Dribble Dynamics: NBA Stat Visualization Portal

Jay Parmar
Rutgers University
Piscataway, NJ, USA
jay.parmar@rutgers.edu

Jonathan Wang
Rutgers University
Piscataway, NJ, USA
jw1303@scarletmail.rutgers.edu

***Abstract*—** We aim to build a multi-featured interactive portal that enables statistical insights of The NBA players using a culmination of various NBA related datasets totalling around 14+ million records provided by NBA API datasource [1]. On our portal, users will be able to select players in the portal and view data visualizations like Player Career Progression, Play by Play performance, Shot Heatmaps, similar player clustering etc. We aim to process around atleast 2-3 GB of data on local development machine/ distributed system, provide an extremely responsive interface and a fluid experience to the users gathering useful information.

***Keywords*—** The NBA, Basketball, Classification, Game analysts, Research, sport statistics, Full Stack Web Application, Data Processing, heat maps, MVT Architecture, MySQL, Interactive applications, SQlite, Python3, nba-stats-api, nba-api

I. Project Description

The NBA is regarded as one of, if not the, top premier professional basketball league in the world. Since it's inception in 1946, the NBA has provided countless hours of entertainment to fans, endless commercial value to companies and teams, and consistent improvements towards how the game of basketball is played. At the highest level of individual skill, even the smallest statistical improvements and advantages can make a difference. The motivation behind embarking on this NBA data visualization project stems from a deep appreciation for the dynamic and rich world of basketball statistics and a desire to provide fans, analysts, and enthusiasts with a visually immersive experience. In the past, the NBA has generated extensive dataset comprising player performances, team dynamics, and game statistics. By harnessing this wealth of information, this project aims to unlock valuable insights and trends, enabling users to delve into the intricacies of the sport. Whether exploring the career trajectories of star players, analyzing team strategies, or tracking the pulse of a game through interactive visualizations, this project seeks to transform raw data into a compelling narrative. Ultimately, the goal is to enhance the understanding and enjoyment of the NBA by presenting complex statistical information in an accessible and engaging manner, catering to both casual fans and avid analysts alike.

To achieve this we aim to establish an interactive data visualization portal for NBA player related statistical insights. As of the time of writing, the primary focus for this project is on player related statistical data. In the future, we can also expand this towards entire teams as a whole.

A. Stage1 - Brainstorm and requirements gathering Stage.

After going through some rigorous brainstorming process; combining ideas and considering interest of all team members, we narrowed down to some selective following datasets depending on the usecase we want to derive for our project from the Kaggle source. Note this kaggle sources as listed below were only for inspiration purposes, looking at these dataset gave us an idea of what kind and amount of data we need to fetch from the official NBA API which acts as our only data source [1]. This is an widely used API client package to access the APIs of the NBA. Infact most of these kaggle sources have been built using this package only :

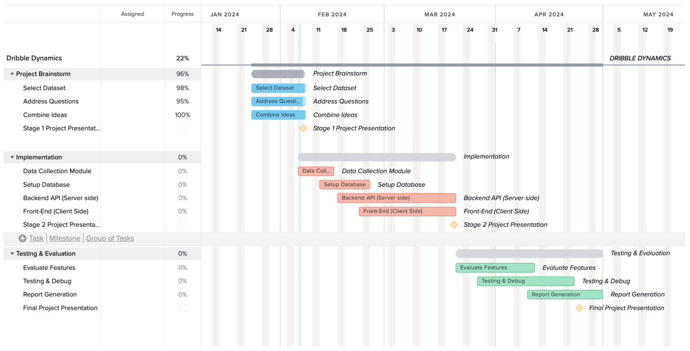
- NBA Database [2]
- NBA Player stats since 1950 [3]
- NBA shot logs [4]
- NBA Player Shot Dataset (2023) [5]

