NBA Team Builder

Akshay Pahade, McCory Marks, Shagun Bhardwaj, Zack Manfredi
Northeastern University, Boston, MA, USA

Abstract

In the abstract, provide a summary overview of your project, its goals and accomplishments. For the mid-project status report, I don't expect this report to be complete, but I do want, at minimum, the sections on System Architecture and Database Design to be completed so that the TAs and I can begin to offer you technical feedback.

Introduction

Explain what your project is about and its significance and the main features or use-cases. This text can be drawn extensively from your project proposal but include any updates that reflect instructor feedback if needed.

- 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. 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.
- Also lets you pick a coach - based on their style. This will further factor into our algorithm as we analyze player performance under different coaches with different playstyles.
- Recommendations include players who are signed to another team (trade targets) or players who are free agents (signing targets) - based ultimately on their contract.
- Undecided - luxury tax? You can go over the cap to resign your current players, but you cannot go over it to sign free agents or in a strict trade (if already over the cap, you can take back as much salary up to the current; if not over the cap you can only take salary up to the salary cap)
- 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!)
    - Win the championship!
    - Tank (maximize the amount of losses) and get better odds for a top draft pick

- Make playoffs (even extending the season by a few games can be a huge revenue booster for teams who might not make the playoffs often, or are right on the cusp!)
- Make conference finals
- Win your team's division
- Develop young talent (can we sign cheap, young players who have shown some improvement in recent years and could be developed into superstars)
- Make space for max contracts (what the modern NBA is seeing a lot of is teams structuring their roster based on maximizing how much cap space they can spend in the specific years when highly coveted players become free agents)
- If it's a new team, maybe in its first or second year, the goal wouldn't be to win a championship but it might be just to win more games than last year

System Architecture

Describe and diagram the architecture of your application and how different components communicate with one another. Include, for example, application components, data sources, databases, web-servers, application servers, etc. What we want is a high-level summary of how your project is intended to function as an integrated system.

Application components:

Our application will be a GUI that the user interacts with. They can click and add existing players to their team and the GUI will display the current team lineup as well as a small picture and position for each player. The current team, and as players are added, will be on the left hand side of the application. Once the user has set their current team, or if they are building a team from scratch, the application will query the user on a few different items (what is the salary cap for the NBA this year, what position is missing, who is the coach of the team and what is the user's team's goals for the year?).

Based on the current team lineup and the user's inputted answers, our application runs an algorithm that feeds off our database in the background and returns a handful of players that we recommend to add to the user's lineup. Our GUI and underlying algorithm will be written in Java and use the inputted information to interact with the database in the background. To do this, there is a MySQL Java Database Connectivity Driver we will install that will handle most of the communication. Originally we were planning on using Python given how powerful language is for data analysis, however upon narrowing down the application function to the teambuilder, we realized that the way the user interfaces with the database is very important and we have decided to use Java instead as its more apt towards building a GUI.

Data sources:

Given the NBA is one of the most popular sports in the world, there is a massive amount of data available on the internet for player stats over time, and much of it can get as detailed as we want. We have a general idea in our database design of the relevant data we will pull for the players in our league in order to do our analysis and recommend players for the user's team, but in theory we could get as in-depth as we'd like.

Database Design

Explain your database design. Include a figure with your ER Diagram / Conceptual Model. Explain the *key* entities and their relationships. Use the MySQL modeling tool to address both these requirements. We'll assess the quality of your design and its effectiveness in addressing your project requirements. We may offer suggestions to improve your design. If you are using a non-relational database like Mongo, explain your document model including collection structure, document layout, fields, etc.

