ENSF 608 – Final Project

NCAA Men's March Madness Database

Stephanie Walsh, Chelsea Johnson, Toya Okeke
14 April 2020

Abstract

This project was used to demonstrate our knowledge of how to build an effective database using SQL queries and a web framework. The database we built was of the NCAA basketball statistics needed to correctly predict a winning bracket in the annual March Madness tournament. Each year, the NCAA hosts a tournament in bracket style to crown the National Champion. Individuals compete with their peers or enter in live contests to see if they can predict the first ever perfect bracket. For the regular competitor, they need a simple but comprehensive breakdown of the previous tournaments stored in one place, so they can make better predictions. For data scientists and engineers, they need a database that can be preprocessed easily for input to their predictive models, and scalable for future use as new data arrives. Therefore, we built our website using a PHP framework as our front end. The backend of our system is an SQLite database that interacts with our front end and queries the relevant information the user wants to see.

Introduction

Our team has built a database containing statistics on teams and players from the 2015-2019 NCAA March Madness tournaments. This data was taken from the Kaggle website for the 2020 NCAA ML Competition.[1] The front end of our system is a web application that was developed using PHP. The main scenario for the end user will be to go through historical data for the past years. They can look at team and player statistics (stats) and see how they progressed through each tournament. The backend of our system runs on an SQLite server, which will query the results of the user selection. Our API will retrieve these results and display them for the user. Additionally, we will be able to login to the system as administrators and send insertion, deletion, and update queries to the database through our API. This will become more evident in the Functional Model section later.

Project Design

Background and Scope

Each year, the National Collegiate Athletic Association (NCAA) hosts a tournament in bracket style to crown the National Champion in Men's Division 1 basketball. There are 68 teams that participate in the tournament every season, so the odds of predicting the perfect bracket are 1 in 120.2 billion (if you know a little something about basketball). If you know nothing about basketball and base each game on 50/50 chance, the odds go down to 1 in 9.2 quintillion.[2] Yet, people continue to pay into a bracket pool with their peers and enter other competitions.[3] The person with the most correct guesses wins a large sum of money. To make things more interesting, Warren Buffet (CEO of Berkshire Hathaway) issued his own bracket challenge for his employees. Not only does he promise a $100,000 cash prize to the bracket that stays intact the longest, he goes as far as saying he will pay $1,000,000 every year for life to the employee that predicts the perfect bracket (i.e. they guessed every game outcome correctly).[4] Because of the cash value of these bracket competitions, participants are constantly looking for sources where they can gather information to build their brackets. For data scientists and engineers, it is especially important that there is an organized database that they can use build predictive models for generating outcomes.

Based on this, we decided to build a database containing stats on teams and players from the 2015-2019 tournaments. The front end of our system is a web application that was developed using PHP. The main scenario for the end user will be to go through historical data for the past years. They can look at team and player stats and see how they progressed through each tournament. The backend of our system runs on an SQLite server, which will query the results of the user selection. Our API will retrieve these results and display them for the user. Additionally, we will be able to login to the system as administrators and send insertion, deletion, and update queries to the database through our API.

Changes in Original Design

User Interface (UI)

The main scenario for end users and administrators in our original proposal have not changed. However, we have slightly changed the original design of our UI. At first, we were going to create our interface as a single web page, where the user would perform an action and a different view would appear on the same page (see Figure 1 below).

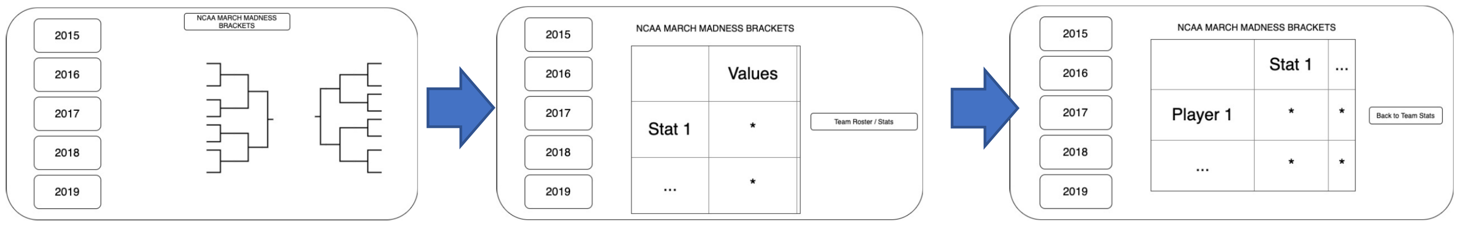


Figure 1 Old UI Implementation

Our final design is still similar, but each action the user performs directs them to a new page. Please refer to the User Manual section for more details.

Enhanced Entity Relationship Diagram (EERD)

During the first phase of designing our database, we determined the main entities we needed were:

- Player
- Team
- Game
- Tournament
- Conference
- Division
- Stats (both Player and Team Statistics)

At the time, we did not have access to the data that was on Kaggle. However, we were able to assume which attributes we needed and developed our first EERD based on those assumptions. Our first EERD is shown in Figure 2 below.