The total size of our data pool is roughly around 2.5 GB with 14M+ records across 25 CSV Tables subject to fluctuations in the later part of implementation stage after pre-processing data and loading it into our database. We aim to log the historical data of the NBA since it's inception in 1946-1947 season. Current version of our application only supports data from 1984 season till now. If we want, some of the datasets on the pool can be provided with the ability to update from the official NBA-API on a daily or monthly basis (Related updating scripts provided in datasets' references). This makes our data extremely time variant and time dependent which ultimately also allows our users to work with most recent data. The area of focus is currently on player, team and shot related data and hence corresponding entities are being selected from all above datasets and put together in a MySQL database through our Data ETL Module as mentioned in out following Mode of processing flowchart. (Fig. 2)

This application is designed for the following target users:

- **Players:** Track their progress, make play-by-play comparisons, and identify similar players to learn from
- **Coaches/Analysts:** Analyze player performances, identify statistical weak points, and strategize gameplay
- **Fans:** Follow and learn from their favorite players stats/playstyles

Below is our Project Timeline and Division of Labor:



	Assigned	Progress
Dribble Dynamics		22%
▾ Project Brainstorm		96%
Select Dataset		98%
Address Questions		95%
Combine Ideas		100%
Stage 1 Project Presentat...		☐
▾ Implementation		0%
Data Collection Module		0%
Setup Database		0%
Backend API (Server side)		0%
Front-End (Client Side)		0%
Stage 2 Project Presenta...		☐
▾ Testing & Evaluation		0%
Evaluate Features		0%
Testing & Debug		0%
Report Generation		0%
Final Project Presentation		☐

Fig. 1. Gantt chart for the project

The division of labor is best estimate and subject to minor changes as we progressed through the course:

All members: Brainstorm and Questions addressing, Data ETL module, Database Setup
Jonathan: UI, Front-End client side, Application Styling
Jay and Jonanthan: UI, Backend, background data processing, Features Oversight
Jay: Testing, evaluation, Documentation

B. Stage2 - The Design Stage.

As the first step towards the design of our project, we first solidify what important questions are we going to address and build the application around this backbone. Following are some fundamental questions we want to answer about the data we have:

1) **Question 1** : What player based statistics can we visualize to understand their trends, trajectories, and performance? And how?
2) **Question 2** : What key takeaways can our users make from these visualizations?
3) **Question 3** : Can we potentially extend analytics to team based profiles with synergistic stats?

Some of the **main features** the users would be able to access from our portal in terms of visualizing the data are:

- Player Career Progression: Ability to see a summary stats of a particular player through out their career
- Play-By-Play Performance: A highly interactive visually rich court map depicting play-by-play of a particular game between two teams
- Similarity between Players: Side by side comparison between players and specific statistics

Following figures showcase our processing flow chart employing a standard Client-Server Architecture (see Fig.2) and Entity relationship Diagram (ER) (See Fig.3) between all the entities we consider important from our data pool to achieve desired result.