- PLAYER
    - One of the core tables of our database, players are what our application revolves around. We will be analyzing player performance compared to other players at the same position, and make recommendations to the user based on a ton of factors, including how old the player is, how long they've been in the league, where they're from (some players play much better in their hometown!), the player's injury history, how the player is trending in terms of his statistics, and more. We are structuring our database so that over time, new players who get drafted can be added as well as add their performance and stats as the seasons go by.
    - Player_id; int -> PK
    - player_name; varchar(50) -> NN
    - birth_date; date() -> NN
    - Height; float -> NN
    - Weight; float -> NN
    - Experience; int -> NN, default 0
    - Reputation; varchar(50)
    - FK position -> POSITION table
    - FK team_id -> TEAM table; this attribute is not required as a player can be a free agent
    - Status; tinyint (either active or inactive)
- TEAM
    - Team_id; int -> PK
    - Team_name; varchar(45) -> NN
    - FK city_id -> CITY table

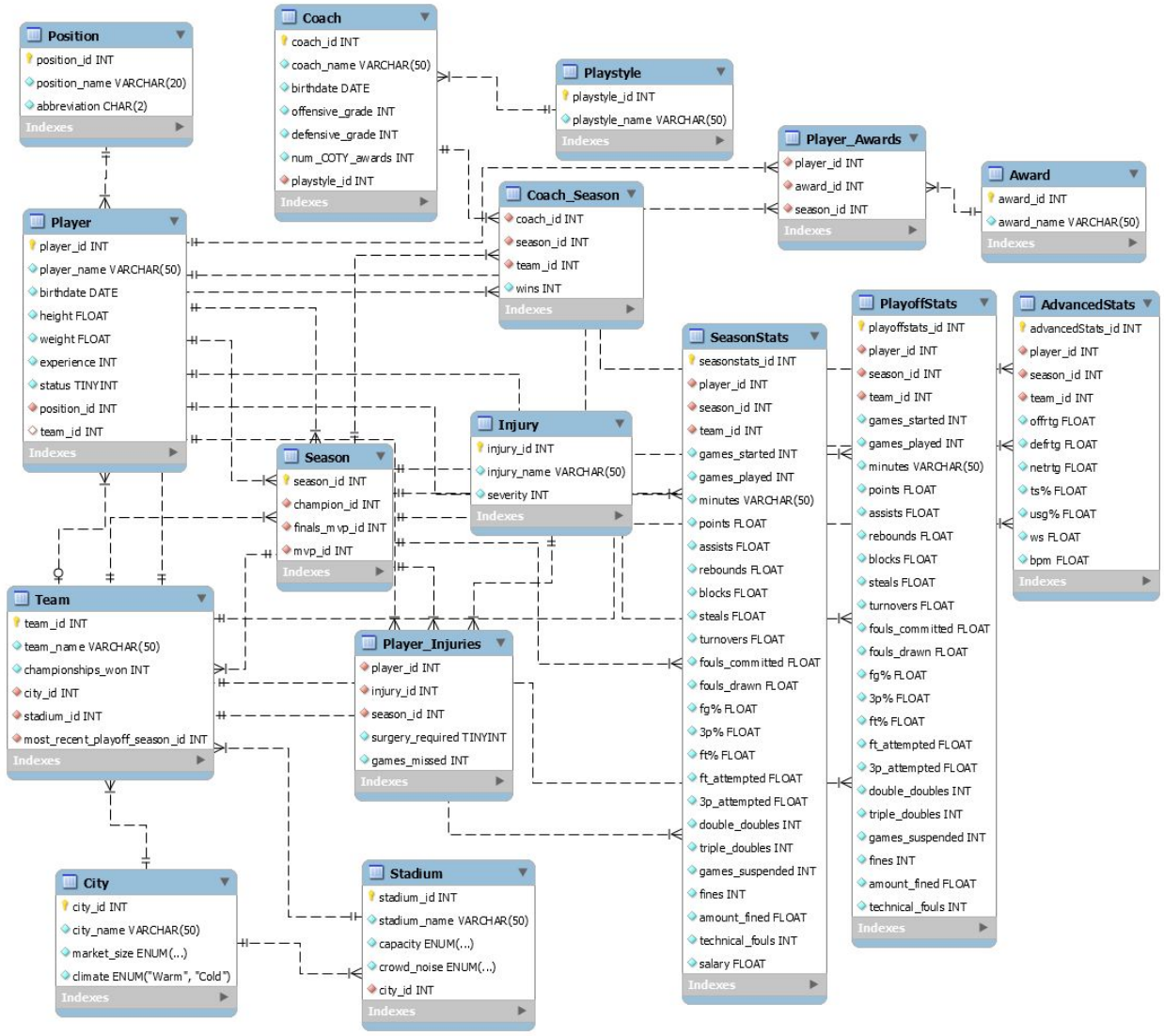
    - Championship_won; int -> NN, default 0
    - FK most_recent_playoff -> SEASON table
    - FK stadium_id; int -> STADIUM table
- COACH
    - An important part in building a team is the coach picked to run the team. Different coaches have different playstyles and some are more offensive-minded while others are more focused on defense. In the SEASON STATS table, we track how each player performed in a given season statistically and which coach they player for. Given the user knows who their coach is and what that coach's playstyle is, our algorithm can select and recommend players who perform above average (or have their best seasons) while playing under a certain playstyle. This is going to be one of the most important variables in our algorithm as it might recommend a much cheaper player who might actually be a better fit. The benefit of that would be maximizing the remaining salary space while adding a player who is a great fit.
    - Coach_id; int -> PK
    - Coach_name; varchar(50) -> NN
    - birth_date; date() -> NN
    - Offense_grade; int -> NN, min 1 max 10
    - Defense_grade; int -> NN, min 1 max 10
    - COTY_awards; int -> NN, default 0
    - FK playstyle_id -> PLAYSTYLE table
- PLAYSTYLE
    - Style_id; int -> PK
    - Playstyle_name ; ENUM
        - Space and pace
        - 7 seconds and less (hyper tempo)
        - Grit and grind
- SEASON
    - Year; int (might change to year) -> PK
    - FK champion -> TEAM table
    - FK finals_mvp -> PLAYER table
    - FK mvp -> PLAYER table
- STADIUM
    - We will be accounting for how certain players play in different stadiums. Between two players with similar stats, one might play much better in a large market arena like New York in the spotlight with a loud crowd, while another might prefer smaller, quieter crowds. There is information available on how different players play in certain arenas, and how crowded and loud those arenas are. The user knows the team they are representing and the arena that team plays in, and can factor into which players they might like to have play for their team.
    - Stadium_id; int -> PK
    - Capacity; ENUM -> NN