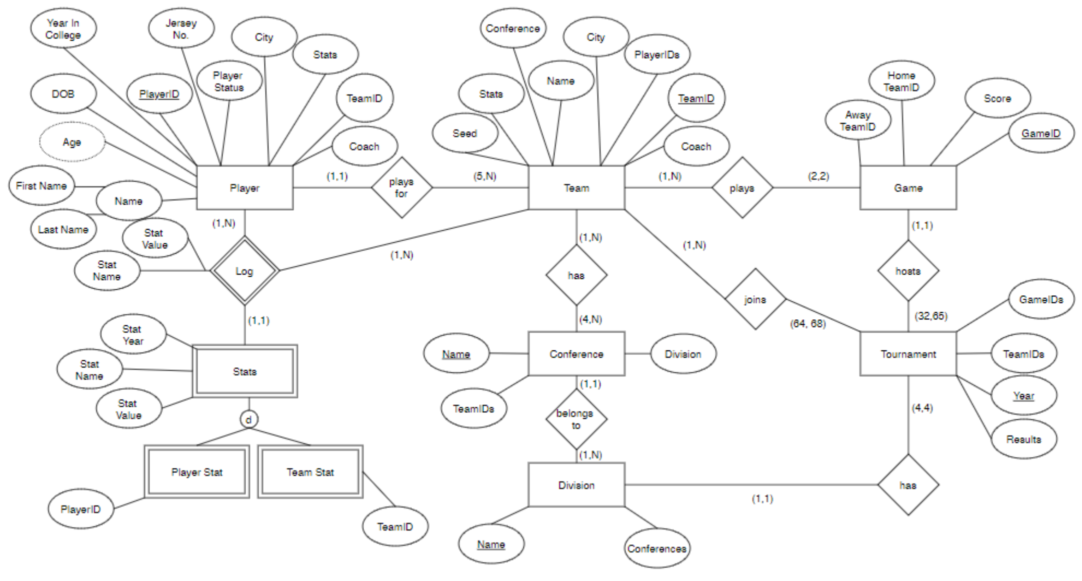


Figure 2 Old EERD

Once we had access to the data on Kaggle, we noticed that a lot of the attributes we wanted to build our database with were not included. Therefore, we had to simplify our design based on the data we had. This resulted in the EERD for our final design, as shown in Figure 3 below.

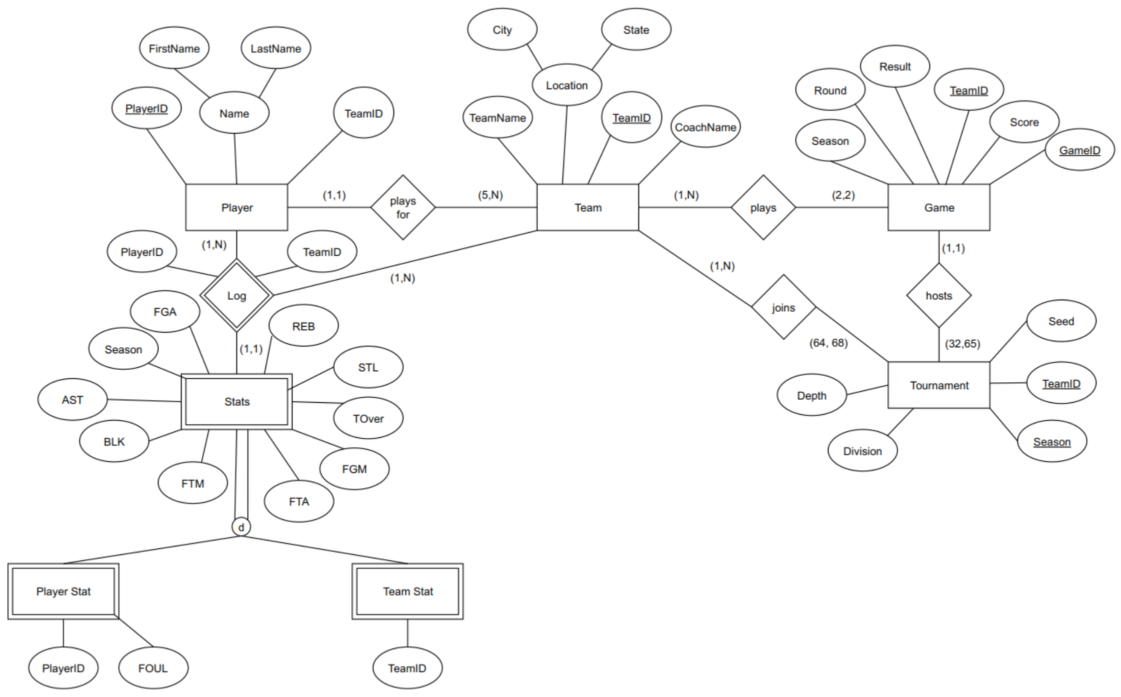


Figure 3 Final EERD

Relational Model

While developing the relational model, we noticed some of the attributes we originally designed for were missing. Most of the missing attributes were in the Player entity, which included:

- DOB
- Player Status
- Year in College
- Jersey No.
- City

We recognize that some of this data might be useful to people trying to predict how far a team will go in the NCAA tournament, however we do not believe that this data is as important as the individual player stats, so we opted to remove them from our database. The relational model we developed originally can be seen in Figure 4 below.

