Building an NBA Team Using MySQL Database

Shagun Bhardwaj, Zackary Manfredi, McCory Marks, Akshay Pahade

Northeastern University, Boston, MA, USA

Abstract

Over recent years, almost every NBA team has access to sophisticated tracking records that can enhance them with the position of the ball and can also give them information on how the players react to different situations almost every second of the game. The data which is made available by using advanced technology and analytic tools has started to revolutionize the game of basketball, influencing everything from game strategies to trade offs and how different players play under different conditions, coaches, and teams. Fans and sports analysis are taking advantage of the available data to either built their fantasy team or for different analytical needs.

The NBA has changed drastically in the last decade or two. We've seen the rise of different team strategies, including stockpiling assets (draft picks), strategic contracts (reserving max space for when a superstar is going to become available 3 years down the road), and tanking (purposely losing as many games as possible to maximize the chance of getting a very high draft pick. It is important to remember that the NBA is a business and many team owners and General Managers often base decisions around simply how they can turn a profit for that season. The Philadelphia 76ers were sort of the pioneers of the tanking strategy in which they were a horrible team for a few seasons, drafting very talented players and simply benching them. Ticket sales collapsed until eventually the pieces came together and they were a very talented championship contender. Season tickets then sold out in one year!

Our application will be essentially a team building application where the user can build the ideal team to reach their goal for the next season while staying within the salary constraints of the league. In a sense, the user fills a General Manager ("GM") role for their existing roster. What are our goals for next season? Which players available on the market would be a good fit to help us reach that goal?

Introduction

Data analytics had played a large role in the NBA and the way teams approach roster building today. Traditionally, NBA rosters were built around two specific types of players: 1)a dominant big man, traditionally a center or 2) a ball dominant wing scorer. This is shown throughout NBA history, as players like Kareem Abdul-Jabbar, Shaquille O'Neal, Kobe Bryant, and Michael Jordan have all won multiple championships. The rise of data analytics showed that the three-pointer was a more valuable shot, on average, than a midrange two-pointer. NBA teams now focus on spreading the floor and shooting more threes than before. This is just one example of how data analytics have influenced the NBA.

This project focuses on analyzing the goal of the GM of the team for the upcoming season and then recommends him the best fit player for a particular position considering his roster and the salary cap as well. The user can input the players the team currently has or start from scratch with a hypothetical team, and the application recommends players that are available that you should add to your team based on which ones could potentially add the most value. Recommendations include players who are signed to another team (trade targets) or players who are free agents (signing targets) - based ultimately on their contract. User will enter some goal into the application based on what they want their team to do in the next season (or even thinking ahead multiple seasons!) such as: winning the championship, tanking (maximize the amount of losses) and getting better odds for a top draft pick and developing young talent (can we sign cheap, young players who have shown some improvement in recent years and could be developed into superstars)

System Architecture

We used MySQL to store our NBA database. We chose MySQL because that is the DBMS we are most familiar with, and that is the one we learned during this course. We used Java JDBC to connect to our database and allow us to access and alter the data. We used JavaFX to create the GUI for our program to allow the user a more interactive interface.

Database Design

Having total of 16 tables and connection between was one of the critical tasks of this project. Our main focus was on the player statistics including the history of the player as these were the entities which majorly contributed in the output. Tables such as coach were also included to get the results such how does a player performs under a particular coach or also if the playing style of the coach and the player matches. Tables such as city and stadium were also included to get some insights how well certain players perform in certain stadiums or how well they are suited to the weather of the city.

The list of tables that comprises the overall design of the database starts with the player table that holds the attributes of each individual players like their primary identifier, player name, age, height, weight, experience in years playing in the league, and status determining if the player is active or retired in the league. This table contains a foreign key that is linked to the position table that holds the position name and abbreviation that was helpful in determining what stats should be highlighted for each position in the algorithm to help with the team's goals for the year. The other foreign key was linked to the team_id that is associated with the team table that consisted of the team name, championships won, and its most recent playoff appearance if there was one. The team table is then linked through a foreign key to its respective stadium that they play in whose table holds the stadium name, its approximate capacity that is gauged approximately using the enums small, medium, and large and the crowd noise, which also uses enums medium and loud. The stadium table is linked to the city table that holds the attributes of the city they play in that can play a factor to the player's performance on the team such as the climate and the market size and the overall popularity of the sport in the city.