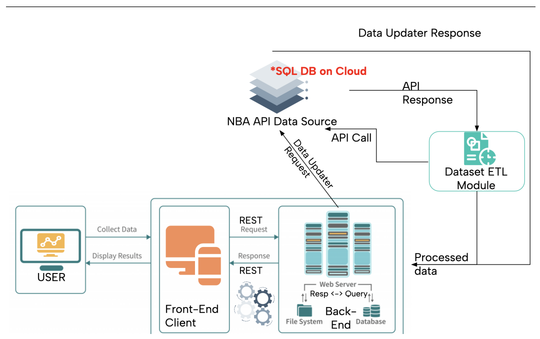


Fig. 2. Data flow chart/ Mode of Processing

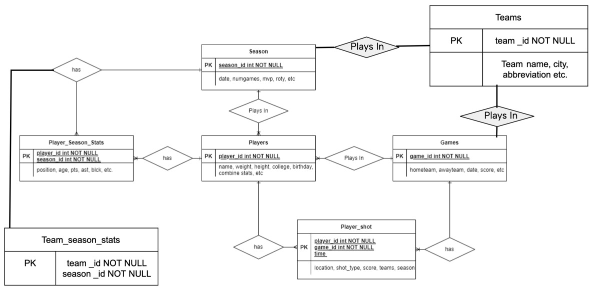


Fig. 3. The ER Diagram

- Our users will predominantly interface with our application through their own web browser. Within the web app, they can search their favorite players and visualize information about them. When they perform an action, their client will communicate to our backend servers via RESTful API. In our backend, we will be running a Flask server and utilizing Python's rich data science libraries for processing. The actual data itself will be hosted on a MySQL server and our Flask server will manipulate queried data into the proper format for our front end to receive and visualize.
- The design of our MySQL server is still in progress, but for now we have our entities as Seasons, Player stats per Season, Players, Games, and Player shots, Team, Team Stats per Season. These are the minimum required entities we have in mind for now, but they may be expanded as we continue adding features. Likewise, the specific fields within these entities is also subject to change to keep in correspondence with features.

C. Stage3 - The Implementation stage.

This web application primarily uses client-server architecture with MVC (Model, View, Controller) design pattern. The raw datasets had multiple csv files with over three million records. After preprocessing and filtering, all the data will be stored in a MySQL database (Model). MySQL workbench has been used for importing csv data files into the respective tables. The controller for our architecture will be a Python Flask server running our backend data processing algorithms and handling queries on our SQL database. When users perform data visualization requests, the server will handle all the data manipulation from filtering to processing to formatting and send only the required data to the individual user views. These views will be the web application that our users will interface with. This will be a React.js web application running client side in our users' web browser. It will communicate with our controller via RESTful API calls and receive data in the form of processed JSONs which can then be visualized.

The steps methodology for such project implementation are as follows:

- Data Sources: NBA API or Other Datasets: Fetch NBA data from sources like the NBA API, Basketball-Reference, Kaggle, or ESPN API.
- Backend (Server-side): Server Application: Develop a backend server application (Python with Flask/Django, Pandas, etc.) to handle data retrieval, processing, and API requests. Database: Store data in a database (MySQL) for efficient retrieval. API Endpoints: Create RESTful API endpoints for different data types (players, teams, games) to serve data to the frontend.
- Data Processing: Data Cleaning and Transformation: Process raw data, clean it, and transform it into a format suitable for storage in our database and for visualization.
- Frontend (Client-side): Web Application: Develop a frontend web application (React.js, HTML/CSS/JS) for user interaction. Charts and Visualization Libraries: Utilize charting libraries (e.g., D3.js, Chart.js) for creating dynamic and interactive visualizations. API Calls: Use RESTful API to communicate with the backend, fetching and displaying data in the UI.
- User Interaction: User Interface (UI): Design an intuitive and user-friendly interface for users to interact with the visualizations. Filtering and Sorting: Implement features for users to filter and sort data based on different criteria.
- Client-Server Interaction: API Requests: Frontend makes requests to the backend API endpoints to fetch required data. Asynchronous Communication: Implement asynchronous communication to avoid blocking the UI during data fetching.
- Deployment (if necessary): Hosting: Deploy both the frontend and backend on hosting platforms (e.g., AWS).
- Testing and Debugging: Unit and Integration Testing: Test the backend and frontend components separately for functionality

and integration. Debugging Tools: Use debugging tools for identifying and fixing issues during development.

A list of softwares/ libraries we intend to use in this project:

- Front-End (Client Side): ReactJS [6], D3.js [7], etc. for dynamic and interactive visualizations
- Back-End (Server Side): Python with flask for data processing, retrieval and API request [8]
- Datastore: MySQL Database Server Side [9]

Front-End Summary: Our front end web application will be running on React.js with D3.js as the visualization backbone. Interactions with the backend will be executed through REST API requests.