        - Large, Medium, Small
    - Crowd_noise; int -> NN
        - A scale on how loud the arena is
    - City_id; int -> FK
- POSITION
    - Position_id; int -> PK
    - Abbreviation; CHAR(2) -> NN
- AWARD
    - Player can have more than one award
    - Award has a season_id as well
    - Award_id; int -> PK
    - Award_name; varchar(50) -> NN
        - MVP
        - MIP
        - DPOY
        - ROY
        - All-Star
        - 1st Team All-NBA
        - 2nd Team All-NBA
        - 3rd Team All-NBA
        - 1st Defensive Team
        - 2nd Defensive Team
        - COTY
        - 6th MOTY
- JOIN PLAYER-AWARD
    - Players who have won multiple awards, like making the All-NBA team multiple times, will be considered much more valuable by the algorithm than those who have not.
    - FK player_id -> PLAYER table
    - FK award_id -> AWARD table
    - FK season_id -> SEASON table
- JOIN SEASON-STATS
    - The season stats table will be the most important table in our database, as from it we will draw analysis on each player to recommend to the user. It will include historical data, allowing us to analyze trends (is the player getting better at 3 point shooting, or more efficient by turning the ball over less, or injured more). We will be identifying players to target based on their position and the position needs of the team, as well as the coach and his/her style. To narrow down the possibilities in our recommendation algorithm, we will likely be using the season stats table to find how a player compares to the average player at their position, the top 10% at that position, etc.
    - Season_id; int -> FK -> the actual year
    - Player_id; int -> FK -> player