We also have to take into account the history of player injuries as they can have a large correlation to the overall impact of a player's performance for the upcoming season. There is an injury table that holds all the injuries that professional basketball players typical can fall victim to in play and its severity that rated from a scale of 1 - 10. We created a link table that highlights the many to many relationship between the player and injury tables using each of their primary keys in their respective tables and will hold attributes like if surgery was required and how many games were missed due to the said injury.

The stats tables that include seasonal, playoffs, and advanced metrics hold the vast majority of the data compiled from stats.nba.com. The season table holds the season year and the player_ids of who won the MVP of the season and the MVP in the finals and the team_id of the championship team. The season and playoff tables were comprised of almost the same stats, except with the seasonstats table holding the salary that the player is making that would contribute to the salary cap of the NBA teams. Some of the stats that were included were games played, minutes, points scored, assists, offensive and defensive rebounds, turnovers, steals, blocks, personal fouls, field goals, three-point three goals, and their respective percentages. For the advanced stats table, this consisted of mainly advanced metrics such as offensive, defensive, and net ratings, assist and rebounding percentages, offensive and defensive rebounding percentages, number of possessions per 48 minutes for the player and the player impact estimate (PIE). All three tables had foreign keys linking them to the corresponding player_id, season_id, and team_id.

The coach season table is essentially a link table that holds foreign keys for the coach, season, and team ids. The coach have a critical impact on how a player performs based on their past experience and if their playstyles complement one another. The coach table contains their name, age, and a grading system we have in place for their offensive and defensive strengths. It also provides the number of coach of the year awards one has been awarded. The final two tables deal with the awards of the player with a link table connecting the many-to-many relationship between the player, award, and season tables.

Data Acquisition

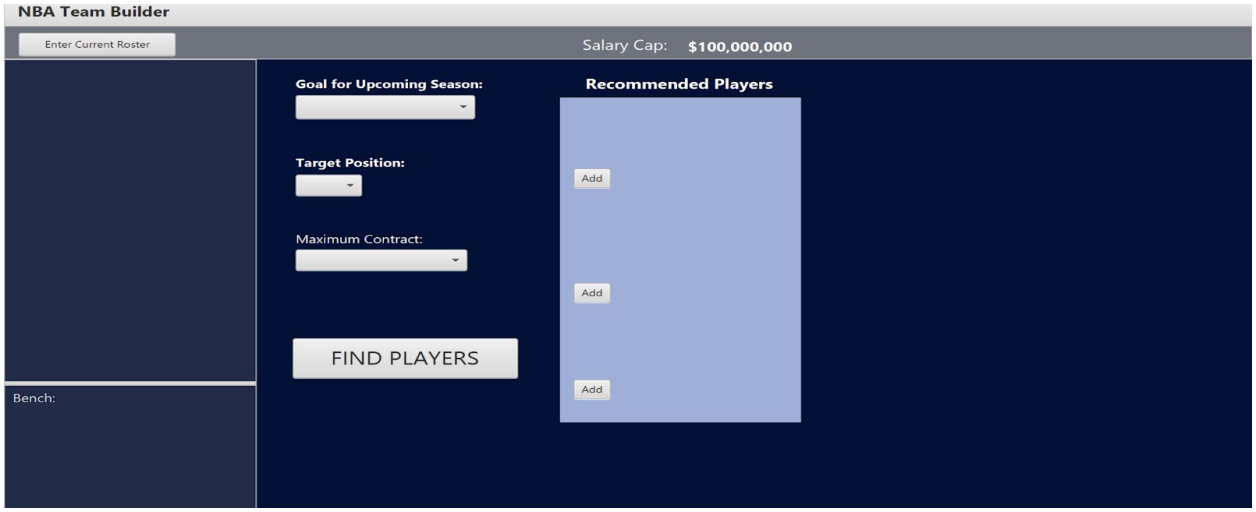
There are no currently ready-made data sets specifically aiming for our project. What we found is several websites on NBA games. Data sources from different websites had to be merged using invariable format. Different websites have player names and season labels differently. For example, Danuel House Jr. on stats. NBA and Junior Danuel House on basketball reference are in fact the same person from different players.