Backend-End Summary: Our back end will be running as a Python Flask server to receive and handle REST requests and perform the necessary real time processing for serving visualization data. It will handle querying our database for requested data, parsing said data, and formatting it to serve back to our user clients.

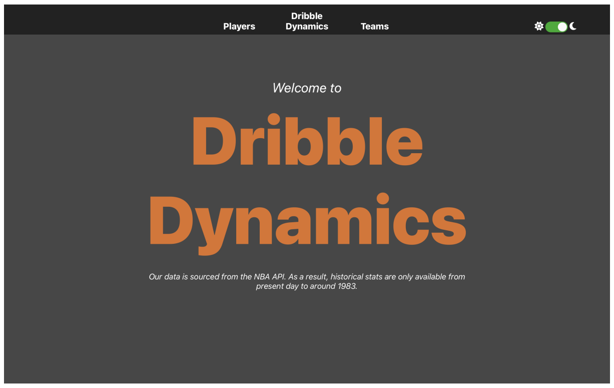
Database Summary: The database hosting our NBA data is built with MySQL according to the afforementioned ER diagram.

D. Stage4 - Visual Representation / User interface/ Interactivity.

In our intermediate project report we showcased mock design interfaces made in Figma. Our final application version consist of 4 screens, a landing home page, a player search page, and individual player visualization pages with various charts and graphs, a team search page along with team statistics. To make these, several essential libraries will be required:

- numpy, pandas, scipy for data processing
- d3.js for drawing graphs and plots
- React.js for UI/UX

Our home page (see Fig. 4) primarily consist of a description of the project as well as additional notes that we want our users to be aware of. On the top for all our pages will be a navigation bar containing links to navigate between views of Players segment to the left or Team segment to the right. For user preference, there is also a light/dark mode toggle switch on the far right.



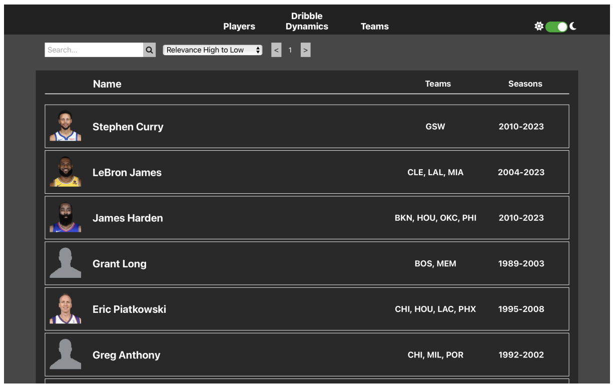
Welcome to

Dribble Dynamics

Our data is sourced from the NBA API. As a result, historical stats are only available from present day to around 1983.

Fig. 4. Home page

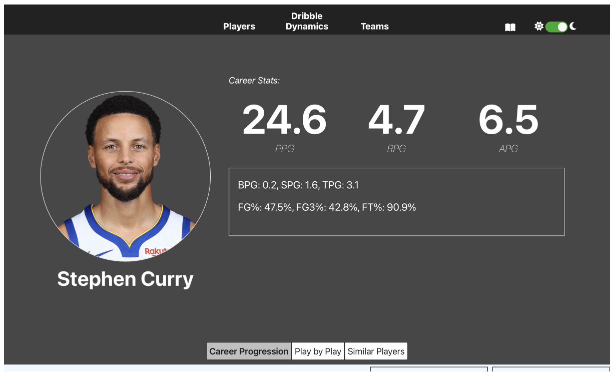
The player search page (see Fig.5) will be focused on a list view containing all players' names, teams, and years played. There will also be interactive functionality for searching by name, sorting, and filtering by things like teams or years played.



Name	Teams	Seasons
Stephen Curry	GSW	2010-2023
LeBron James	CLE, LAL, MIA	2004-2023
James Harden	BKN, HOU, OKC, PHI	2010-2023
Grant Long	BOS, MEM	1989-2003
Eric Piatkowski	CHI, HOU, LAC, PHX	1995-2008
Greg Anthony	CHI, MIL, POR	1992-2002

Fig. 5. Page providing options to browse the database.

