**BASKETBALL PLAYER DATABASE**

Iain Cline, Ryan Welch, Dacey Wieland

Database System Concepts

John Keller

**INTRODUCTION**

The past several years the transfer portal in college sports has been running rampant and one example is men’s college basketball. A player’s history gets murky after years of being on multiple different teams. This system is designed to alleviate the pain of trying to find the current information of a player, by providing the status of men’s college basketball players in Division 1 basketball. If the user needs to find a player’s measurables, team, position, etc., this system will be able to detail each. This system will display the sports on startup, and users can click through the hierarchal structure to get to the conference, team, or player, use a filter list to help filter results, or use the search bar to search for the exact player/team they are wanting.

**TECHNICAL DESCRIPTION**

This system uses C#, SQL, and Visual Studio. The user interface is displayed in a Windows Form App. Our code also uses a sports data API for all the information related to College Basketball itself. The API is used to house the data needed until our code pulls it into our own database. The API that we are using has a NuGet package for use inside Visual Studio, the package is FantasyData.Api.Client, and in order to get the information that we need from the API int our database we also had to use the System.Data.SqlClient NuGet package.

The FantasyData.Api.Client allows us to connect to the API through an API key that is hard coded into the main Program so that you can create an API instance ranging from College Football to College Basketball and more. For the current iteration of our project, we are specifically using the Cbbv3StatsClient to pull all the information into our database. System.Data.SqlClient package allows our DataImporter class to connect to the SQL database that we set up for the project, also allows us to create commands to insert into tables as well as query tables.

**PHYSICAL DATABASE MODEL**

A diagram of a sports team

Description automatically generated

**SYSTEM DESIGN**

A screenshot of a computer

Description automatically generatedHere is how our program is structured in Visual Studio Code, visualized with a UML Diagram

A diagram of a computer

Description automatically generated  
A screenshot of a computer

Description automatically generated

For our UI we have this State Diagram to explain each state of the machine

A diagram of a basketball team

Description automatically generated

**SYSTEM FEATURES AND USAGE**

This is how our program looks on startup:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedFrom here, the user can click a sport, in this case, Basketball, and press the little right arrow. This brings us to this page.

Now there is a list of all the teams, the user can filter by conference they would like to select, or simply choose which team they would like to see. The user can also search for their team.

**A screenshot of a computer

Description automatically generated**

Searching up Kansas State brings us to the players and coach. The user can filter by position and can search up whichever player they would like.

**A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated**This brings us to the player information and update player form. The user can click the button and bring up the second form seen, to update the players information to the database. After clicking okay, the information will be updated accordingly.

**A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated**Here the user is looking for Kansas State, and then after clicking the search button, it takes them to the Kansas State page.

**AGGREGATING QUERIES**

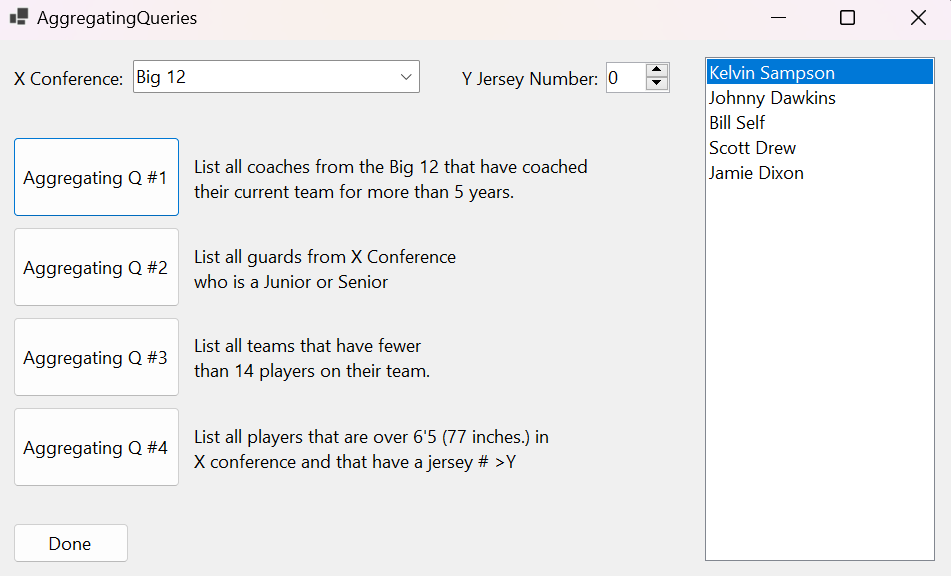
1. List all coaches from the Big 12 that have coached their current team for more than 5 years.

