Team 9 Final Project Report

Graham Wood, Mitchell Blazer, Tanner Bayless, Tyler Trammell

Web Address: <http://nbamanager.azurewebsites.net/>

**Introduction**

For the Group Final Project, we decided to make a web application that gives copious information about the current (2018-2019) NBA season. With this being said, the intended user(s) of our application will be people who are involved with the league itself -- coaches, referees, commentators, NBA staff, the commissioner, etc. The scope of this application is solely the current season, meaning that it will not provide information about past seasons and the history of the sport. The functionality of the application is fairly simple. It has a homepage with several different tabs that differ based on our relations from the database design (teams, coaches, arenas, players, games). This allows easy functionality because it allows the user to easily maneuver to a new webpage if he/she wants information on lets say a team versus information on a player. Each of these pages that branch off of the homepage will have their own abilities for the user.

The “Teams” page will allow the user to do three different things. The first option is a button that, when pushed, displays information (columns in tables) about each team in the NBA (“Show All Teams”). Another option is the the user to use the dropdown to do a search on a team. Once the user selects a team, he/she will see information relevant to that team and that team only (essentially one row of a table is displayed). The final option, which also depends on the user selecting a team, allows him or her to see the roster for the selected team. This roster will consists of all the players on a given team, with information about the players (such as name, position, etc.). This roster and team information is scoped in such a way that assumes the trading that will be taking place (updates) will not occur in season, thus, there is no update option for a team.

The next page option is for lookups and information on the coaches. The first thing a user can do is click a button to show all coaches. This could allow for easy comparisons for whoever is using the application. When displayed, various information will also be shown about the coach. The user can also do a lookup. For ease on both the user side and the system side, we added another dropdown list with every coach currently in the NBA. When the user selects a coach and clicks the “Search” button, they will get information about that coach. This is another page that is going strictly off of the current season information and does not include coach hirings/firings (updates).

There is also an “Arenas” page which is equally as simple. The user can do a couple of different things on this page as well. The first thing a user can do is click a button to show all the arenas in the NBA (30 of them). This information could be valuable in the case that the user wanted to compare two or more stadiums (such as capacity, number of games played there, etc.). If the user wanted specific information about just one arena, he or she could search a specific arena using the dropdown listbox. These two functionalities are simple yet very useful to the NBA league, which is our primary intended audience.

A “Players” page allows the user to perform two different operations. If the user is wanting information on a certain player. He/she can search a player by first or last name and get information on that player(s). This could be very useful for people like commentators who want some quick information on a specific player. Another thing a user can do is add a new player to the list. This could happen if a player gets called up from the G-League -- essentially NBA’s “minor leagues” -- to the actual NBA Team. This could also be very useful at the very beginning of the season when there is a draft. The user can insert the player into the database to be used for later use. If the user wanted to see a list of players, he or she will have to go to the “Teams” page and look up a roster.

The last page of the application will be the “Games” page. This page will have three different functionalities for the user to try out. The first thing he/she could do is search a game by a certain arena. This is equivalent to looking at all the games for a home team (granted that the user knows what the arena name is of the home team). A user can also search all of the games by a certain team. This will give a list of games, both home and away, for the specified team. Both of these searches, the search by arena and search by team name, will be chosen by the user in a dropdown list box. This will once again, make it easier on the system and on the user to avoid typos. The last functionality of the “Games” page is for a user to add a game. This is especially useful for people like the NBA staff as to when it is midseason. This will allow the user to add a game with various information to the database. This can then be used later on, by anyone else using the application.

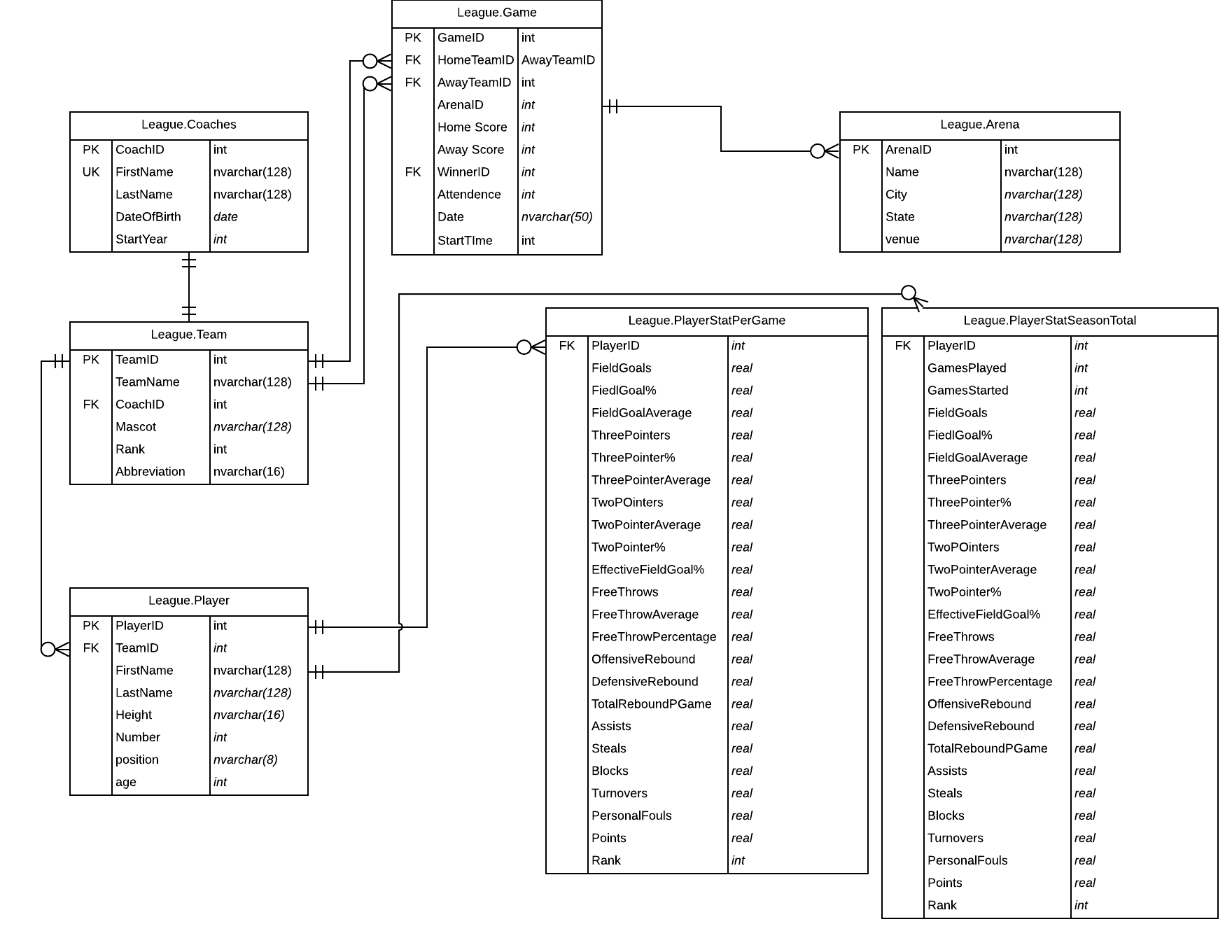
All this information is a brief overview and introduction as to what our web application can do (functionalities) and who the intended users are. Although simple, it could be very useful to its users for quick lookups and information about the National Basketball Association.

**Technical Description**

As briefly mentioned in the “Introduction” of this report, this project is a web application. The system was built using in the Visual Studio 2017 environment as a Microsoft ASP.NET webform application. This coupled nicely with where we created our SQL database, which was coined “NBALeague”, and was stored on the Microsoft Azure cloud computing service. The “NBALeague” relation itself was built (obviously) using the SQL language and in Microsoft’s SQL Server Management Studio. Keeping everything under this Microsoft umbrella kept the application very clean from an implementation perspective.

The Application connects to Microsoft Azure through Visual Studio’s built in SQL Server Object Explorer that is compatible with Azure. This is how we connect that application to our “NBALeague” database. Our procedures, which contains the primary functionality for our web application, was written in SQL also in SQL Server Management Studio. By creating these procedures as objects, they were placed inside a folder in SQL Server Management Studio that could be accessed by programs -- in other words, they became programmable objects.

The front end and back end were both written in the Visual Studio environment. The front end, or the graphical user interface, is written in HTML and CSS as these are the primary languages for web development. The HTML is for the content on the web page as the CSS is for the styling and actual aesthetic. The back end takes care of when a button is pressed or an event needs to occur. To accomplish this, we incorporated C# ASP.NET. The C# takes care of taking the information in from the user and then calling the database procedure/function to then display the information back to the user on the user interface (web page). One tool/library we used was a Bootstrap template. This template encorporated CSS and HTML that gave us the simple webpage design that we use -- the banner on the top with links for each page. The C# code worked in conjecture with the HTML/CSS to give use our final product. This template came in really handy as to give us a place to start for a GUI. We used default libraries for the back end of the application.

**Database Design**

**Database Creation**

The creation of the database had us worried at first due to the massive amounts of data. After a small amount of time doing some research we found how to take advantage of SQL Server Management Studios build in import and export system. This is what we used for about 99% of the data input. The biggest challenge with using the tool was getting the data to be formatted so that the tool could load it in properly. The use of notepad++ was used heavily here to help delimit the data read in so that it could be inserted into the appropriate columns into the database. An example of how powerful this too is was the first and last names for instance, the name was formatted as one with first and last from the initial dataset. Using notepad++ allowed me to separate the first space in the row with a single comma. Using this technique saved us a lot of time and resulted in easy data input. Going into the creation of the project my initial thought that it would have been mainly populated with the use of INSERT statements.This was not the case however as we soon found how easy some of the data entry could be handled through CTE’s and Merge statements as learned in class. Some of the queries that were written had to be quite extensive as well to ensure the desired data was inserted into the table. One of these harder solutions was when inserting player’s individual stats. The simple idea of putting a player identification number into the table soon became a big issue. This would be easy, but the players table considered for players that had switched teams during the season. The way this was handled in the player table was a team name TOT that was their accumulated season information. Each player could now have multiple player ids based respectively on whether they had switched teams or not. This was handled in the stats table’s by finding the minimum id for a subset of players containing distinct names and returning the correct id representing their total stats. Besides some of the outlined issues the data input and creation of the database was pretty straightforward.