Once a player is selected from the player page, the user will be brought to that player's individual profile (See Fig.6). At the top it will consist of all their relevant career stats (of which the NBA has a lot of). This will act as their main profile. Then, below will have expandable options for plots and graphs of the player's statistics, broken up into three categories: Career Progression, Play by Play, and Similar Players. A thing to note here for users is that all the following features are focused around their selected player and it remains consistent throughout their navigation in this page. For example in the below fig player Stephen Curry is selected and his relevant career statistics are displayed.



Career Stats:

24.6 PPG 4.7 RPG 6.5 APG

BPG: 0.2, SPG: 1.6, TPG: 3.1

FG%: 47.5%, FG3%: 42.8%, FT%: 90.9%

Stephen Curry

Fig. 6. Individual Player Summary

The Career Progression section (see Fig.7) will contains line plots for stat progressions over their seasons played or a histogram showing how many career games they had measuring a specific level of statistic (both along with stat section for selecting stats i.e the information will change dynamically depending on selection made). A legend is also fixed here for users to easily differentiate between different stats.

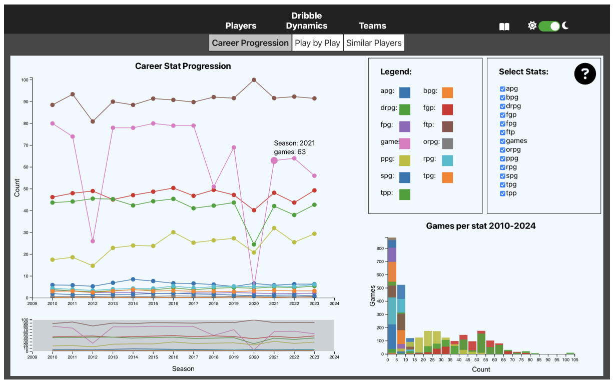
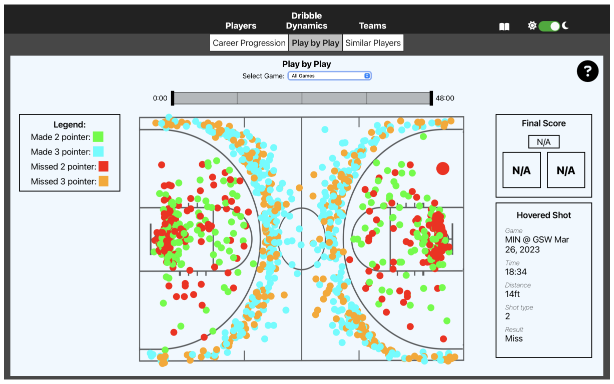


Fig. 7. Player Career Progression Plots

The Play by Play section (see Fig.8) contains a game visualizer where users can select a game played and navigate through the entire game time via time slider (temporal filter) highlighting each quarter as they drag. The main plot will show a plot of shots the player took at each point in time, of which points will be hoverable for more info. Users can also select 'all games' options in the dropdown to have an overview of all shots taken by their selected player in all games currently supported the portal. For simplicity again a legend is fixed on the left side showing color differentiation between 2 pointer and 3 pointer made or missed. We consider this section as the most visually engaging aspect of our project as users can gather a lot of knowledge about the game and player in terms of realizing trends or performance. In terms of data, this section takes by far the most space in our database (also the case in kaggle sources), hence current version only supports a limited amount of games for this feature and our database can grow significantly in size as we push more play-by-play data



Hovered Shot
Game
MIN @ GSW Mar 26, 2023
Time
18:34
Distance
14ft
Shot type
2
Result
Miss