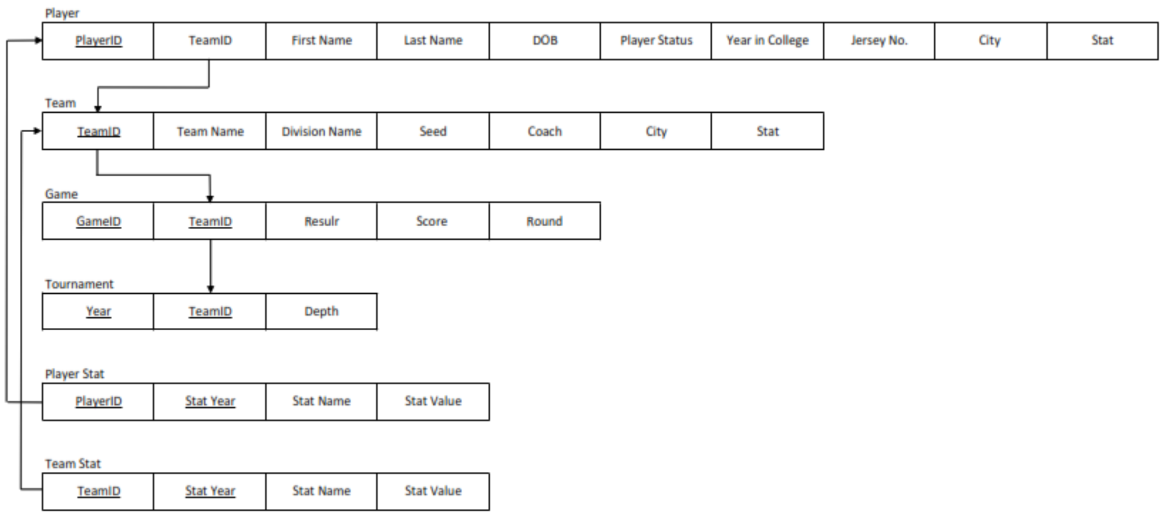


Figure 4 Old Relational Model

Some issues we noticed early on was that we had no way of connecting Game and Tournament back to the Teams relation schema. Once we started building the database in SQLite, we identified the foreign keys (FK) as either PlayerID or TeamID for all the relations. After removing the attributes that were not available and connecting the relations through the FKs, we created our final relational model design, shown in Figure 5 below.

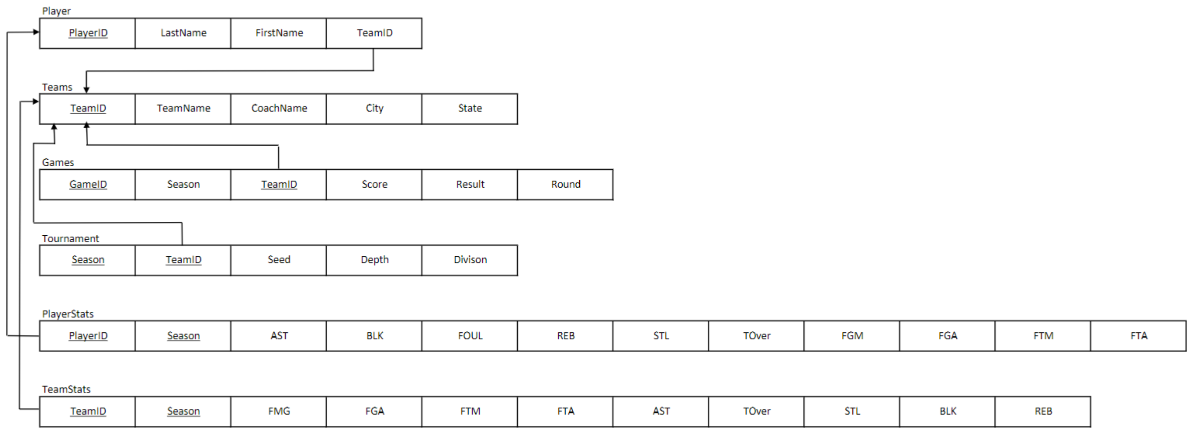


Figure 5 Final Relational Model

Functional Model

The functionality of our final design is still consistent with what we proposed. The user will start at the main page of our application. Here, they can choose which tournament year they would like to visit. Once they select a year, they will be taken to a page that shows the tournament bracket for that year. From there, they can click on a team in the bracket, which will direct them to a page containing the team's stats for that year, as well as the stats for each player in that year. Finally, if the user clicks on a player, they will be directed to page containing all the player's stats for each year that they played in the tournament (between 2015 and 2019).

On the main page, the user also can also log into our system as an administrator through our API if they are in fact an administrator. If they select this choice, they will be directed to an administrator login page. Here they will verify their identity by entering their username and password. If the user exists in the system, they will be taken to a page where they can manage the database. The functional model can be found in Figure 6 below.