Though the data were already in table format on both the websites converting those into csv was an easy task using Microsoft Excel's inbuilt feature which scraps the websites table data easily but merging the data from two different websites to form different required tables was a difficult task, so we ended up using the player's name, team and season to join both of these datasets together. Some common problems we had were inconsistent spelling of names in both datasets. In the end we were able to sync most of the data from both the websites. That being said, we don't doubt that there is further cleaning left in the dataset.

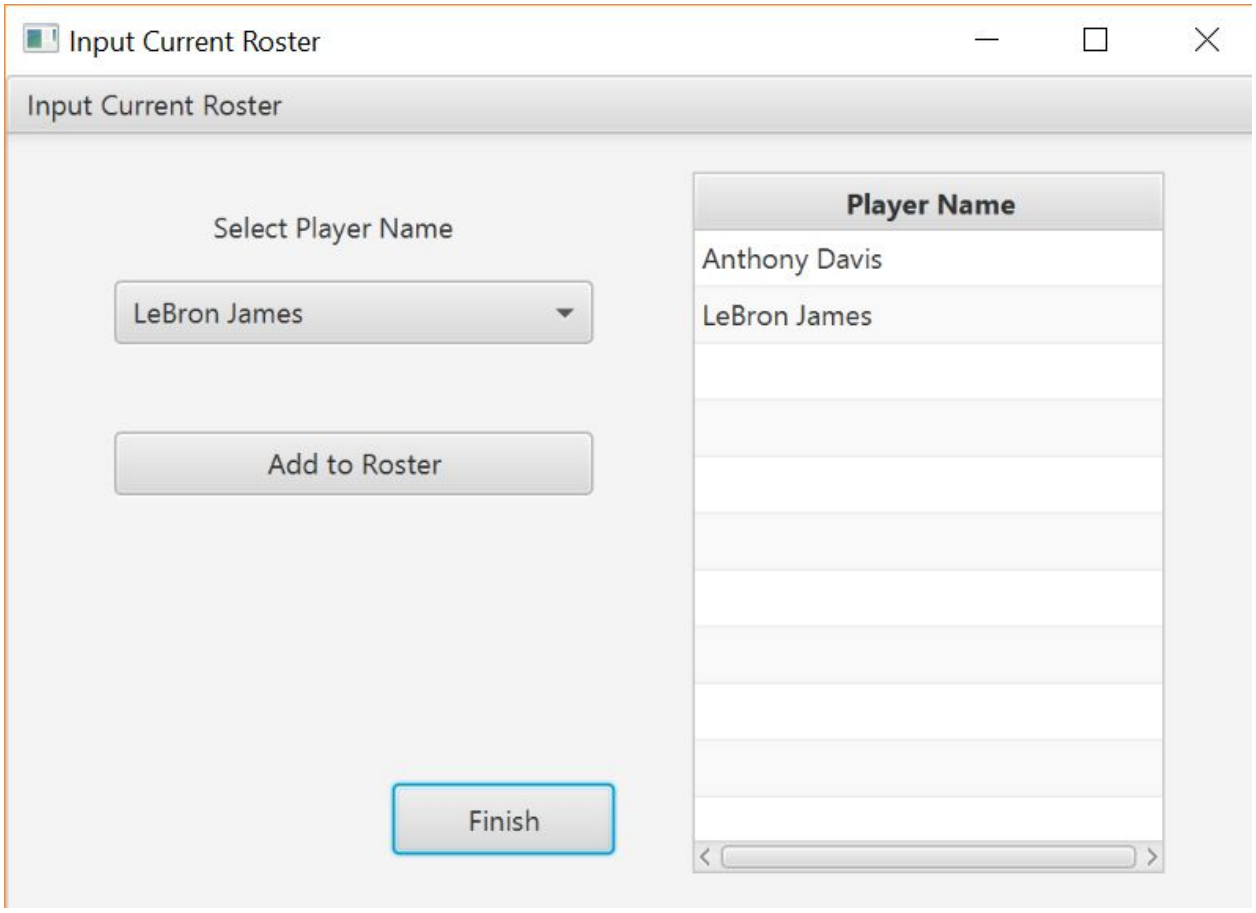
User Interface

For our application, the end goal was a graphical user interface which the user interacts with as the "GM" of their team. We used Java to create the GUI, as Java has some built-in tools that can help easily create a modular and responsive interface. In our case, we chose JavaFX, which is a bit newer and more advanced version of Java Swing. While our interface aesthetically is rather simple (none of us are experts in front-end design, after all!), it is responsive and interactive for the user.