**System Design**

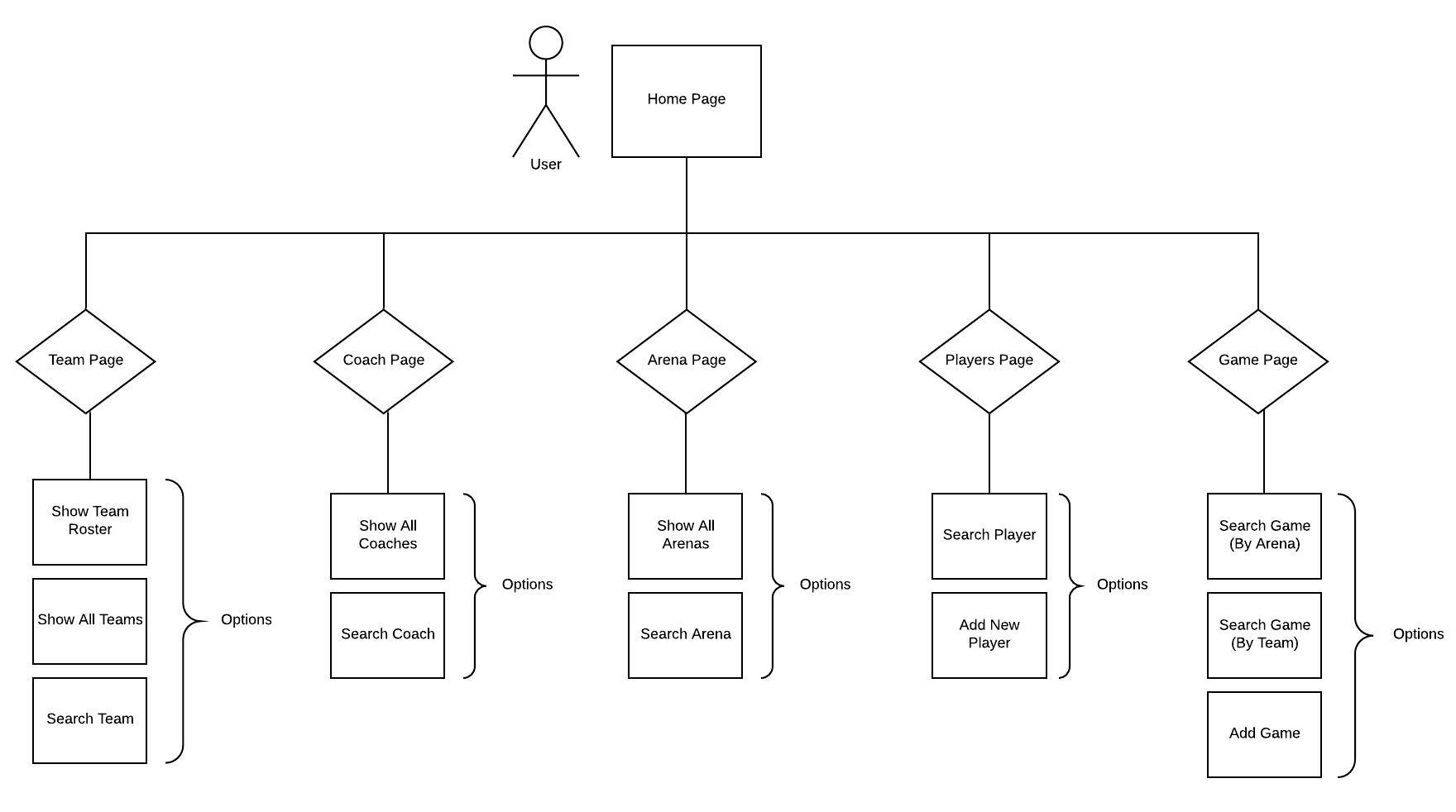
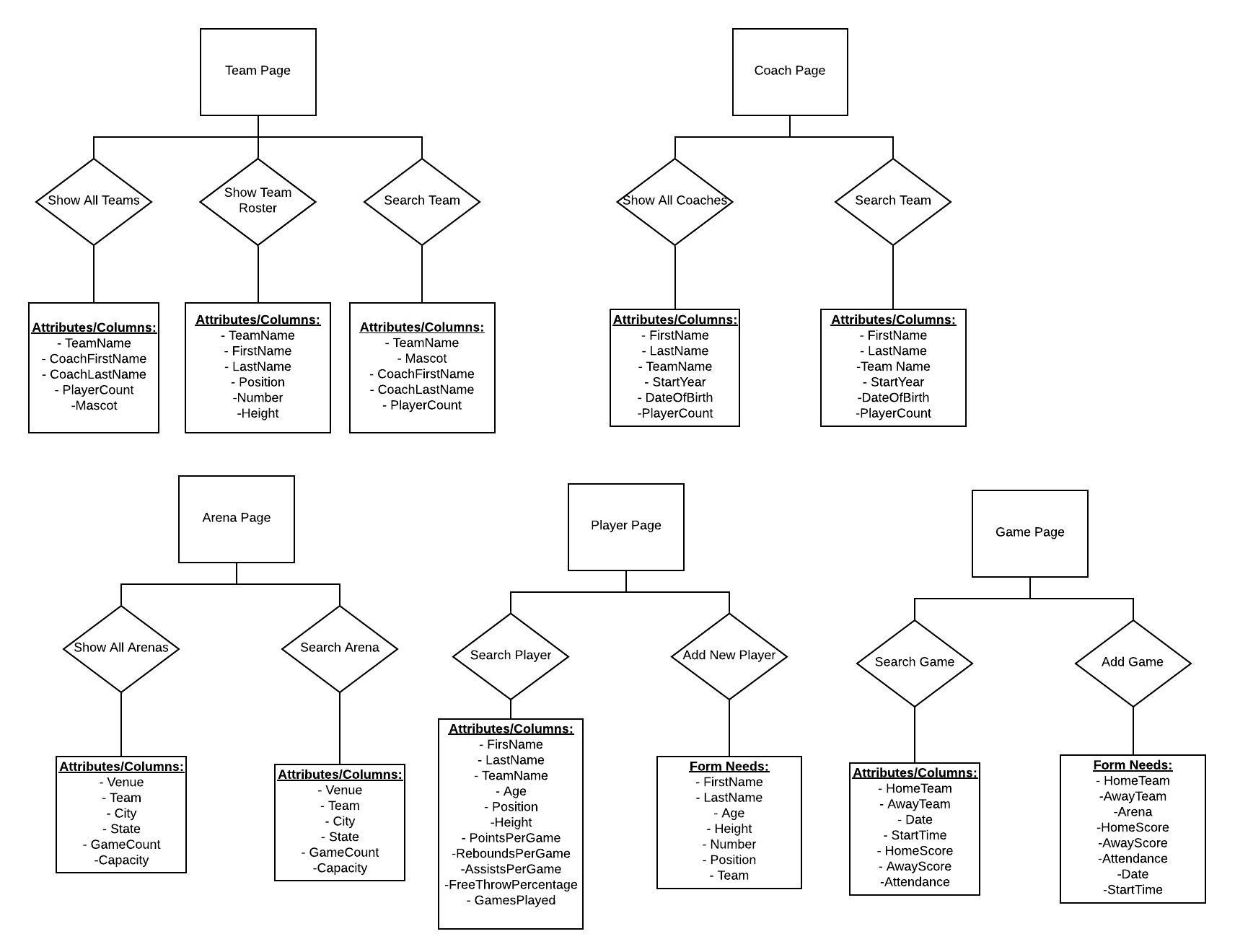
The system design was briefly mentioned in the introduction but we will expand more in this section. The system design is fairly simple and was decently easy to implement from the development perspective. Once again, we have built a web application (website) that consists of six pages. The first page is a homepage with some graphical elements for aesthetic purposes.It consists of five links in the upper banner - named teams, coaches, arenas, players, and games. The application always has an “NBA Management Application” label in the upper left which allows the user to go back to the homepage. The teams page has three different options for a user to interact with the database and extract information. The first thing the user can do is hit a button to display all the teams. This is great for comparing teams to one another. The user can also click a button to the right of a team and show that team’s roster. The last thing that the user can do is use the dropdown list box to select a team and search that team and get specific information on that team. Figure 1 shows the attributes (columns) that are located within these use cases. The next page option is for a user to click is the coach page. This page has two use cases for the user. A user can show all coaches or search a coach. Both of these are also good for comparisons or quick lookups and Figure 2 also shows the attributes for this display. The next page is the arena page and it also allows the user to either show all arena or to search a specific arena (also by dropdown listbox). The players page is next and allows a user to either search a specific player or add a new player. Searching a player is done by name and compares the string that is given in the textbox to the database. Adding a player will open up a new page with a form for the user to type the needed information (also shown in Figure 2). The last page is the game page. This page allows a user to search a game by an arena, by a team, or by both (both of which drop down list boxes for ease on both us and them). Lastly, the player can add a new game. When this option is selected, another new page will open and another form will be provided for the user to fill out. This, as well as the adding a player option, will add the player/game to the database and can later be used in lookups or comparisons. Figure 1 shows the different pages on the application as well as well as the use cases for each page. Figure 2 shows the attributes associated with each option a user can select which is equivalent to what is displayed to the user.****

Figure 1 - Shows the overall system design from a macro level.

Figure 2 - The “Attributes/Columns” show what is displayed to the user and the “Form Needs” are the different text boxes the user has to fill out to add a team/player

The back-end is the code that is written for each web page and handles what happens when an event is triggered on the front-end (such as clicking a button or writing in a textbox and hitting search). Therefore, this continues the trend of ease in our system design. Each web page will have methods that fire whenever their respective event takes place. These methods then call the stored procedures (which are obviously ran on our database) to return the information back to the method which is then ultimately displayed back on the web page for the user to see. This is very simple due the the fact that our procedures are used as objects in the C# language. A very simple sequence diagram is shown in Figure 3.