Fig. 8. Interactive Game Play-by-play heatmap

The Similar Players section (see Fig.9) contains things like side by side radar charts for comparing players strengths and a cluster plot containing players within a close statistical proximity given selected statistics. This section is also a highlighting feature of our project and was praised during final project presentation. To generate this similar player clustering we make use of combination of techniques namely Principal Component Analysis + K-nearest Neighbours that we learned during this class (specifically in Data analytics Module : Extracting information from data, statistical learning, dimension reductionality). Once the user selects more than two stats in this section, PCA reduces the view to two dimensions and generates an aggregate plot for player of interest and similar players around him. When user hovers around a particular point in the plot is shows the overlaid radar chart comparison of player of interest and hovered player. On right-hand side below is the PCA loadings plots which dynamically shows how each selected stat contributes to the principal component (i.e. PCA breakdown Per Stat)

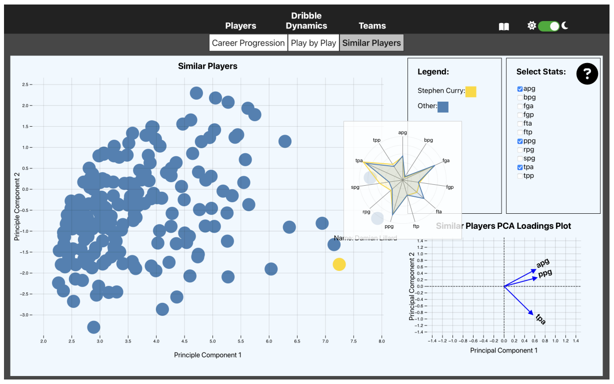
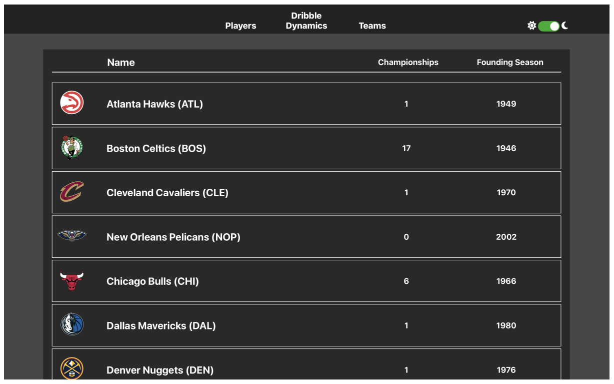


Fig. 9. Side by side comparison Radar charts and clustering

The teams section of our portal has two subsections just like player (see Fig.10). Current version of our application only supports navigating through the list of teams in the the history of the NBA and display basic summary statistics. This section is subject to more development as highlighted in the future work. Some ideas to include would be Team progression plots just like player progression, ability to create a fantasy team and judge overall offensive, defensive rating based on historical performance or create so called 'Keys to Success' feature which would highlight what adjustments to be made or small in-game goals to target in their upcoming matchup to increase odds of winning (e.g. minimum no. of assists required, double team opponent's star player, increase defense intensity in last quarter etc.)



Name	Championships	Founding Season
Atlanta Hawks (ATL)	1	1949
Boston Celtics (BOS)	17	1946
Cleveland Cavaliers (CLE)	1	1970
New Orleans Pelicans (NOP)	0	2002
Chicago Bulls (CHI)	6	1966
Dallas Mavericks (DAL)	1	1980
Denver Nuggets (DEN)	1	1976

Fig. 10. Navigating through list of all teams

Following is the complete list of interactivity features:

- **Home Page**: Players and Teams segment selection dropdown with light/dark mode switch
- **Players Page**: Player Search Bar, Sort filter - Relevance, Current to Next Page Button
- **Career Progression** : After Selection of Player - Legend of Stat Color, Multi-Stat Checkboxes, Question Button to initiate Current Page Tutorial, Hoverable Points for additional information, Plot Panning Scroll
- **Play-By-Play** : Dropdown menu for game selection, Temporal filter to slide through all quarters, Legend of Shots, Hoverable