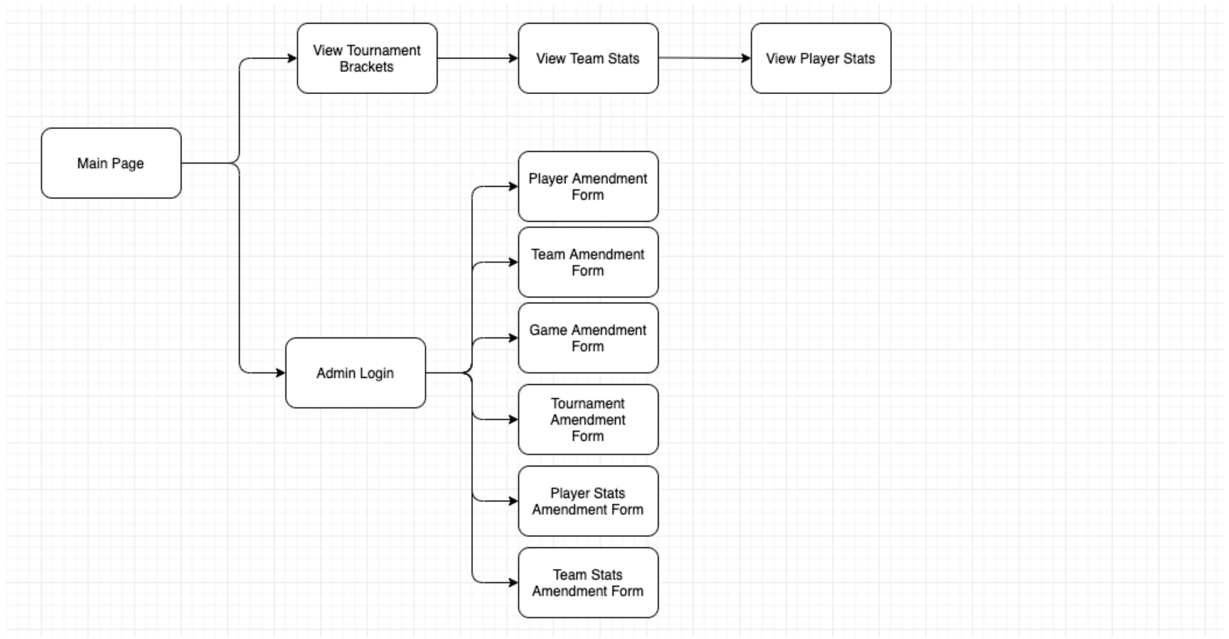


Figure 6 Final Functional Model

Implementation

Relational Model

Our final relational model in Figure 5 was able to meet our minimum requirement of 3rd Normal Form (3NF). Our candidate keys for each relation are the following:

- Player(<u>PlayerID</u>, LastName, FirstName, TeamID)
  CK = PlayerID
- Teams(<u>TeamID</u>, TeamName, CoachName, City, State)
  CK = TeamID
- Games(<u>GameID</u>, <u>TeamID</u>, Season, Score, Result, Round)
  CK = GameID, TeamID
- Tournament(<u>Season</u>, <u>TeamID</u>, Seed, Depth, Division)
  CK = Season, TeamID
- PlayerStats(<u>PlayerID</u>, <u>Season</u>, AST, BLK, FOUL, REB, STL, TOver, FGM, FGA, FTM, FTA)
  CK = PlayerID, Season
- TeamStats(<u>TeamID</u>, <u>Season</u>, FGM, FGA, FTM, FTA, AST, TOver, STL, BLK, REB)
  CK = TeamID, Season

Each of our non-key attributes are functionally dependent on **only** our CKs, meaning our design is in Boyce-Codd Normal Form (BCNF).

DBMS Design

Once we finalized the EERD and relational model, we started building our SQLite database. The first thing we did was manipulate the data in the feather files from Kaggle so that it was in the format we wanted. Using Python's built-in libraries, we manipulated the data and loaded it into our database. The structure of each relation schema are as follows:

Games CREATE TABLE "Games" ("GameID"INTEGER, "TeamID"INTEGER, "Season"INTEGER, "Score"INTEGER, "Result"TEXT, "Round"TEXT, PRIMARY KEY("GameID","TeamID"), FOREIGN KEY("TeamID") REFERENCES "Teams"("TeamID"))	**Tournament** CREATE TABLE "Tournament" ("Season"INTEGER NOT NULL, "TeamID"INTEGER NOT NULL, "Seed"TEXT, "Depth"TEXT, "Division"TEXT, PRIMARY KEY("Season","TeamID"), FOREIGN KEY ("TeamID") REFERENCES Teams("TeamID"))
Player CREATE TABLE "Player" ("PlayerID"INTEGER, "LastName"TEXT, "FirstName"TEXT, "TeamID"INTEGER, PRIMARY KEY("PlayerID"), FOREIGN KEY (TeamID) REFERENCES Teams(TeamID))	**Teams** CREATE TABLE "Teams" ("TeamID"INTEGER NOT NULL UNIQUE, "TeamName"TEXT, "CoachName"TEXT, "City"TEXT, "State"TEXT, PRIMARY KEY("TeamID"))
CREATE UNIQUE INDEX "player_index" ON "Player" ("PlayerID"ASC)	CREATE UNIQUE INDEX "teams_index" ON "Teams" ("TeamID"ASC)
Team Stats CREATE TABLE "TeamStats" ("TeamID"INTEGER, "Season"INTEGER, "FGM"INTEGER, "FGA"INTEGER, "FTM"INTEGER, "FTA"INTEGER, "AST"INTEGER, "TOver"INTEGER, "STL"INTEGER, "BLK"INTEGER, "REB"INTEGER, PRIMARY KEY("TeamID","Season"),	**Player Stats** CREATE TABLE "PlayerStats" ("PlayerID"INTEGER, "Season"INTEGER, "AST"INTEGER, "BLK"INTEGER, "FOUL"INTEGER, "REB"INTEGER, "STL"INTEGER, "TOver"INTEGER, "FGM"INTEGER, "FGA"INTEGER, "FTM"INTEGER, "FTA"INTEGER,

FOREIGN KEY ("TeamID") REFERENCES Teams(TeamID))	PRIMARY KEY("PlayerID","Season"), FOREIGN KEY("PlayerID") REFERENCES "Player"("PlayerID"))
Users (Admin) CREATE TABLE "Users" ("username"TEXT, "password"TEXT, PRIMARY KEY("username"))	

As you can see, the FK is either PlayerID or TeamID for each relation, which is in line with our relational model design. We also included indexes on the Player and Teams relations for faster access.