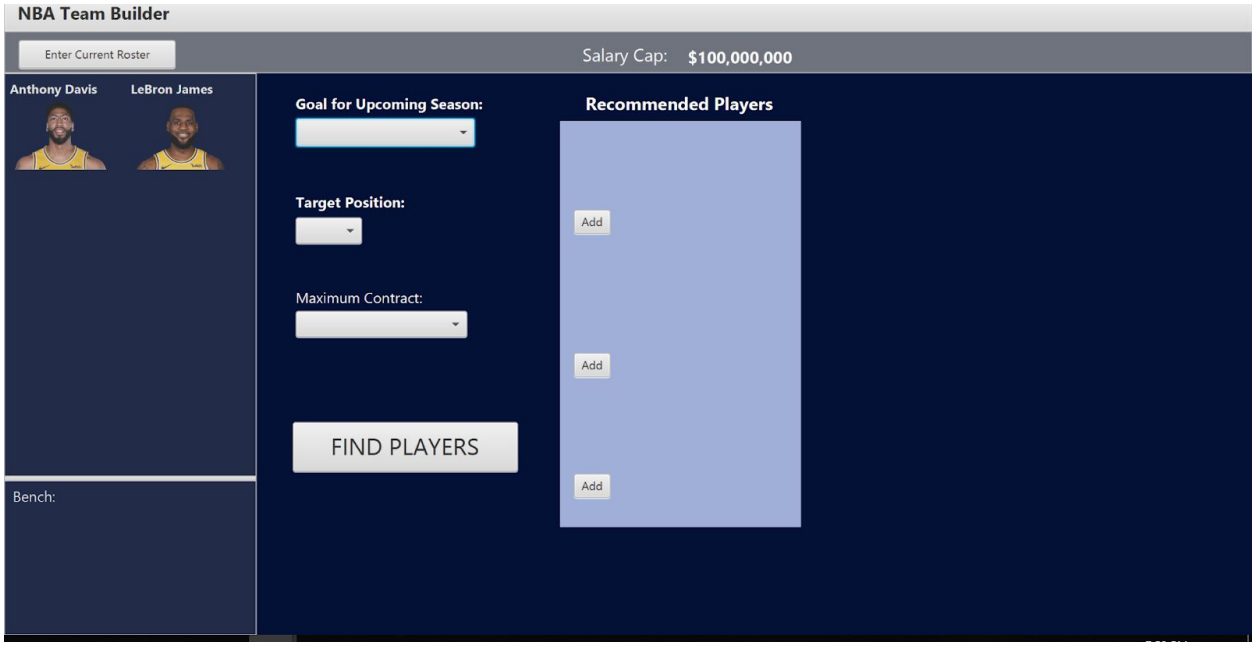
When first booting up the application, we launch the general main window, as seen below.