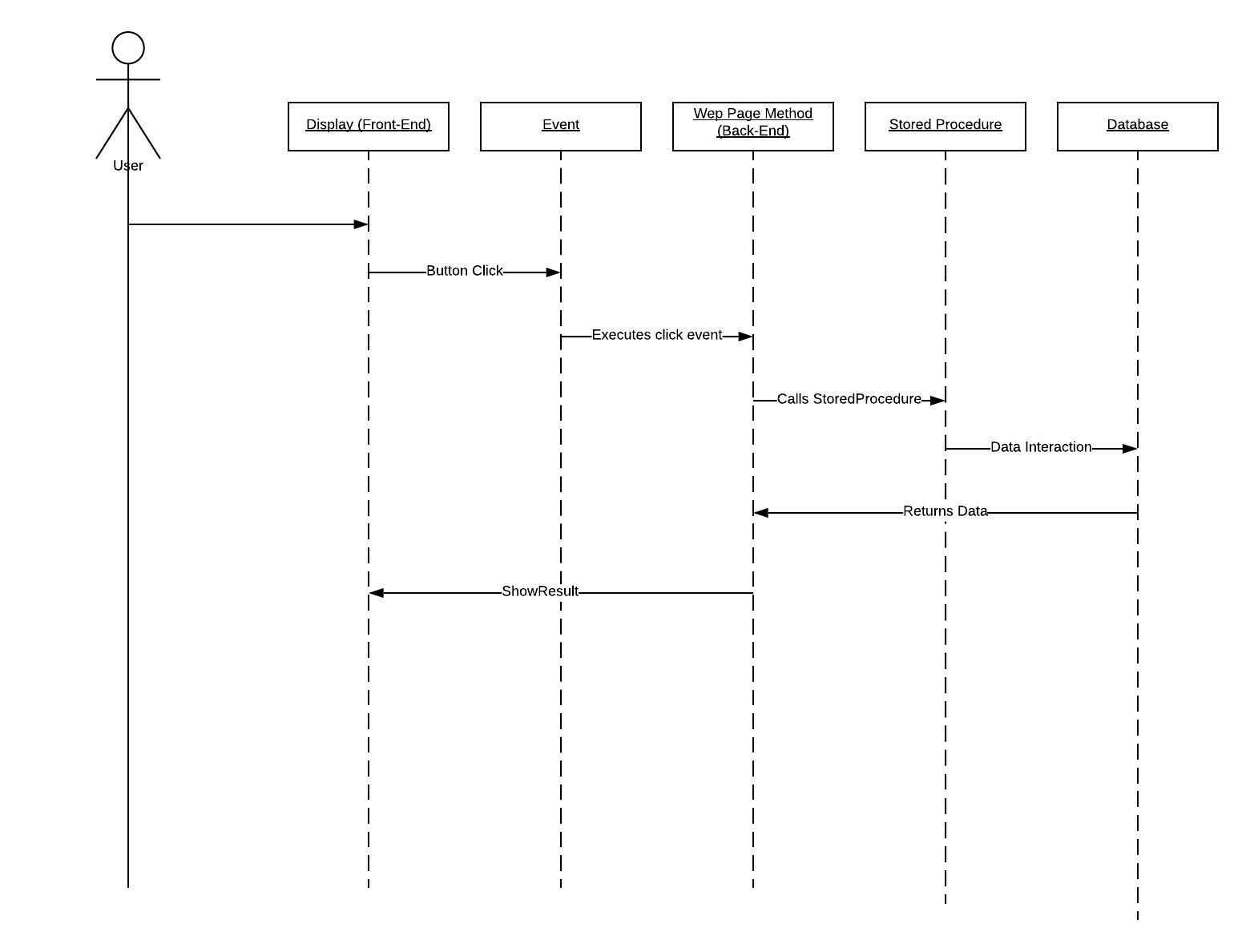
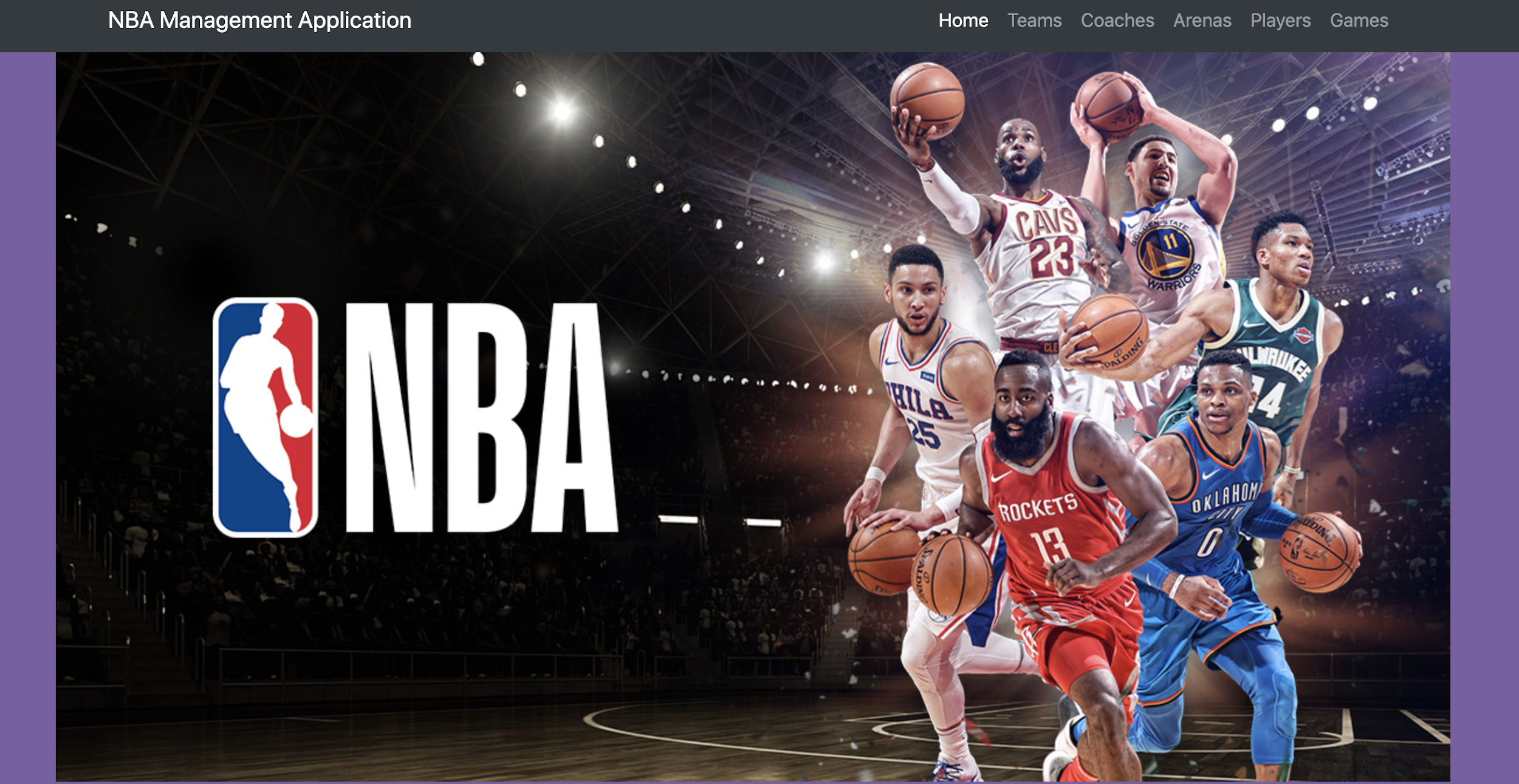
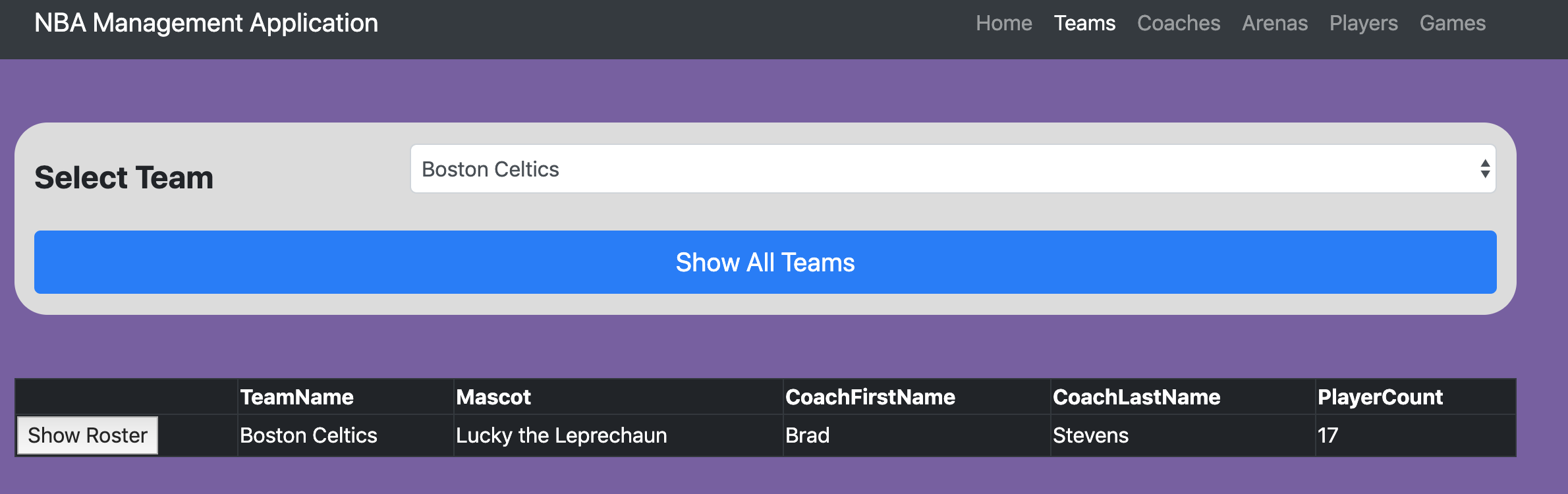