Points on map to expand details, Question Button - Initiate Tutorial
- **Similar Players**: 2D plot with panning and zooming scroll, Legend, Stat Selection Box, Hoverable points with overlaid Radar Chart
- **Teams Page**: Dropdown for team selection and Team Stat

II. Project Summary and Evaluation.

In conclusion our system yields a multitude of interesting and useful facts depending on the features used:

1) We were able to successfully develop and integrate the main visualization features of our project in Player Career progression, Play-by-Play performance and Similar Player Clustering as mentioned in Stage 2 based on the techniques we learned throughout our course.
2) The web application is intuitive, responsive and easy to use to wide variety of users we claim to support.
3) We support the data of almost all 5000 players throughout the history of the sport and have potential to support all individual games data depending on the performance of the hardware infrastructure we utilize.
4) We are able to address all the fundamental questions proposed at the beginning of the project as - we enable the users to gauge the performance of their player of interest through variety of stat selection and dynamic analysis on top of them; from mere numerical data enable a visually rich interactive play-by-play map so users can gain 'not so apparent before' insights in an individual game and come-up with patterns or point out an adjustment to be made in a particular time segment (quarter) of the match; It also enables us to visualize how a particular player is evolving positively or negatively in terms of his shot selection; Through similar player clustering a user can quickly infer how a player is related/different from other players corresponding to a range of stats and make useful comparison to decide a potential matchup.

In our evaluation of our performance, we primarily focused on two things: algorithm efficiency and user experience. Under algorithm efficiency, we performed quantitative analysis on statistics like response times and query times. Under user experience, we trialed with and surveyed some of our acquaintances to get an idea of how an unfamiliar party interacts with the application.

Efficiency: Our initial goal was for our response times on any query to be less than 2 seconds. In our final product, we found that our queries on average took only fractions of a second, with no cases extending beyond 1 second response times.

User Experience: From user testing, we found that user response to the product was mostly positive. Users found the UI comprehensible and user-friendly. However there were occasional points in which we had to explain certain aspects. For example, some users were unfamiliar with PCA so we had to explain that aspect to them. So, going forward, we would likely want to include some kind of detailed breakdown of each chart and their purposes in the app. Our users were all familiar with the NBA so they were able to understand the data well and it didn't take long before they were examining their favorite players. Overall, users reported that they found the data and visualizations insightful and appealing. We didn't have time to do formal surveying but, from the general response, we found our results a renounding success.

III. Future work.

Following is the scope of future improvement we can undertake for this project:

- Integrate our large diverse dataset to a capable and powerful enough cloud service for even faster processing of huge data than what we were able to showcase.
- Port our back end to a production server running on Cloud
- Add the ability to automatically update the database periodically from official API
- Development required in the team segment of the project as described in Stage 4 and extend our analytics to team based profiles.
- More sophisticated library can be used to visually enhance shots heatmap.
- Potentially enable even more filtering and control for users visualizing similar player clustering.

References

[1] "Client package for nba api." [Online]. Available: https://github.com/swar/nba$_api$

[2] "Nba database." [Online]. Available: https://www.kaggle.com/datasets/wyattowalsh/basketball/data2

[3] "Nba player stats since 1950." [Online]. Available: https://www.kaggle.com/datasets/drgilermo/nba-players-stats/data

[4] "Nba shot logs." [Online]. Available: https://www.kaggle.com/datasets/dansbecker/nba-shot-logs

[5] "Nba player shot dataset," 2023. [Online]. Available: https://www.kaggle.com/datasets/dhavalrupapara/nba-2023-player-shot-dataset/data

[6] "React js client side." [Online]. Available: https://react.dev/reference/react

[7] "Dynamic and interactive visualization using d3 javascript library." [Online]. Available: https://d3js.org

[8] "Flask server python - server side." [Online]. Available: https://flask.palletsprojects.com/en/3.0.x/api-reference

[9] "Mysql database server." [Online]. Available: https://dev.mysql.com/doc/