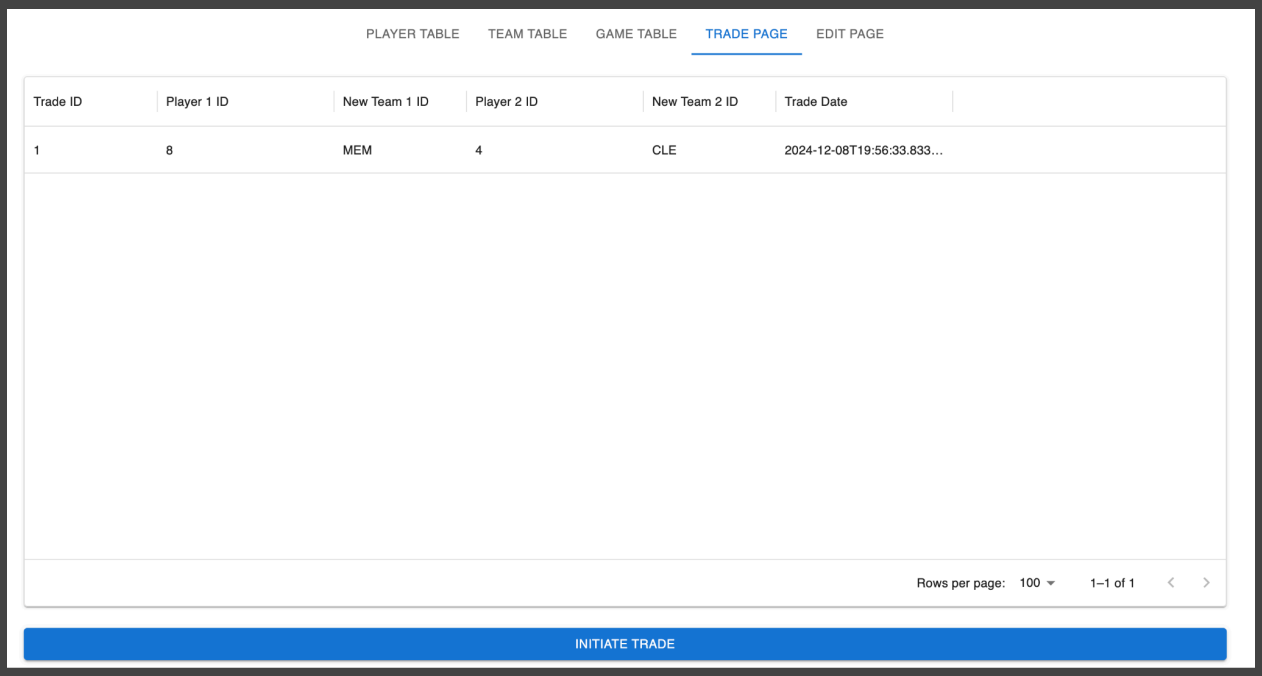


The game table shows all games between teams and their video links for the season. The table is searchable by team_id, so searching BOS will yield all the home and away gaems for the Boston Celtics for the season.