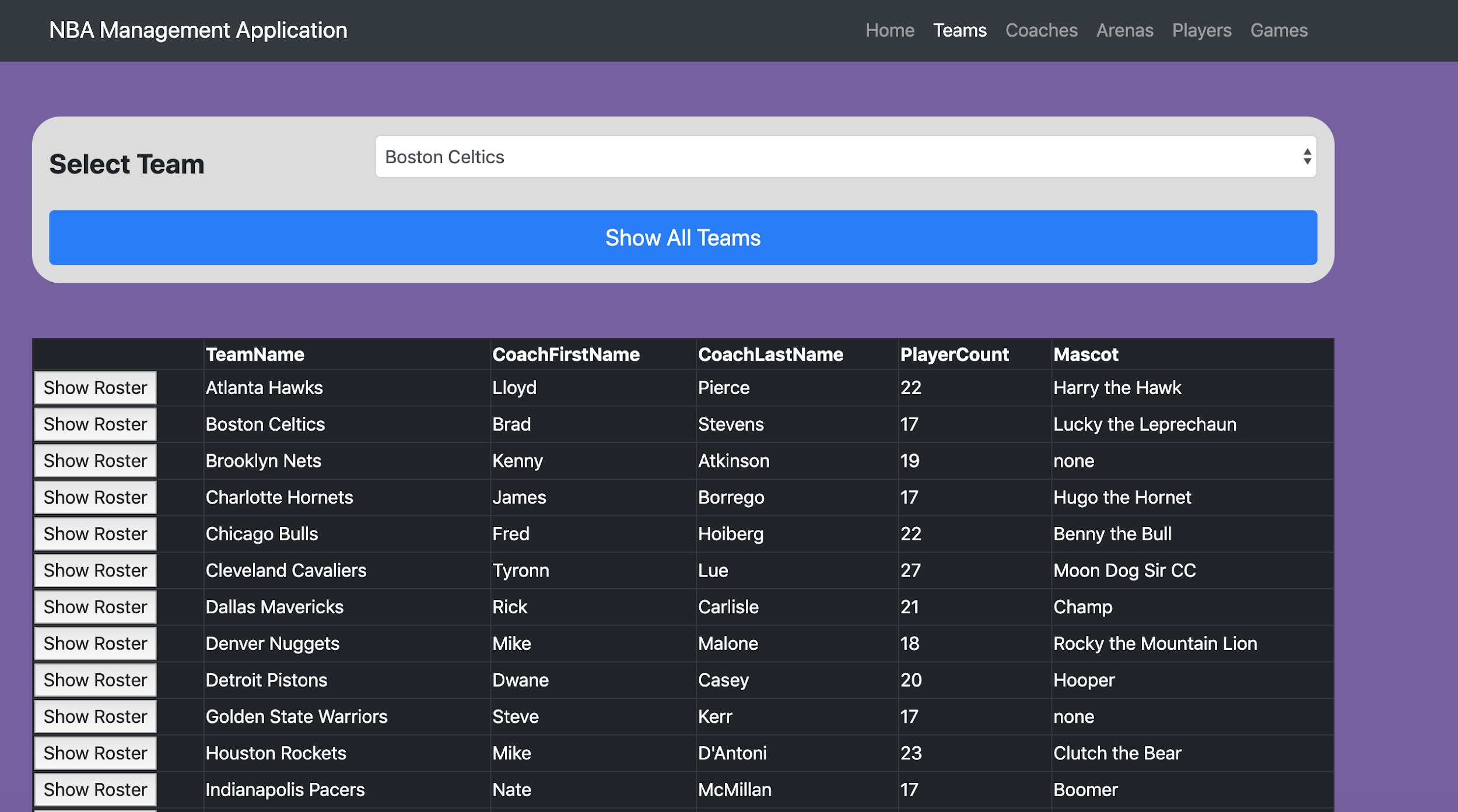
Figure 3 - A sequence diagram of how the system works as a whole.

**System Features and Usage**

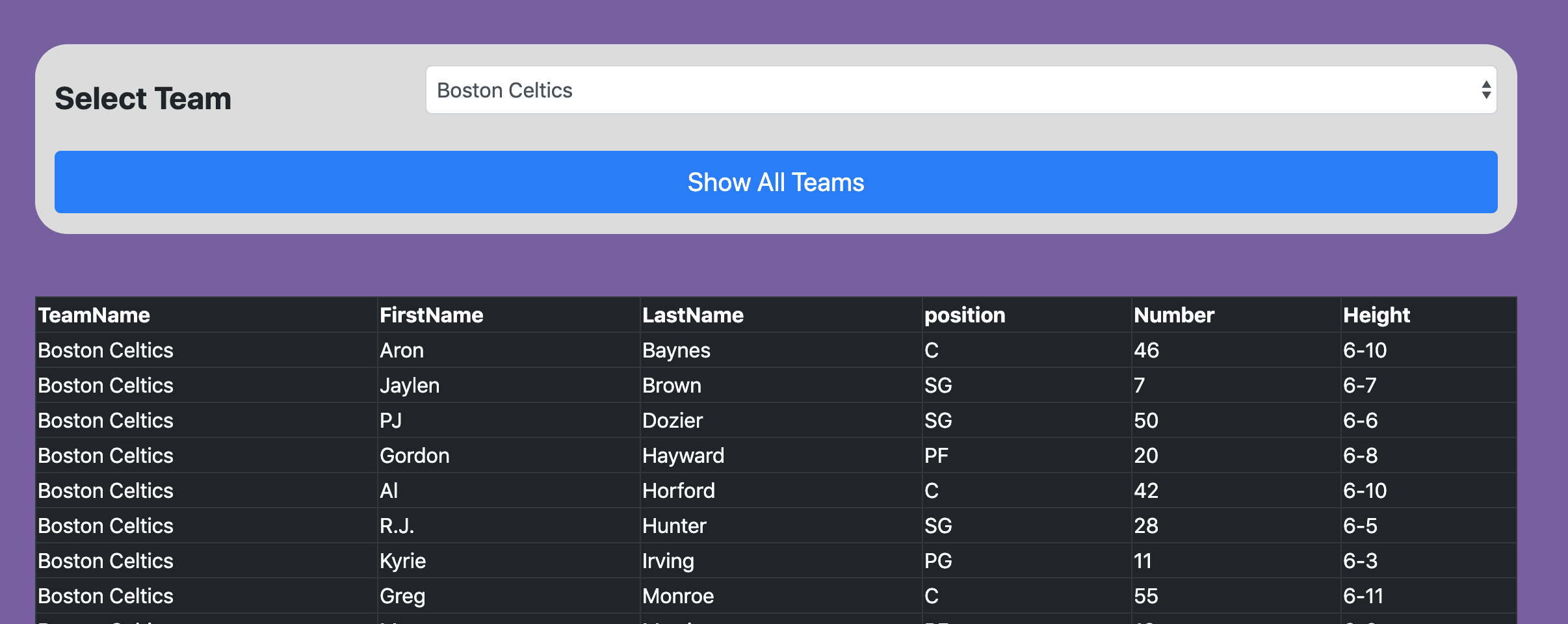
This is a screenshot of the homepage. As it can be seen, the five options are on the top right of the page with the links to new web pages.



Showing the searched team (Boston Celtics). This is displayed once the user clicks on the team from the drop down.



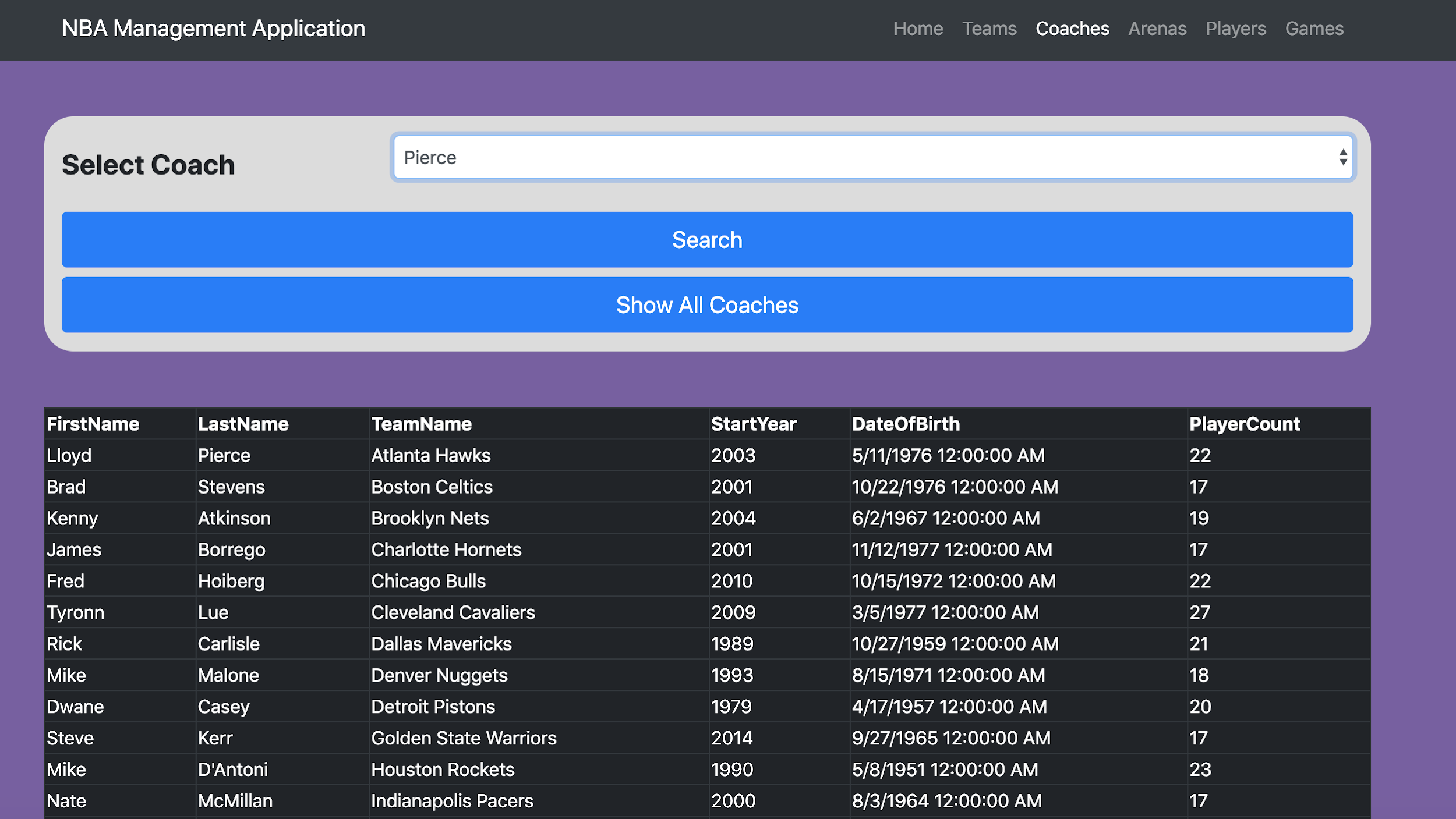
This screenshot depicts what happens when the user clicks on the “Show All Teams” button.