SQL Queries

INSERT Query

```
$stmt = $db->prepare("INSERT OR IGNORE INTO Player (PlayerID, LastName, FirstName,
TeamID) VALUES (:playerID, :lastName, :firstName,:teamID)");

$stmt->bindValue('playerID', $playerid, SQLITE3_INTEGER);
$stmt->bindValue('lastName', $lastname);
$stmt->bindValue('firstName', $firstname);
$stmt->bindValue('teamID', $teamid, SQLITE3_INTEGER);
$stmt->execute();
```

This query will insert a player if it is unique but will ignore it if it is already in the database.

UPDATE Query

```
$stmt = $db->prepare("UPDATE Player SET LastName = :lastName, FirstName = :firstName,
TeamID = :teamID WHERE PlayerID = :playerID");

$stmt->bindValue('playerID', $playerid, SQLITE3_INTEGER);
$stmt->bindValue('lastName', $lastname);
$stmt->bindValue('firstName', $firstname);
$stmt->bindValue('teamID', $teamid, SQLITE3_INTEGER);
$stmt->execute();
```

This query will update a player based on the primary key (PK) of PlayerID.

DELETE Query

```
$stmt = $db->prepare("DELETE FROM Player WHERE PlayerID = :playerID");

$stmt->bindValue('playerID', $playerid, SQLITE3_INTEGER);

$stmt->execute();
```

This query will delete a player based on the PK of PlayerID.

The queries for inserting and deleting stats and other relations are similar. The main difference is the number of attributes and PKs. To view the queries for the remaining relations, please refer to the ‘functions.php’ file in our source code.

SELECT / JOIN Queries

The SELECT and JOIN queries were designed for the user pages. The query below will pull the tournament and teams for their selected year using a combination of SELECT and JOIN. The result of this query is displayed in the bracket for that year using a while loop.

Tournament & Teams Query

```
$stmt = $db->prepare("SELECT * FROM Tournament INNER JOIN Teams ON Tournament.TeamID
= Teams.TeamID WHERE Seed = :seed AND Season = :season");

$stmt->bindValue('season', $season);
$stmt->bindValue('teamid', $teamid);

$results = $stmt->execute();
```

We have also made queries for collecting team stats and player stats based on the user's selection. Once this statement is executed, we use a while loop to display the results on the screen for the user.

Player Stats Query

```
$stmt = $db->prepare("SELECT DISTINCT Player.FirstName, Player.LastName,
PlayerStats.* FROM Player,  (
SELECT DISTINCT Teams.*, TeamStats.Season, TeamStats.FGM, TeamStats.FGA,
TeamStats.FTM, TeamStats.FTA,
TeamStats.AST, TeamStats.TOver, TeamStats.STL, TeamStats.BLK, TeamStats.REB FROM
Teams, (SELECT Tournament.Season, Tournament.Seed,
Tournament.Depth, Tournament.Division, Teams.* FROM Tournament INNER JOIN Teams ON
Tournament.TeamID = Teams.TeamID
WHERE Tournament.Season = :season ) AS TeamsX INNER JOIN TeamStats ON Teams.TeamID =
TeamStats.TeamID
WHERE TeamsX.Season = TeamStats.Season AND Teams.TeamID = :teamid) AS TeamStatsX LEFT
JOIN PlayerStats ON
PlayerStats.PlayerID = Player.PlayerID
WHERE Player.TeamID = TeamStatsX.TeamID");

$stmt->bindValue('season', $season);
$stmt->bindValue('teamid', $teamid);
$results = $stmt->execute();
```

Team Stats Query

```
$stmt = $db->prepare("SELECT DISTINCT Teams.*, TeamStats.Season, TeamStats.FGM,
TeamStats.FGA, TeamStats.FTM, TeamStats.FTA, TeamStats.AST, TeamStats.TOver,
TeamStats.STL,
TeamStats.BLK, TeamStats.REB FROM Teams, (SELECT Tournament.Season, Tournament.Seed,
Tournament.Depth, Tournament.Division, Teams.* FROM Tournament INNER JOIN Teams ON
Tournament.TeamID = Teams.TeamID WHERE Tournament.Season = :season) AS TeamsX INNER
JOIN TeamStats ON Teams.TeamID = TeamStats.TeamID WHERE TeamsX.Season =
TeamStats.Season AND Teams.TeamID = :teamid");

$stmt->bindValue('season', $season);
```

```
$stmt->bindValue('teamid', $teamid);

$results = $stmt->execute();
```