    - Team_id; int -> FK -> the team they played for
    - PPG; float -> NN, default 0
    - APG; float -> NN, default 0
    - RPG; float -> NN, default 0
    - BPG; float -> NN, default 0
    - SPG; float -> NN, default 0
    - MPG; float -> NN, default 0
    - Turnovers; int -> NN, default 0
    - Free_throws_taken; int -> NN, default 0
    - Free_throws_made; int -> NN, default 0
    - FFP; float -> NN
    - FG%; float -> NN
    - 3ptrs_taken; int -> NN, default 0
    - 3ptrs_made; int -> NN, default 0
    - 3P%; float -> NN
    - Fouls_committed; int -> NN, default 0
    - Fouls_drawn; int -> NN, default 0
    - Double_doubles; int -> NN, default 0
    - Triple_doubles; int ->> NN, default 0
    - games_suspended; int -> NN, default 0
    - Games_played; int -> NN, default 0, max 82
    - Technical fouls; int -> NN, default 0
    - Coach_id
        - This will tell us a lot about what playstyle a player played well in
- ADV_STATS
    - Advanced stats are readily available on the internet and go beyond the basic, objective statistics that are observable in the game. There are players like Al Horford, who might not be the flashiest or highest scorers, but are so fundamentally important to the team and the team is much more likely to win when they play. The advanced stats will be useful in our algorithm in differentiating between a handful of players who we have used their normal stats and history to identify. As the user, if we want to add a player to the team and stay within the salary cap, advanced stats can help identify areas where a player can contribute to the success of the team, while staying within that salary cap threshold.
    - Offense_rating
    - Defensive_rating
    - Net_rating
    - true_shooting_%
    - player_efficiency_rating
    - Win_shares
- CITY
    - City_id int; PK

    - City_name; varchar(50) -> NN
    - Market_size; ENUM
        - Small market, Medium market, Large market
    - Climate; ENUM -> NN
        - Warm, Cold, could add more here
- INJURY
    - Injury table is another important factor in building a team and identifying players to pursue in the offseason. We include the severity of the injury as there are some injuries (ruptured Achilles, for example) that can materially impact a player's performance when they return. Some players might not be the same after returning from a major injury, and combined with age that can be a factor in whether or not a player is recommended to sign to the user's team.
    - Injury_id; int -> PK
    - Injury_name; int -> NN
    - Severity; int -> NN, default 1 (lowest severity)
- JOIN PLAYER-INJURY
    - FK player_id -> PLAYER table
    - FK injury_id -> INJURY table
    - FK season_id -> SEASON table
    - Surgery_required; tinyint -> NN, default 0
    - Games_missed; int -> NN, default 1
- PLAYOFF_STATS
    - Based on the user's goal for the season, they may want to make the playoffs, advance to the conference finals, or win the championship. In situations like these, part of the recommendation algorithm will factor in how much experience a player has in the playoffs - this can be a huge determinant in success. Someone like Lebron James or Kawhi Leonard who have brought teams to the playoffs year after year, and won multiple championships, will be far more valuable on a team trying to win the championship than a player who has very high stats during the season but rarely performs well the playoffs.
    - Player_id
    - Season_id
    - Games_played
    - All the regular stats that we included in our SEASON-STATS table….
- JOIN COACH_SEASON
    - FK coach_id
    - FK season_id
    - FK team_id
    - Wins; int -> NN; default 0

ER Diagram:

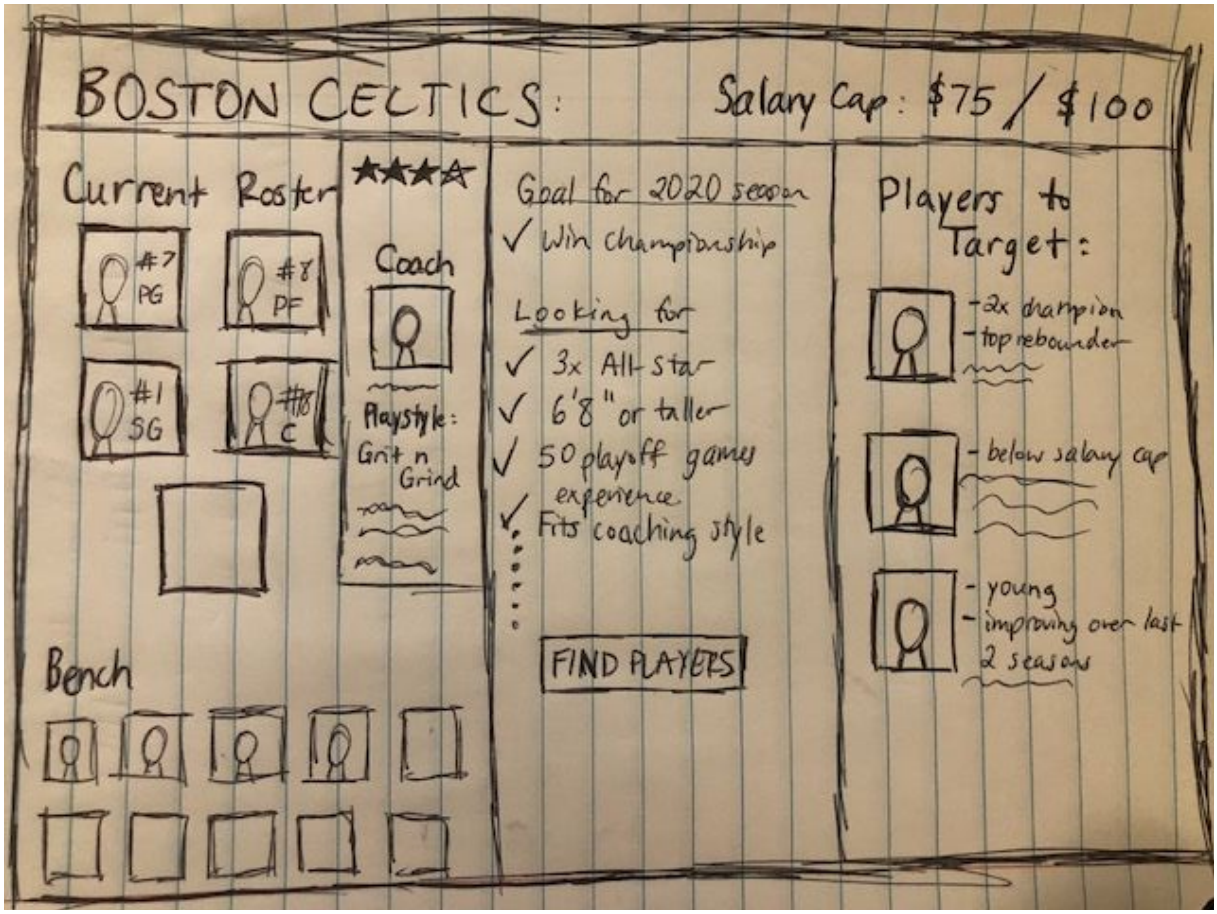


Data Acquisition

Explain step-by-step how you acquired your data including all data sources. Use numbered citations like this [1] or like this [2, 3] and list any references at the end of your report following a consistent style. Describe any work you did to modify or clean the data prior to being loaded into the database. If you made up your data as part of an application prototype document any assumptions that may have been built into the data-generation process. Provide sufficient detail to enable the reader to reproduce your results.

User Interface

If you created a proof-of-concept application then describe your user-interface and its capabilities, use-cases, etc. Include one or two screen shots that conveys to the reader what it is like to use your application. For the status report, it is sufficient to include mocked diagrams or even a hand-drawn sketch.



Analysis and Results

Those of you doing a data-analytics-type project should present the results of your analysis here. Include charts, graphs, and other visualizations that demonstrate key insights.

Conclusions

Summarize your results. Be concrete about your accomplishments as well as what perhaps didn’t go so well.

Author Contributions

Describe how each member of your group contributed to the success of your project. There are many ways to make meaningful contributions to a project. I don’t expect each person to contribute to each aspect of the project. Some of you are more experienced web-developers, others make tackle the database design, or you may be primarily responsible for creating the class presentations and the writing of this report.