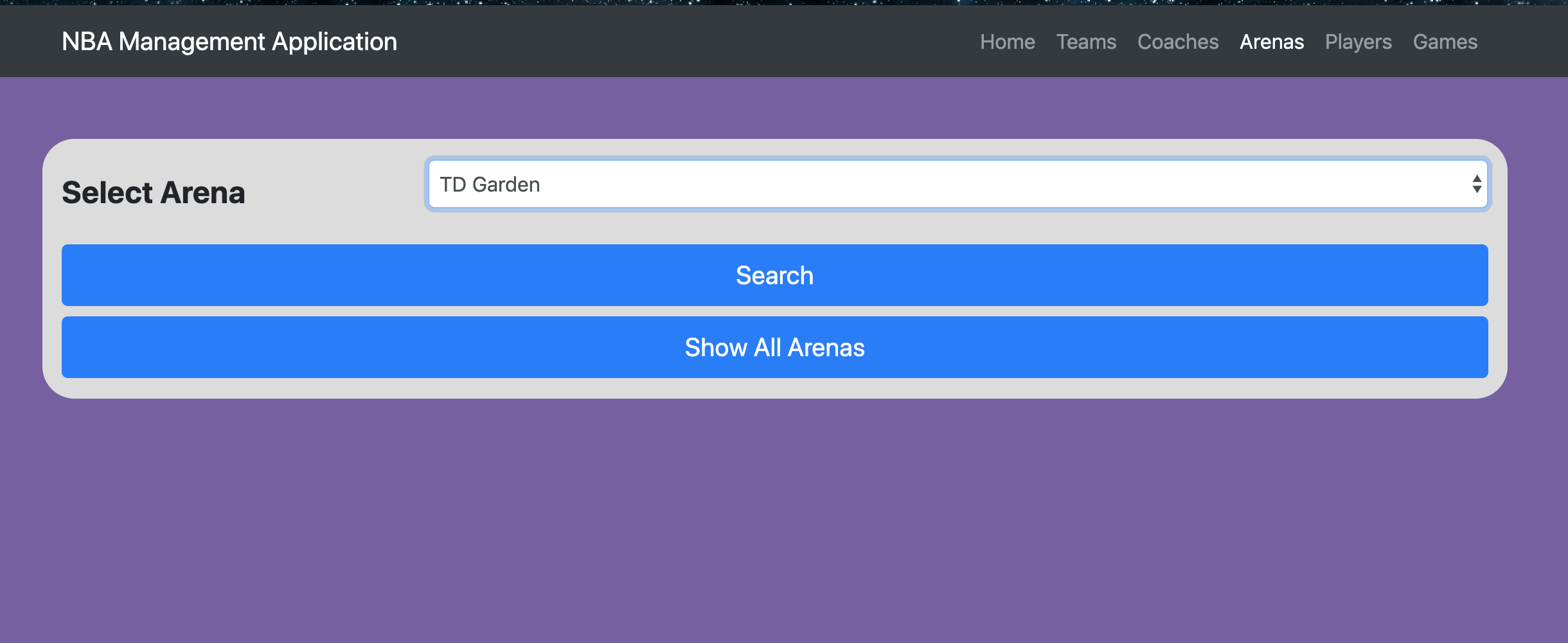
This shows what happens when the user clicks the “Show Roster” button of a selected team.



This is illustrating what happens when the user selects a name from the dropdown (such as Kerr) and then hits the “Search Button”.



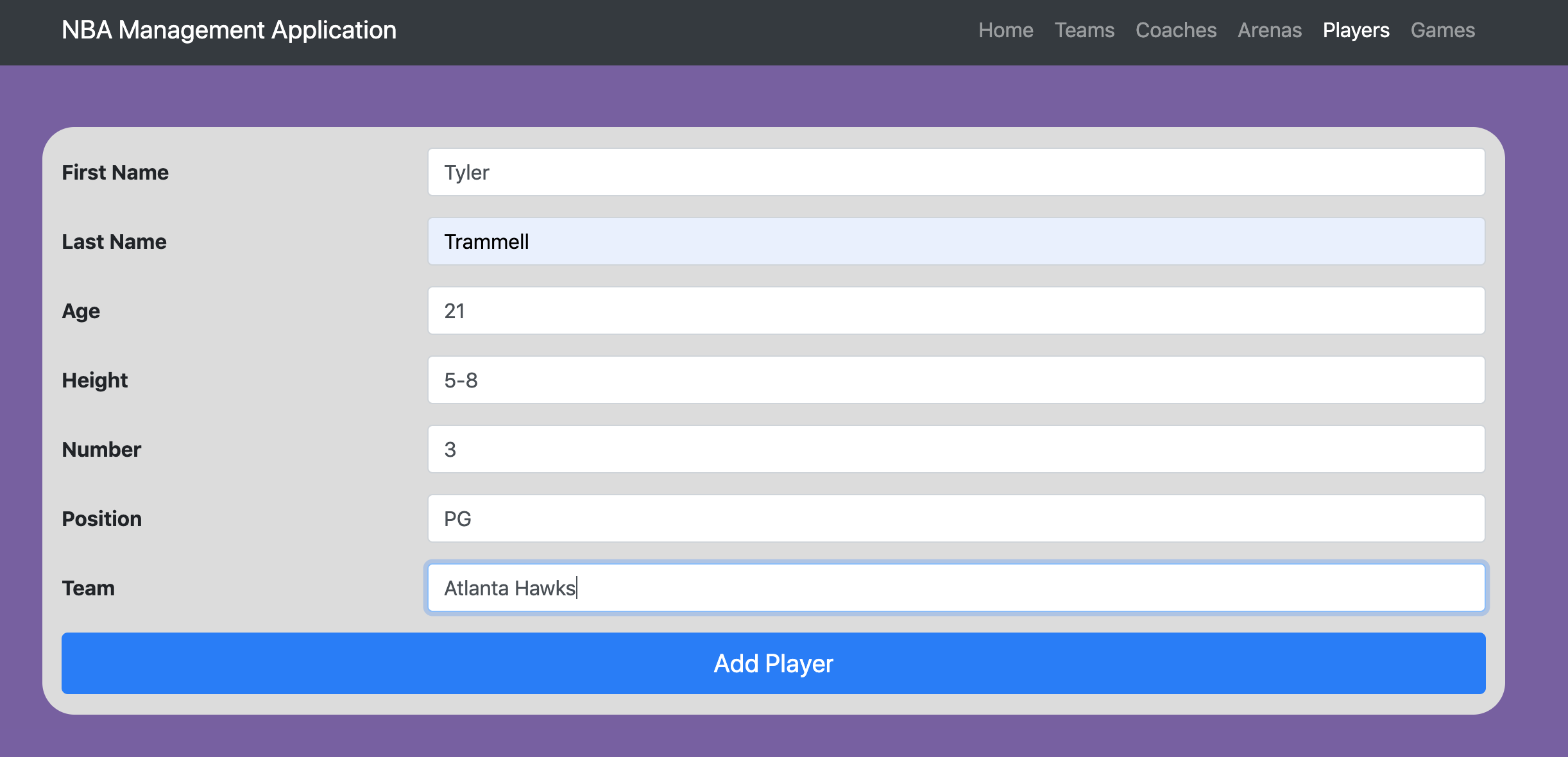
This screenshot is showing what happens when when the “Show All Coaches Button” is selected by the user.