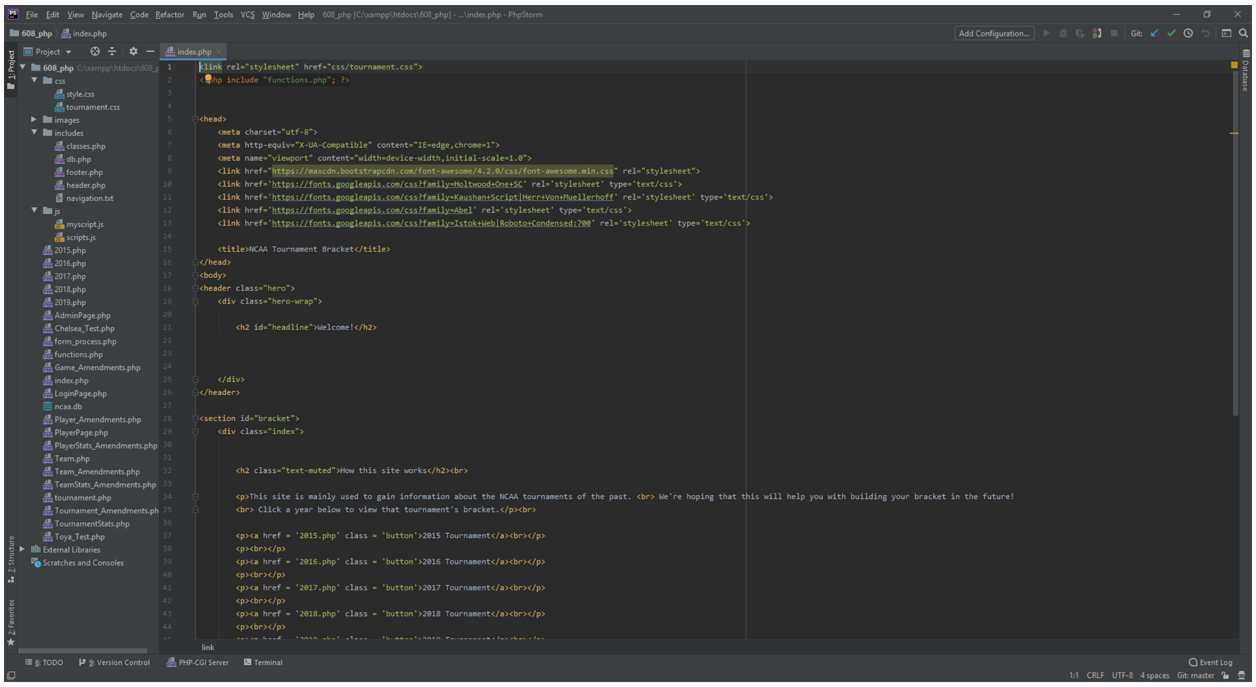
API Documentation

Connecting to the Server to View Pages

To access our API, we used a combination of JetBrains's PhpStorm IDE and XAMPP's Apache HTTP Server for Windows or MAMP for Macs. (You can use any PHP IDE or text editor that you choose, but the set up will be slightly different.)

All files need to exist in the htdocs folder in either the XAMPP or MAMP application folders to be accessed by the servers.

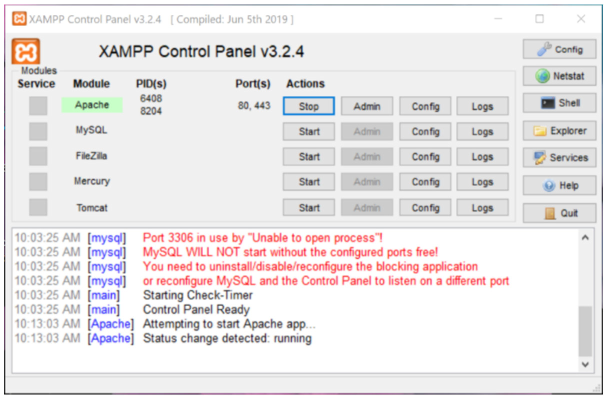
To begin, open PhpStorm and locate the file called 'index.php'.



```
<link rel="stylesheet" href="css/tournament.css">
<?php include "functions.php"; ?>


<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
    <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.2.0/css/font-awesome.min.css" rel="stylesheet">
    <link href='https://fonts.googleapis.com/css?family=Holtwood+One+SC' rel='stylesheet' type='text/css'>
    <link href='https://fonts.googleapis.com/css?family=Kaushan+Script|Herr+Von+Muellerhoff' rel='stylesheet' type='text/css'>
    <link href='https://fonts.googleapis.com/css?family=Abel' rel='stylesheet' type='text/css'>
    <link href='https://fonts.googleapis.com/css?family=Istok+Web|Roboto+Condensed:700' rel='stylesheet' type='text/css'>

    <title>NCAA Tournament Bracket</title>
</head>
<body>
<header class="hero">
    <div class="hero-wrap">

        <h2 id="headline">Welcome!</h2>



    </div>
</header>

<section id="bracket">
    <div class="index">


        <h2 class="text-muted">How this site works</h2><br>

        <p>This site is mainly used to gain information about the NCAA tournaments of the past. <br> We're hoping that this will help you with building your bracket in the future!
        <br> Click a year below to view that tournament's bracket.</p><br>

        <p><a href = '2015.php' class = 'button'>2015 Tournament</a><br></p>
        <p><br></p>
        <p><a href = '2016.php' class = 'button'>2016 Tournament</a><br></p>
        <p><br></p>
        <p><a href = '2017.php' class = 'button'>2017 Tournament</a><br></p>
        <p><br></p>
        <p><a href = '2018.php' class = 'button'>2018 Tournament</a><br></p>
        <p><br></p>
```

Figure 7 – Screenshot of PhpStorm

If using Windows, once the file has been located, open XAMPP and start the Apache HTTP Server.