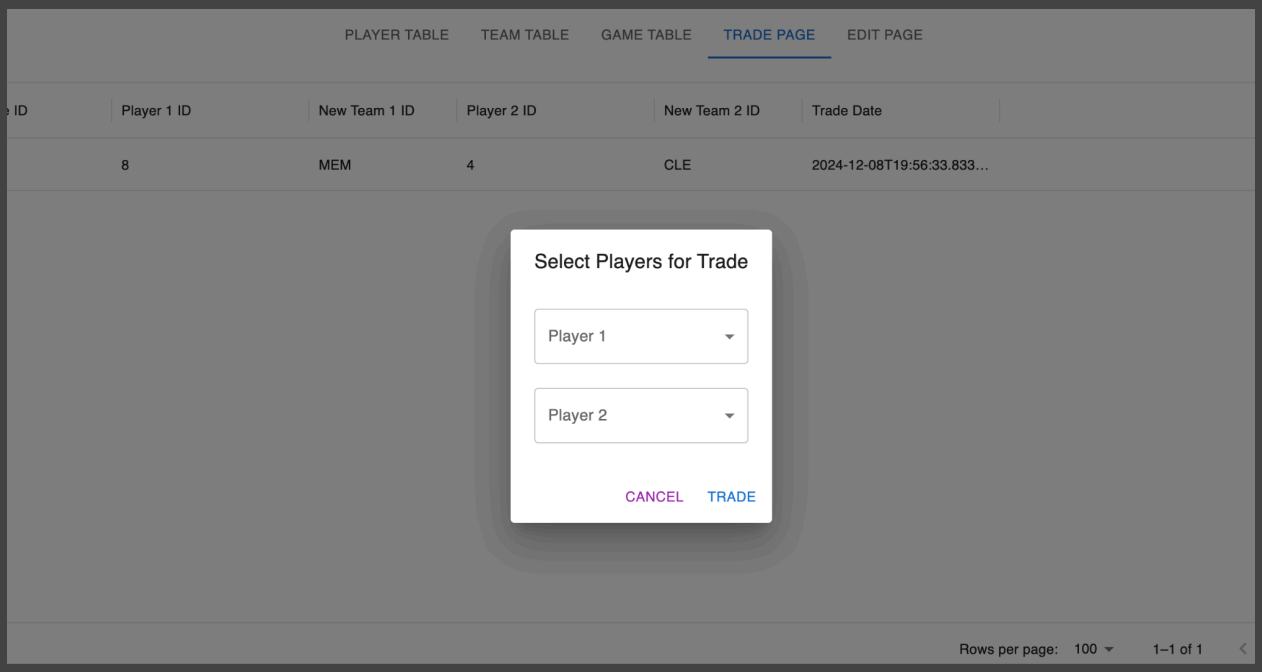


The button below the games table can be pressed to yield the advanced stats for the teams and their games. The advanced stats show Pythagorean wins, Schedule strength, ORTG, as well as the arena and the attendance of the arena. This feature requires more than 5 tables joined as each of these vital game statistics come from several different tables.

Trade Table

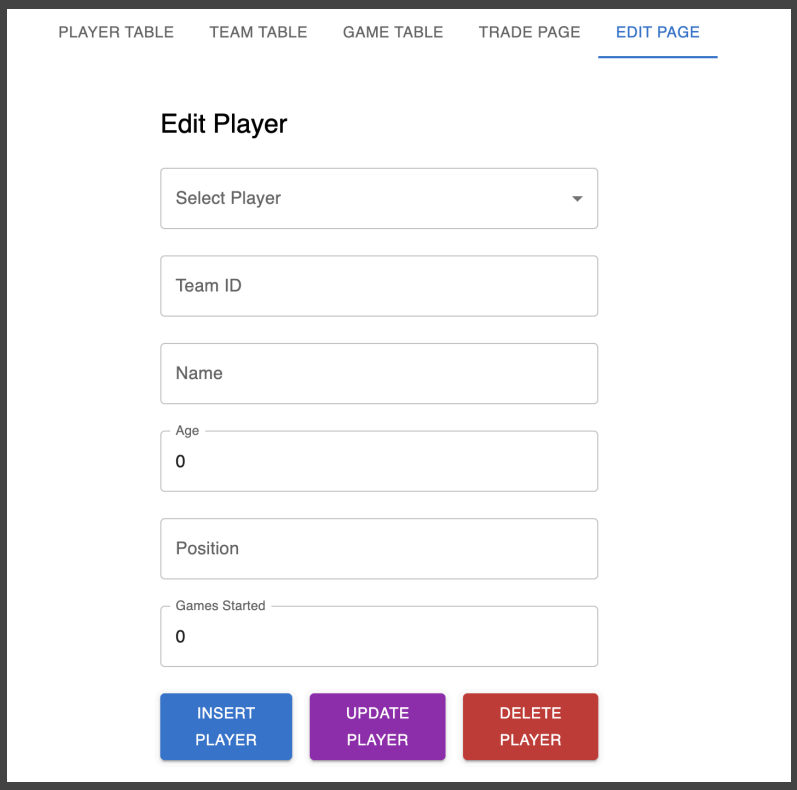


The Trade table is where you can input trades that switch the teams of two selected players.



After clicking the Initate Trade button, a view pops up. Here you can select two players to be swapped. Once you hit the trade button, the database will swap the players on the teams, which will be shown in the trade table view.

Edit Table



This is the edit table page, where you can insert, update, and delete a player in the database.

Inserting Records

To insert a player record into the database, simply fill out the required fields and press the insert player button.