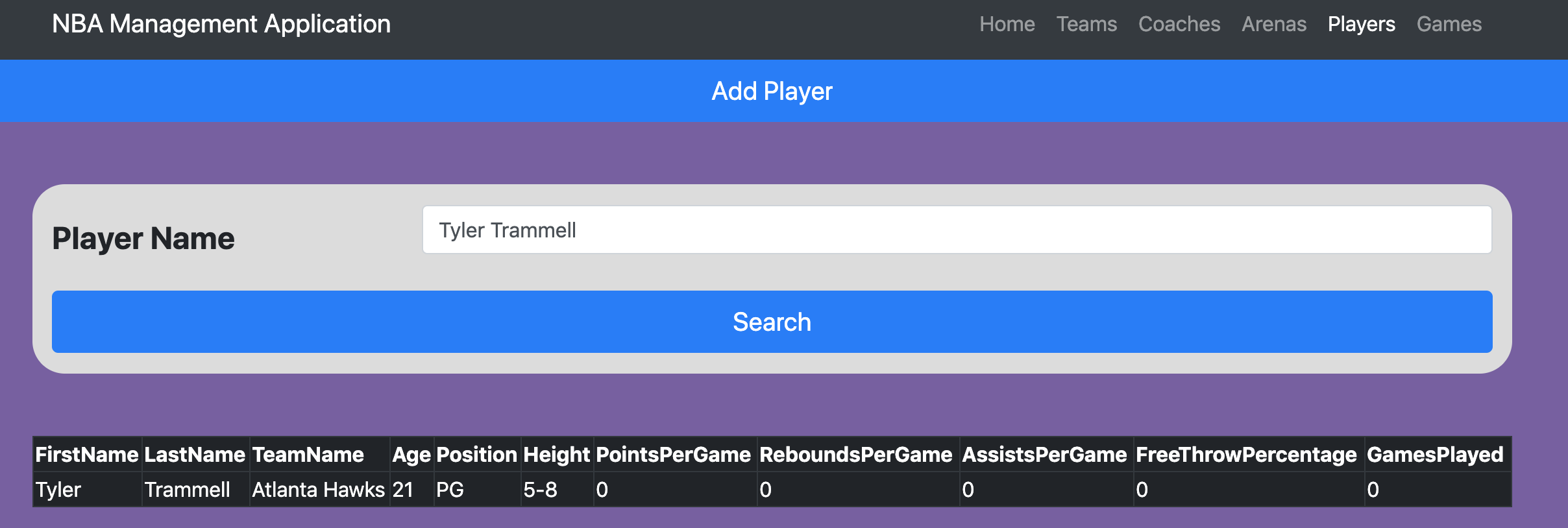
This is the “Arenas” page which is very similar - pretty much identical - to the coaches page. This is the only screenshot of this page we will show because it has the same functionality as the last page. The search will be based off of the dropdown box and the show all will just display all arenas.

Here is a look at the “Players Page.” As it can be seen, there are only two options - searching a player, or adding a player. The screenshot above is what happens if a player is to be searched (Lonzo Ball in our case). If the user types in an invalid name, nothing will be returned below.

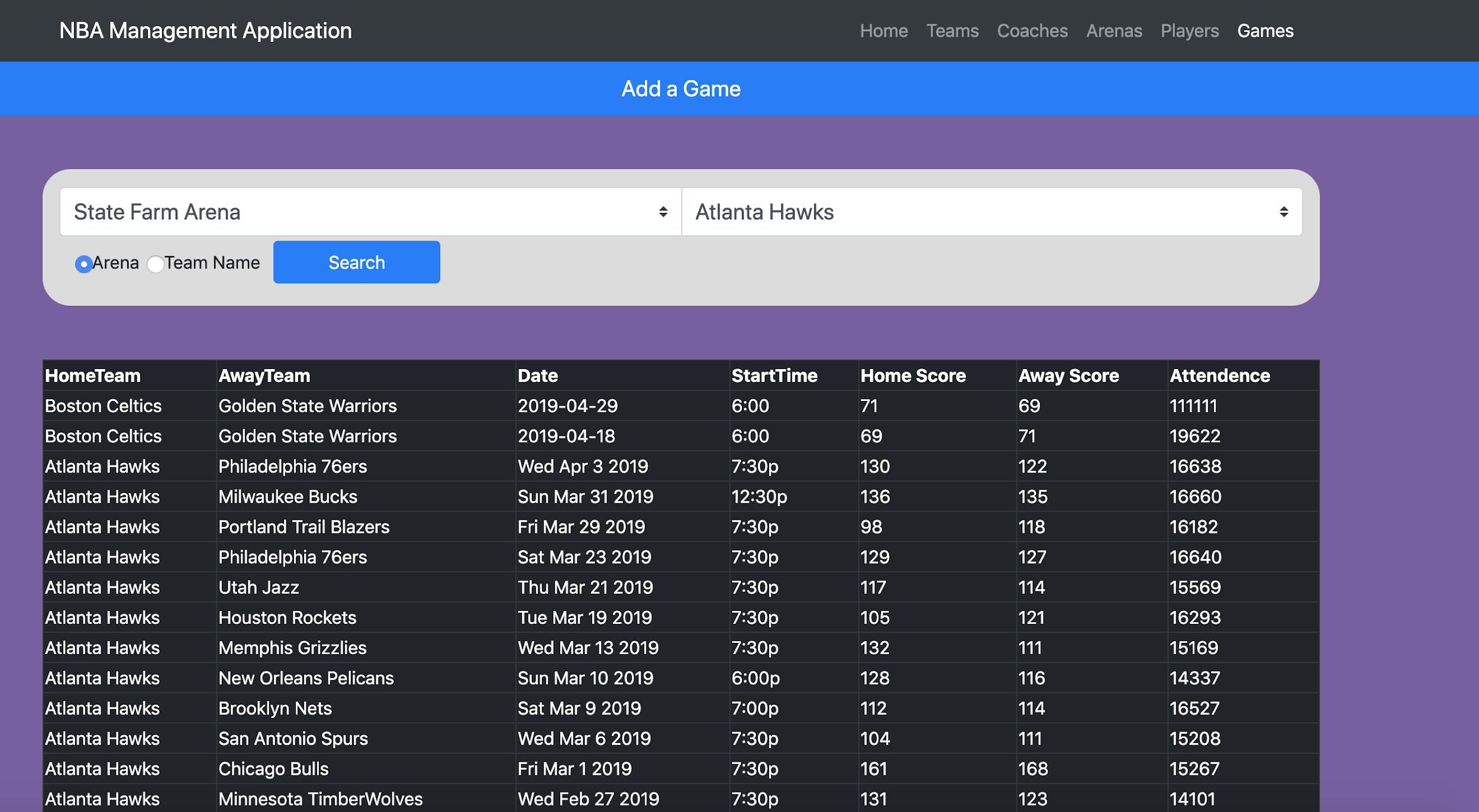
The stored procedure behind this feature of searching a player is mentioned in Report-Query 4. With that said, this is an example of the results that come from the stored procedure.



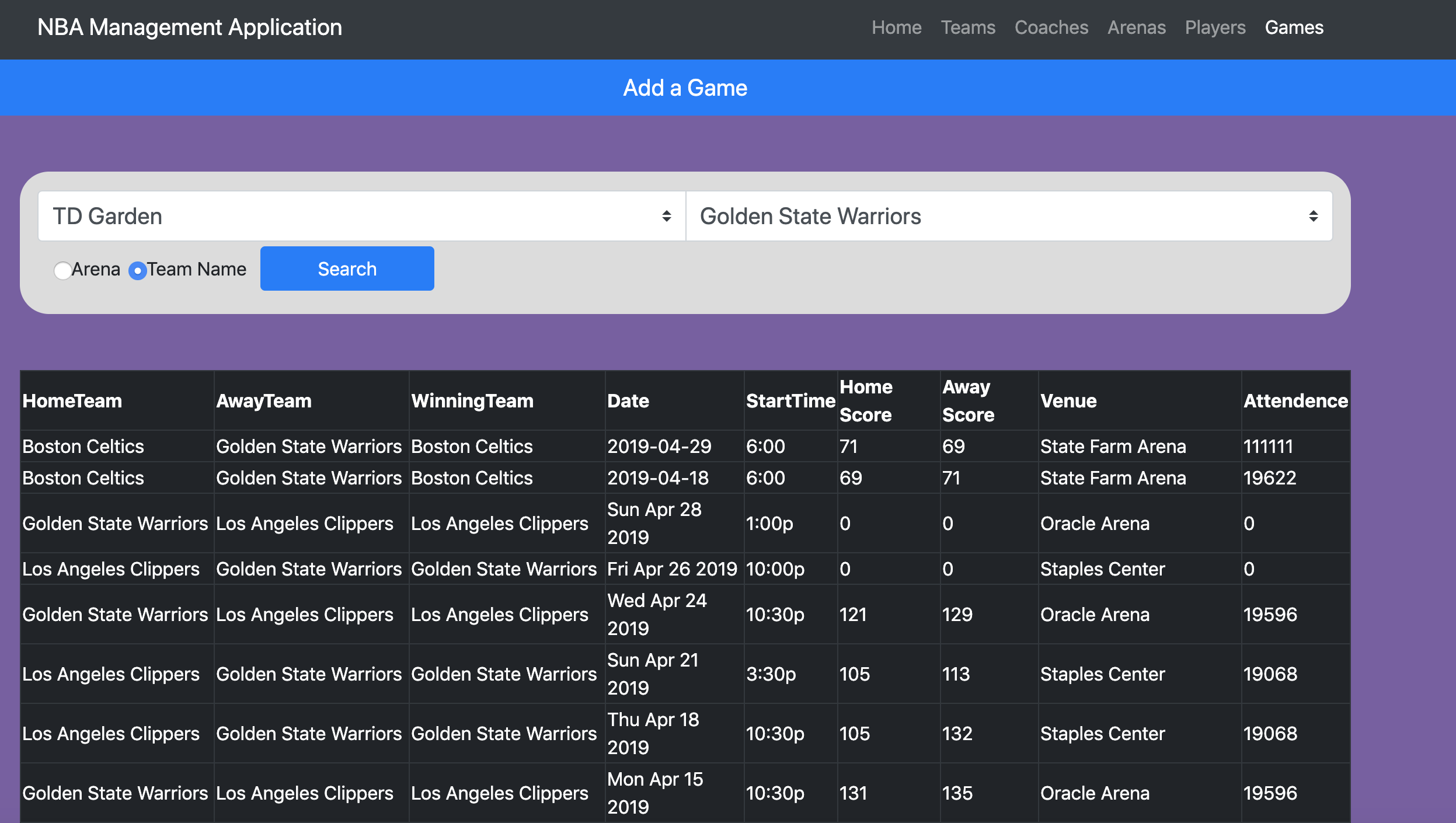
Here is a screenshot of the the “Add Player” form. As soon as the “Add Player” button is pressed, a small label will pop up on the bottom saying that it was added successfully.