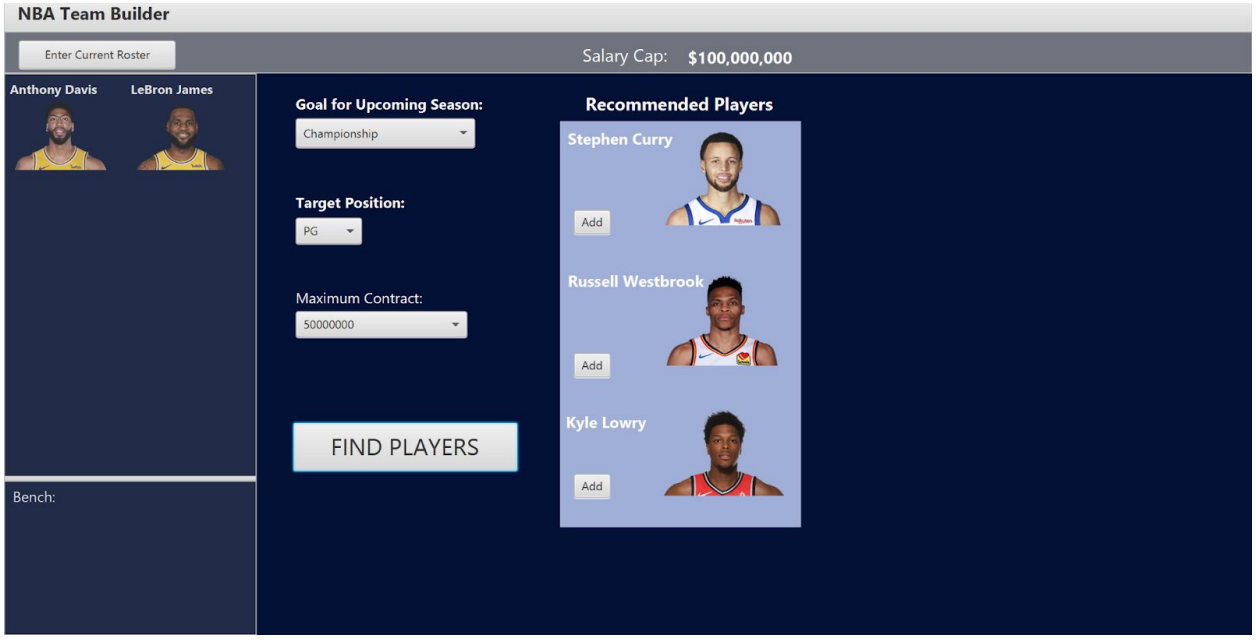
There are no players on the team if the user is just starting off, so the first option is for the user to click “Enter Current Roster”, where a second FXML loader will load a new window controlled by a second controller. Here the user has a list to scroll through and select and add existing players on the roster. Attempting to add a player twice will not do anything. As they enter players, they populate in the table so the user can keep track of who is being added.



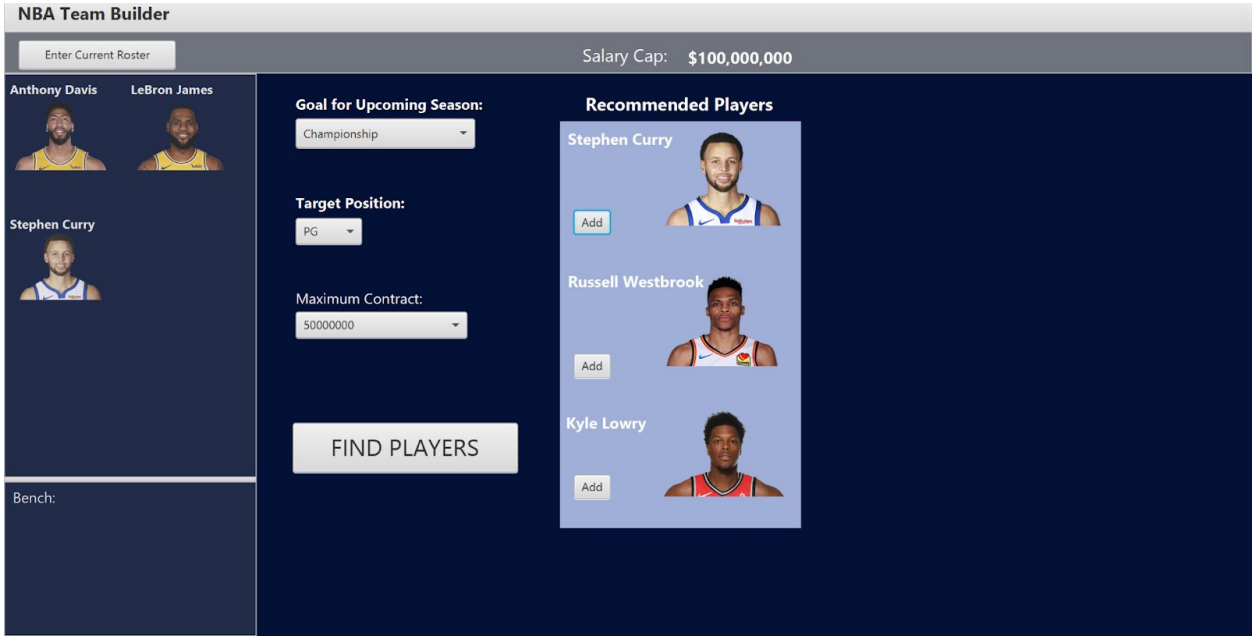
Now, the controller passes the inputted list of players back to the main menu window which will populate the left hand side with player names and images. We did not input all ~500 photos of players, however in theory the application would have a headshot for each player which would be added.