XAMPP Control Panel v3.2.4 [Compiled: Jun 5th 2019]
XAMPP Control Panel v3.2.4
Config
Netstat
Shell
Explorer
Services
Help
Quit

Modules Service	Module	PID(s)	Port(s)	Actions			
	Apache	6408 8204	80, 443	Stop	Admin	Config	Logs
	MySQL			Start	Admin	Config	Logs
	FileZilla			Start	Admin	Config	Logs
	Mercury			Start	Admin	Config	Logs
	Tomcat			Start	Admin	Config	Logs

```
10:03:25 AM  [mysql]   Port 3306 in use by "Unable to open process"!
10:03:25 AM  [mysql]   MySQL WILL NOT start without the configured ports free!
10:03:25 AM  [mysql]   You need to uninstall/disable/reconfigure the blocking application
10:03:25 AM  [mysql]   or reconfigure MySQL and the Control Panel to listen on a different port
10:03:25 AM  [main]    Starting Check-Timer
10:03:25 AM  [main]    Control Panel Ready
10:13:03 AM  [Apache]  Attempting to start Apache app...
10:13:03 AM  [Apache]  Status change detected: running
```

Figure 8 - Screenshot of XAMPP Control Panel

If using a Mac, open MAMP and start the Servers.

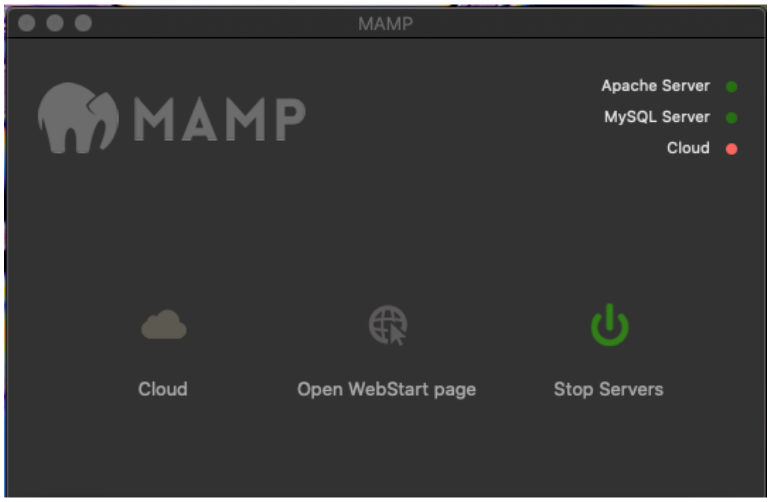
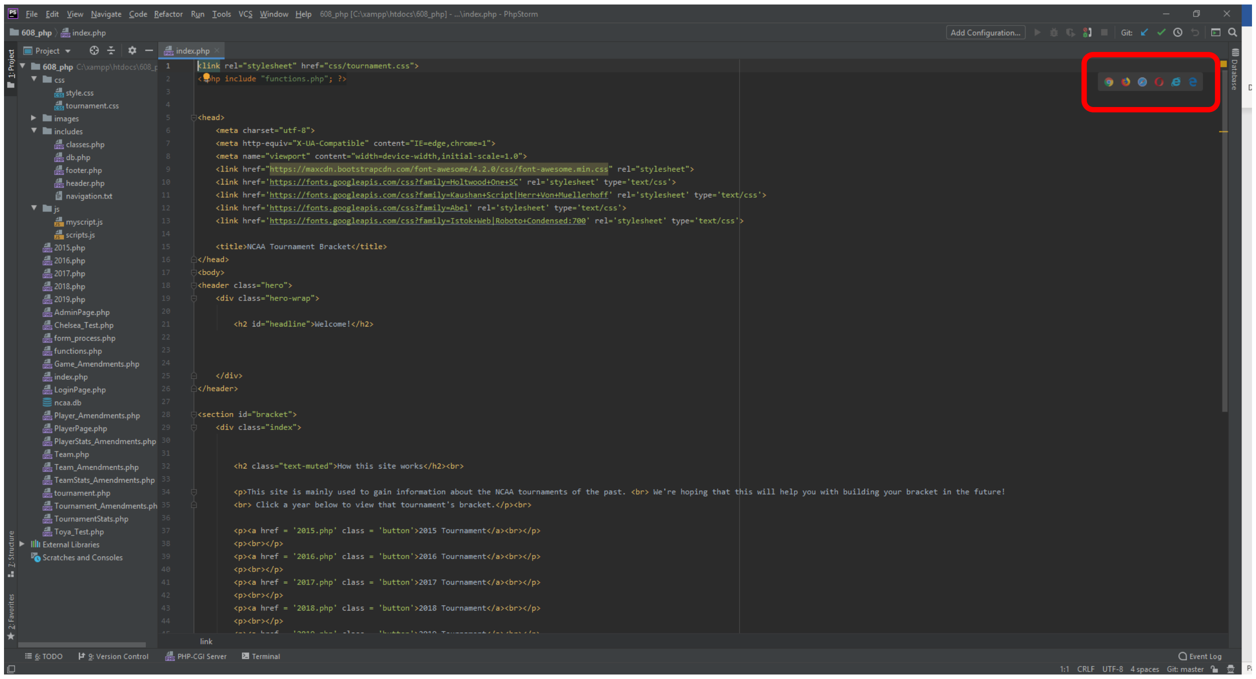


Figure 9 - Screenshot of MAMP Control Panel

With the Apache Server started, you can now launch the API. To do this return to PhpStorm and hover in the opened 'index.php' file until you see the internet browser options appear in the top right-hand corner of the page.