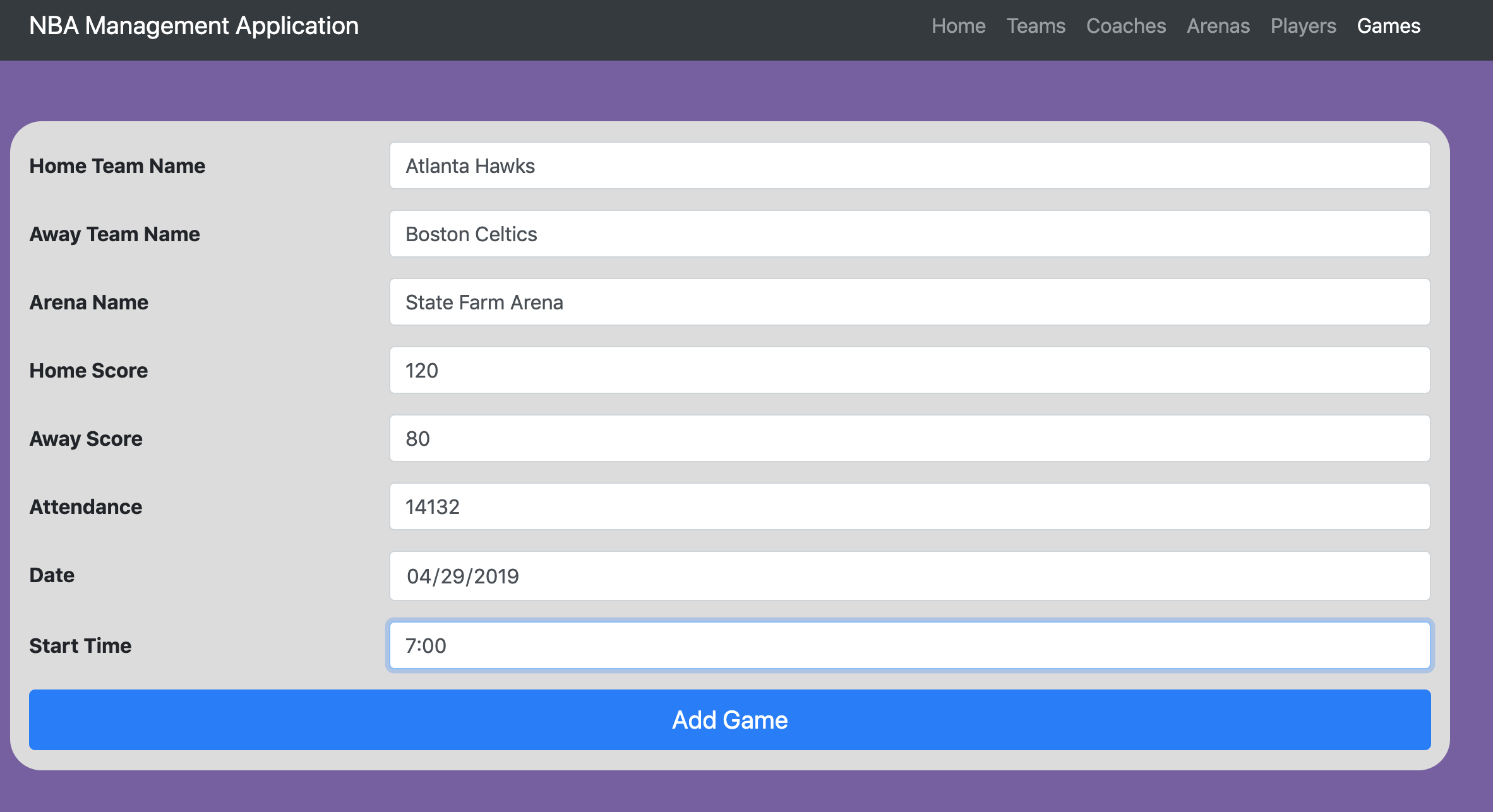
This image shows a search for the player who was just added into the database. It can be seen how the player was added into the database correctly because when searched, the information from the inputted information from the previous screenshot now appears. Thus showing the player has been added. This add player feature is Report-Query 3 in the report query section and this example shows the results that come from that stored procedure.



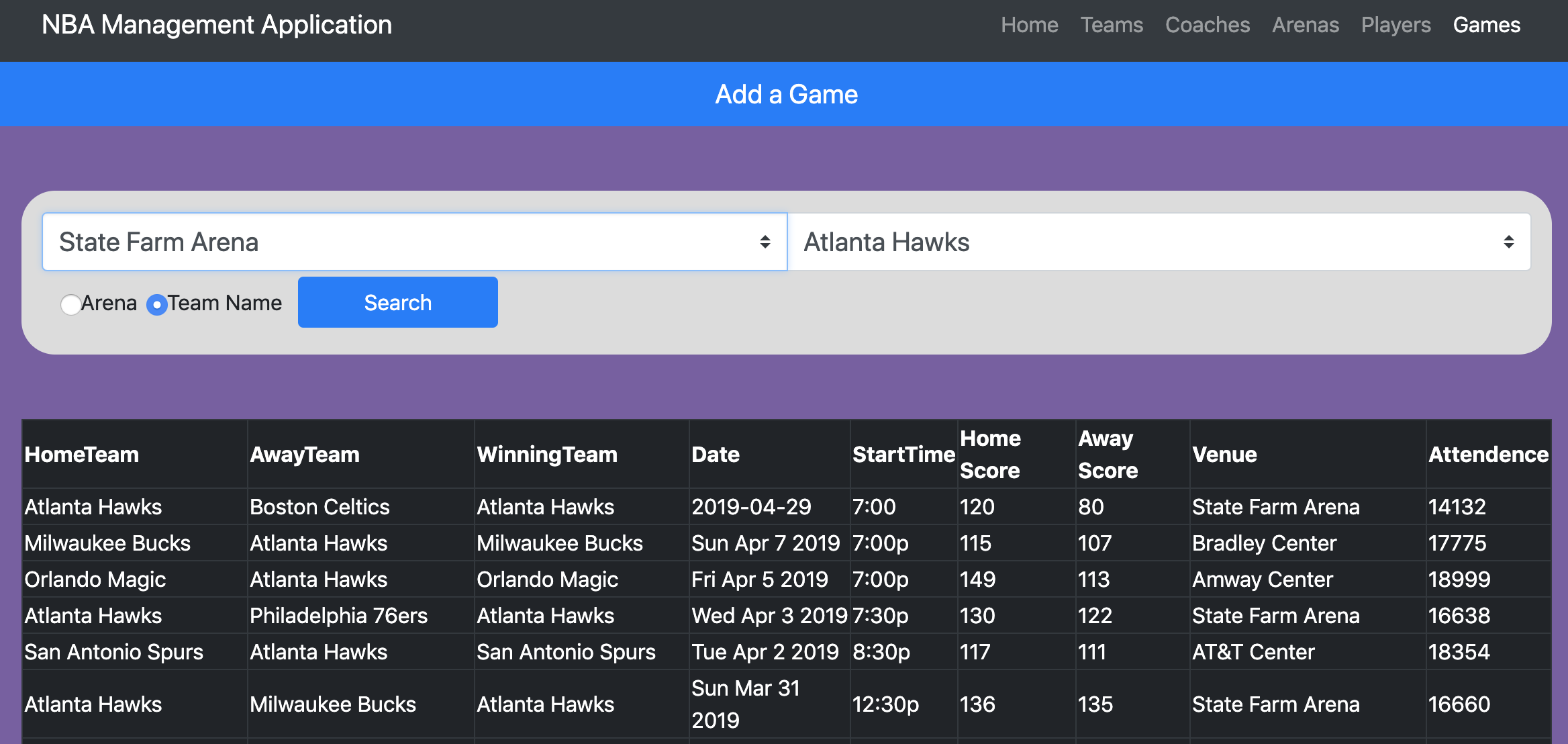
Here is the game page. There are three options for the user. The first is for the user to search by Arena (which is what is depicted in the screenshot above). All the games that are in State Farm Arena are displayed.



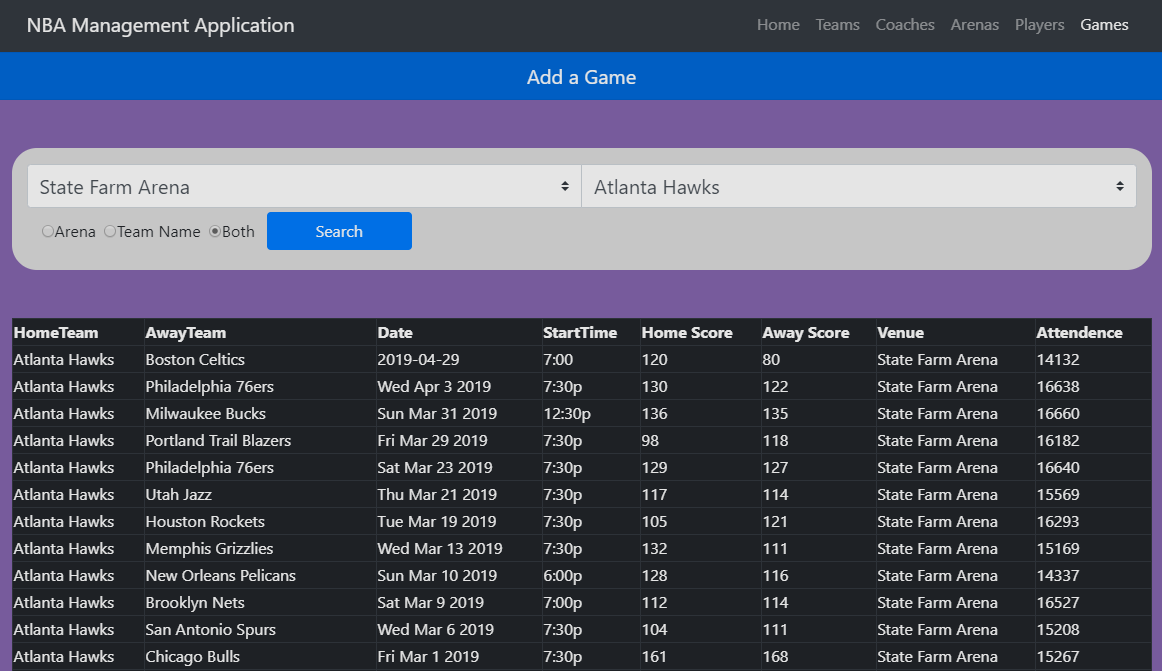
Here is an example of searching a game by a team. As it can be seen, the Golden State Warriors are always a team (whether home or away) because we are wanting to see all of their games.