Now, the user has entered their current team and it's time to search for a new player. We have three main criteria, besides the current roster, as input for our algorithm. The user will enter the position they are looking for, their goal for the season, and the maximum salary range they are willing to pay. Clicking the "Find Players" button will then launch the algorithm. Once the algorithm has done its work, we populate the GUI with the top 3 options by value, along with a picture of the player.



The user can then choose to add one of the three players to add to their roster.



Ultimately, basketball is still a subjective game. There are intangible traits that are impossible to measure using quantitative analysis from a database, like player attitude and team chemistry. We wanted to leave some choice up to the user, because at the end of the day they might prefer the third option over what our algorithm deemed to be the top available player on the market. Once they choose one to add, the player is added to the current roster on the left hand side of the GUI. It's impossible to be 100% objective in our algorithm, as the stats that we considered valuable for a player given a team goal might not be the same as another individual.

Conclusions

In conclusion, we had a great time creating this project. In terms of scope, we felt our project was pretty expansive and bold, given just how complex the project could have become. There were aspects that we wish we could have added or improved given more time and experience. For example, much of the GUI (from an interactive and from an aesthetic standpoint) was new to all 4 of us, so learning how to create it added a lot of extra time that took away from the features we wanted to add for the final project. In terms of the algorithm, we are happy with how it came out and how it interacts with the database we created. As a few of us are big basketball fans, the "aha" moment of seeing the recommended players actually make sense in terms of the league today. As a GM, many of our pics would absolutely make sense to add to the team. A good specific example was adding the playoff stats table, which really helped us differentiate players based on the value they bring to the team that is looking to make a championship. Seeing those names pop up was a testament to all the work we collectively put in to make this thing work correctly.

In reality, the NBA offseason is very complex. We decided to forego the concept of trades versus free agency and just simply recommend players. Including contracts would have required lots of complicated salary cap considerations (it can get very convoluted in real life examples), and having trades would have been very difficult given we would have had to add a whole facet to our algorithm that would recommend which of the existing players (or future draft picks, or cash…) would be good to give up for the target player.

However, this does create some exciting opportunities in terms of future scope for the project. We have the foundation down, and the application could be expanded to include a much longer historical time frame, college players and draft picks, trades and free agency, salary cap and the luxury tax - the list goes on and on. Originally we had brainstormed ideas for simulating the results of adding a specific player to the team, but an NBA league simulator could be an entirely separate algorithm by itself. It would have been awesome to include an estimate for the team's record the next season with the new players added to the team! In the end though, we got to use our databases skills, learn a ton about how to have our database interact with a programming language like Java, and even graphical user interfaces - all of which is extremely relevant real world experience to have. And who knows, this is a project that we all had fun creating (but challenging at times as well) and would love to continue contributing to - hopefully one of us can show it off in an interview someday soon!

Author Contributions

All four members of the group contributed to the development of the overall design for the structure of the database in mySQL and the conceptual layout of the application in Java. Each of us were actively involved as we brainstormed ideas for the project. Given the NBA database applications have so much potential use cases, we all had to collectively narrow down the options before settling on the GM player recommendation algorithm. Akshay and Zack focused on gathering all the stats needed for the NBA database and integrating it into mySQL with the logical model including what table definitions were needed. Shagun wrote out the conceptual ER diagram on mySQL and implemented the java classes that were used. Shagun and Mac brainstormed the overall logic of the algorithm and Shagun implemented the three examples of team goals. Mac spearheaded the front-end GUI interface using JavaFX that takes in user input and returns the best-suited player and photo based on the team's goals for the season. Zack and Mac worked on creating the presentation and creating the status report and everyone contributed equally for final report. All four members collaborated equally to ensure the success of this project.