Updating Records

To update a player's records, select a player through the dropdown, modify the desired fields, and select the update player button

Deleting Records

To delete a player’s records, select a player through the dropdown, and select teh delete player button

Backend Routes

A Quick Note on Return Types

All routes have the same return type of List[dict]. For example, a query returning players with matching initial characters to “VIT” would yield:


```
[
    {
        "player_id": 314,
        "team_id": "ATL",
        "name": "Vit Krejci",
        "age": 22,
        "position": "PG",
        "games_started": 0
    },
    {
        "player_id": 425,
        "team_id": "MIA",
        "name": "Victor Oladipo",
        "age": 30,
        "position": "SG",
        "games_started": 2
    },
    {
        "player_id": 593,
        "team_id": "MEM",
        "name": "Vince Williams Jr.",
        "age": 22,
        "position": "SG",
        "games_started": 1
    }
]
```

For a team:


```
[
    {
        "team_id": "PHI",
        "name": "Philadelphia 76ers",
        "rank": 28,
        "wins": 54,
        "losses": 28
    }
]
```


And for a game:


```
[
    {
        "home_team_id": "DET",
        "away_team_id": "ORL",
        "game_id": 22200003,
        "game_date": "2022-10-19",
        "link": "https://www.nba.com/game/orl-vs-det-0022200003"
    }
]
```


Essentially, these routes all return a List[dict], so if you operate with the routes, keep in mind how the data is stored.

Server setup routes

POST: http://0.0.0.0:8000/initdb

This route drops and sets up the tables again. Any trades or modifications made to the player table will be reset.

Player routes

GET: http://0.0.0.0:8000/players?name=pre

Players with matching characters to “pre” will be returned. In this case, one player, Precious Achiuwa.

GET: http://0.0.0.0:8000/playerPoints?points=10

Players with total points above 10 will be returned. The query parameter points must be an integer greater than or equal to 0.

GET: http://0.0.0.0:8000/playerAssists?assists=1

Players with total assists above 1 will be returned. The query parameter assists must be an integer greater than or equal to 0

GET: http://0.0.0.0:8000/playerFTPerc?FTPerc=.5

Players with a free throw percentage above 50% will be returned. The query parameter FTPerc must be a float between 0 and 1.

Team routes

GET: http://0.0.0.0:8000/teams

All teams will be returned

GET: http://0.0.0.0:8000/teams?name=Phil

All teams with matching initial characters to ‘Phil’ will be returned, in this case, the returned team is only the Philadelphia 76ers. The query parameter name must be a string.

GET: http://0.0.0.0:8000/teamAwards

All teams and their award counts will be returned.

GET: http://0.0.0.0:8000/roadgames

All teams and their number of road games will be returned.

GET: http://0.0.0.0:8000/teamSOS?SOS=.3

Teams above the Schedule Strength of .3 will be returned. The query parameter SOS must be a float between 0 and 1.

GET: http://0.0.0.0:8000/advancedGameStat?team_id=SAS

Teams with matching initial ids to ‘SAS’ will be returned. The query parameter team_id must be a string.

Game routes

GET: http://0.0.0.0:8000/games

All games are returned

GET: http://0.0.0.0:8000/games?team_id=O

All games with a matching team_id to ‘O’ will be returned. The query parameter team_id must be a string.

Trading routes

GET: http://0.0.0.0:8000/trades

All trades will be returned.

POST: http://0.0.0.0:8000/tradePlayers

Body:

```
{
    “player1_id”: 101,
    “player2_id”: 102
}
```

This route will initiate a trade transaction between players with ids 101 and 102. Player_ids must be integers.

Insert, Update, Delete Player routes

POST: http://0.0.0.0:8000/insertPlayer

Body:

```
{
    "team_id": "SAC",
    "name": "Viswa Rath",
    "age": 21,
    "position": "F",
    "games_started": 60
}
```

This route will insert a player with the information in the body of the request. All fields must be initialized. The team_id and postion must match with an existing team and position

Here are the types of the body:

team_id: string, name: string, age: int, position: string, games_started: int.

PUT: http://0.0.0.0:8000/updatePlayer

Body:

```
{
    "player_id": 0,
    "team_id": "TOR",
    "name": "Precious Achiuwa",
    "age": 23,
    "position": "C",
```

"games_started": 45

}

This route will update the player with the matching player_id to the information in the body. All fields including player_id must be initialized. Player_id must match an existing player_id

Player_id: int, team_id: string, name: string, age: int, position: string, games_started: int.

DELETE: http://0.0.0.0:8000/deletePlayer

Body:

{

"player_id": 0

}

This route will delete the player with the given player_id. The player_id must be an integer of a player that already exists.