Here is what it looks like when the “Add a Game” button is pressed. This form will need to be filled out by the user and once the “Add Game” button is pressed at the bottom, the database will be updated with this game.



This screenshot shows how the database has been updated to reflect the newly added game. The last screenshots information was added to the database and when searched, everything is reflected properly.

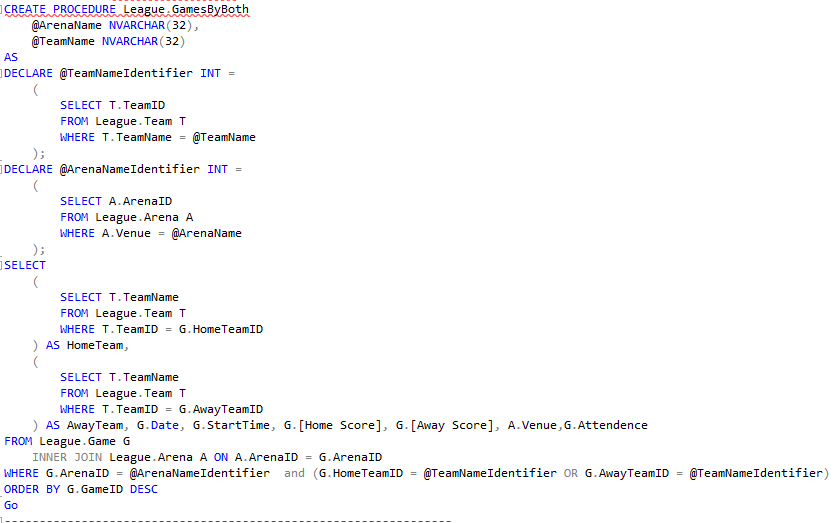


This screenshot shows the app in its latest state and it can be seen how there is now a button to search by both arena and team name. This is especially useful when wanting to see just home games (as the above screenshot illustrates) or away games, and for seeing head-to-head matchups that took place in the season. This feature of searching by both an arena and a team is a stored procedure below (Report-Query 2) and thus, the results that are seen in the screenshot above are the results that come from that stored procedure.

**Report Queries**

Report-Query 1

This query populates the player stats per game table which was mentioned in the “Database Creation” section mentioned above. When ran, this just says that the commands completed correctly with the certain rows affected. The SELECT statement will just return what was inserted. 

Report-Query 2

This procedure is implemented in our Games page of the web application and is to used to help aid in finding home and away game information for teams or just team games at certain arenas. When ran, this query/procedure takes in the team name and the arena name searched by the user and returns a table where both constraints are met with all the pertinent information on these games.

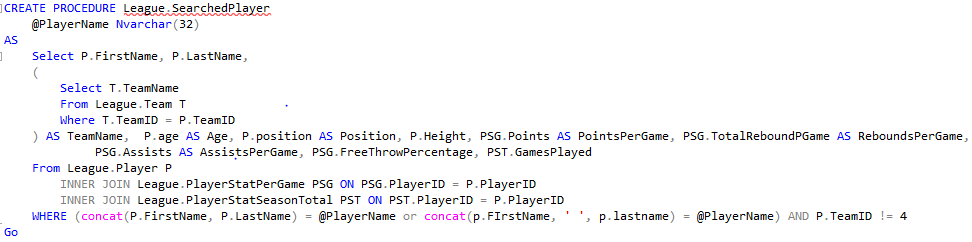
Report-Query 3



When ran, this query/procedure takes in all necessary player information needed to make a player and inserts it into the tables Player, PlayerStatsPerGame, and PlayerStatsPerSeasonTotal.

The results of this stored procedure is also demonstrated above in a screenshot in the section “System Features and Usage” (the example player that was added was named “Tyler Trammell”).

Report-Query 4



This stored procedure takes care of searching for a player. We use this procedure on the players page and use the user input (a string for a first and last name) and the concat function in the WHERE clause in order to search the database. This will return either the information on the searched player, or nothing if the name does not exist in our database. To avoid unnecessary duplication, the results of this procedure is already shown above in the “System Features” section of the report.

**Summary and Discussion**

The project ended up being very similar to what we had envisioned and drew up in our proposal. There were a few exceptions to this however on the database design and the data interactions. In our proposal, we had a “League.GamePlayer” table to hold data about a player’s in-game statistics. This table was scrapped out however and replaced by two tables, one named “League.PlayerStatsPerGame” and the other named “League.PlayerStatsPerSeason.” This change occurred because having the one table as we had originally planned for in our proposal was very difficult to implement given the data we pulled from online. Players who had changed teams were not accounted for and the one table approach could have lacked data integrity to a great extent. The implementation of the two tables not only solved this problem, but also gave us additional information about seasonal statistics as a whole. We now have data about a player’s statistics per game and statistics for the season as a whole, so we have knocked out two birds with one stone. Essentially, we have kept all attributes from our database design from the proposal and have just added many more attributes that could be used. When looking online for data on the NBA, we were able to find CSV files with much more information than we had planned for, and we figured it couldn’t hurt to add more data to the database. In conclusion, we changed up the database design by splitting the “League.GamePlayer” table into two, and added many more attributes (data) to the database.

The other change from our proposal can be found in the ways in which we allow data interaction in relation to the user. We had originally said our application would support the DELETE operation on the coach, player, and arena relations. However, after more thought and recollection from lectures, we decided that using the DELETE table operation could be very dangerous. A user could potentially delete something on accident (or on purpose/maliciously) and have the entire relation be messed up. As it has been mentioned in class, once something is deleted, there is no way of getting that data back, and we figured with an application such as the one we have built, that it was best not to allow the user to have the option of ridding any data at all. We also realized that having a DELETE table operation was pointless because the intended use of our application was for just the current season. We had originally thought we would allow the DELETE operation in the case of a player or coach retiring, or an arena being moved. However, thesen hypotheticals are *very* rare in general, and usually never happen midseason. In conclusion, we scrapped the DELETE table operation from our application due to the fact that allowing DELETEs could not only cause data to be lost, but also because it seemed pointless for the intended purpose of the application.

The other data interaction change we implemented in the final version of our project was the use of the UPDATE table operation. The UPDATE table operation was originally going to be supported in the coach, team, game, and arena relations. However, these were scrapped for similar reasons as the DELETE operation. For example, updating a coach is actually impossible by the way our database is designed. Unless if the coach somehow got a new name, there really is no data that can be updated or changed. A team *could* be updated, however, it is very rare for a team to change midseason (such as a coach or player getting fired/hired/traded) as usually these changes come in the offseason. Updating a player is also not efficient and/or useful for very similar reasons as the ones mentioned above for the team and coach relation. A player’s height, position, and number does not really change, as a trade (which *could* have a number change) does not usually happen midseason. Updating an arena is also very rare and we decided that implementing such a process did not seem useful for the application as we figured this relation is one that will be utilized the least by the user.

We learned a bunch throughout this project. For one, we learned how all the SQL that we have been learning throughout the semester is actually applied in industry. In other words, it was great to see how SQL is implemented in the real world. We also learned how well the MVC architecture works in relation to database applications. It was clear from the beginning that the major groupings of this project consisted of the front-end development (view), back-end development (controller), and the database itself (model). Because of this phenomenon, we split up the work of the project on these groupings. It was also nice to see how procedures work in application. Before the project, we really did not know the significance of procedures and why they were always talked about. However, it is now crystal clear as to why they are so important as they became the foundation for data interaction with the user.

One thing we might have done differently is work on the front-end side of the project concurrently with the database/procedures and back-end. This was difficult to actually accomplish because of the way we split up the work. However, once we were trying to piece the project together at the end, it was kind of difficult to implement the back-end to work with the user interface and the database. We eventually were able to get it all to work but it took some extra effort. One other thing we might have done differently is utilize different attributes of our database. With the “League.PlayerStatPerGame” and “League.PlayerStatSeasonTotal” tables, we had such an abundant amount of data and information that it was very difficult to decide what to display to the user. In other words, it was hard to weigh the most important attributes that the user might want to see. If we had more time we could have added more features and options for the user to utilize.

Adding on to this concept, a possible future improvement to our application would be adding more data interaction for the user. With more time, we could have added more buttons and tools on the front-end side to truly allow the user to see a large range of NBA analytics. For instance, there could be a button to show how well a player can score. We could pull from the PlayerStatPerGame and PlayerStatSeasonTotal tables for information on points per game, field goal percentage, free throw shooting percentage, three point shooting percentage, etc. This is just an example of one possible additional feature we could have added. Another feature we would like to add would be a login. Our application, as it stands right now, is far from being secure. A user, which could be anybody at this point in time, could easily add new games/players that are not real. If he/she wanted to be malicious, then they could add hundreds, if not thousands of random players or games to the system that do not exist and essentially bog down the application and make the app lose all data integrity. This is another reason we did not add the DELETE or UPDATE table operation. With more time, adding a login page and adding other security blankets to the application could really benefit it in many ways.

In general, our application did not differ too much from our original plan. Our database design was slightly different as we split a table into two seperate tables, both consisting of much more attributes then originally planned. We also changed how we wanted the user to be able to interact with the application and it’s data. We use SELECT statements to support displaying data and INSERT data to support adding new data. The DELETE and UPDATE table operations were not supported in our application (yet) because they did not seem necessary for our scope and we did not implement security mechanisms to ensure our data to be maintained safely. In the future, we could add other operations,security, and a login page to ensure only honest users can manipulate and interact with the data. We could also broaden the scope of the application to include information from previous seasons as well to truly allow the user to have a ton of information on the NBA.