SELECT C.Name FROM [560\_proj\_Group09].Coach C

JOIN [560\_proj\_Group09].Team T ON T.TeamID = C.TeamID

JOIN [560\_proj\_Group09].Conference CF ON CF.ConferenceID = T.ConferenceID

WHERE CF.Name = N'Big 12' AND C.YearsCoached > 5;



1. List all guards from X Conference who is a Junior or Senior

SELECT P.[Name] FROM [560\_proj\_Group09].[Player] P

JOIN [560\_proj\_Group09].Team T ON P.TeamKey = T.TeamKey

JOIN [560\_proj\_Group09].Conference C ON C.ConferenceID = T.ConferenceID AND C.Name = N'Big 12'

WHERE P.YearInSchool = N'Junior' OR P.YearInSchool = N'Senior'

ORDER BY P.YearInSchool ASC;

A screenshot of a computer

Description automatically generated

1. List all teams that have fewer than 14 players on their roster

A screenshot of a computer

Description automatically generatedSELECT T.SchoolName FROM [560\_proj\_Group09].Team T   
 WHERE (   
 SELECT COUNT(P.PlayerID) FROM [560\_proj\_Group09].[Player] P  
 WHERE P.TeamKey = T.TeamKey   
 ) < 14

1. List all players that are over 6'5 in X conference (optional: that have a jersey >y)

SELECT P.[Name], P.Number FROM [560\_proj\_Group09].[Player] P

JOIN [560\_proj\_Group09].[Team] T ON T.TeamKey = P.TeamKey

JOIN [560\_proj\_Group09].Conference C ON C.ConferenceID = T.ConferenceID

AND C.Name = N'Big 12'

A screenshot of a sports program

Description automatically generatedWHERE P.Number > 15;

**SUMMARY AND DISCUSSION**

Compared to the original vision of this project, this project is way smaller in scope than intended. Our original intention was very naïve with how much sports and information we intended to gather. While this may have resulted in limiting the scope, it ended up making the information we gathered more complete and more functional. The changes in our design included removing all tables related to seasons and multiple sports. We originally intended to have rosters across multiple years for each team to provide more information on all the teams and players but given time constraints as well as the API we used not providing any roster data or any way to efficiently add seasons to our project we had to scrap the idea, but this could be added in the future fairly easily with more time and possibly a second API with the correct data. The removal of multiple sports only came to be after multiple attempts at getting a separate API specifically for college football to authorize. Even with the API key and the prefix, along with the NuGet package specifically made for the API, we could not get the API to connect to allow us to pull data. We planned to switch the API to one that we could get working, but by the time we had found the API we currently are using for College Basketball, we ran out of time to recode the entirety of the FootballDataImporter and had to scrap that for the time being. Implementing other sports such as Football would not be much of a challenge as we already have the base code that we needed along with all the tables that we would have to insert into set up, we just didn’t have the time to reformat everything in the short time we had left. These changes really stemmed from the API’s information differing from what we required in our design, on top of restricting access to what we could import, as well as time constraints. We realized quickly that we drastically underestimated the amount of work and time it would take to get our database running with all the information we needed and connected to the UI. Knowing what we know now I do not think we would have gone for such a large scope involving multiple sports as well as multiple seasons and dialed it back similar to what we have currently, and then in the future planned on adding more data and features as we went. We also learned that sometimes, even with all the necessary information, some NuGet packages and APIs just do not want to work for you and make you pivot to other solutions on the fly, and even if you do get it working sometimes merging the table information from an API to ours is less straightforward than previously thought. In the future, we would add more information related to basketball like coaches, and player’s various stats. We would also like to access football statistics, but due to time reasons, weren’t able reach a presentable state.