File Edit View Navigate Code Refactor Run Tools VCS Window Help 608_php [C:\xampp\htdocs\608_php] - ...\index.php - PhpStorm
608_php
index.php
Add Configuration...
Project
index.php
link
6: TODO
9: Version Control
PHP-CGI Server
Terminal
Event Log
1:1 CRLF UTF-8 4 spaces Git: master

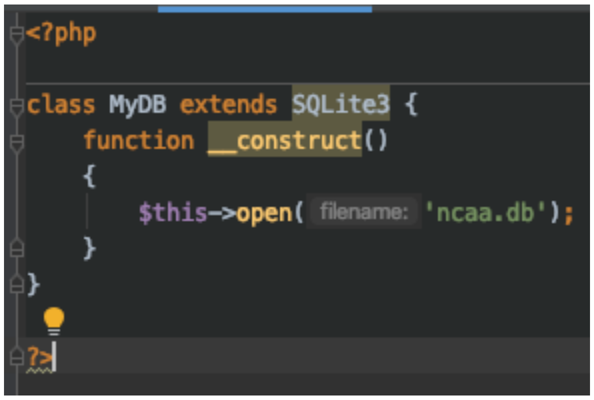
```
<link rel="stylesheet" href="css/tournament.css">
<?php include "functions.php"; ?>


<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
    <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.2.0/css/font-awesome.min.css" rel="stylesheet">
    <link href='https://fonts.googleapis.com/css?family=Holtwood+One+SC' rel='stylesheet' type='text/css'>
    <link href='https://fonts.googleapis.com/css?family=Kaushan+Script|Herr+Von+Muellerhoff' rel='stylesheet' type='text/css'>
    <link href='https://fonts.googleapis.com/css?family=Abel' rel='stylesheet' type='text/css'>
    <link href='https://fonts.googleapis.com/css?family=Istok+Web|Roboto+Condensed:700' rel='stylesheet' type='text/css'>

    <title>NCAA Tournament Bracket</title>
</head>
<body>
<header class="hero">
    <div class="hero-wrap">

        <h2 id="headline">Welcome!</h2>



    </div>
</header>

<section id="bracket">
    <div class="index">


        <h2 class="text-muted">How this site works</h2><br>

        <p>This site is mainly used to gain information about the NCAA tournaments of the past. <br> We're hoping that this will help you with building your bracket in the future!
        <br> Click a year below to view that tournament's bracket.</p><br>

        <p><a href = '2015.php' class = 'button'>2015 Tournament</a><br></p>
        <p><br></p>
        <p><a href = '2016.php' class = 'button'>2016 Tournament</a><br></p>
        <p><br></p>
        <p><a href = '2017.php' class = 'button'>2017 Tournament</a><br></p>
        <p><br></p>
        <p><a href = '2018.php' class = 'button'>2018 Tournament</a><br></p>
        <p><br></p>
```

Figure 10 - Screenshot of How-To Open Page Using Chrome

We recommend using Google Chrome as the browser to open our page. Once the Chrome icon is clicked, our API will launch. From there, you can choose your own path to navigate the site and find the information that you want.

Accessing the Database from PHP

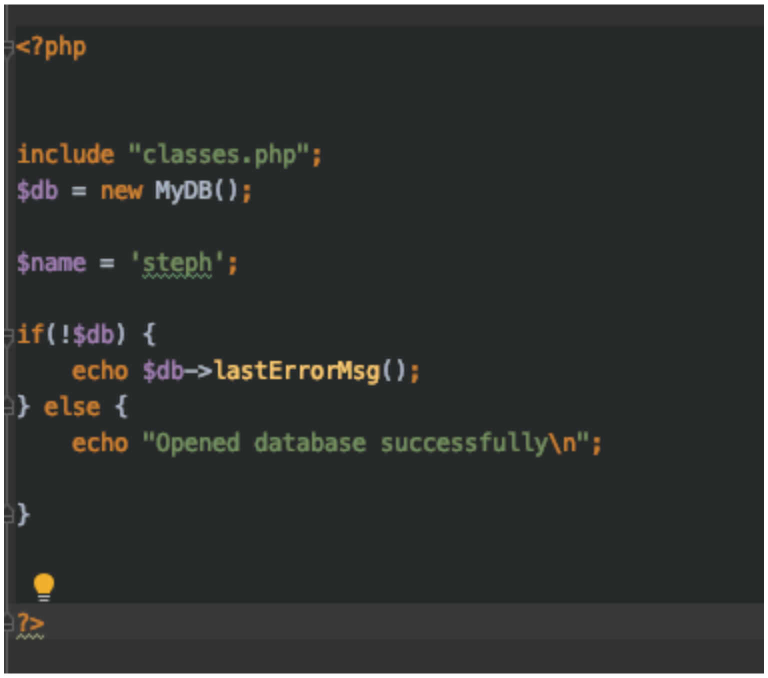
To access the DB from PHP we created a class that opens the database:



```
<?php

class MyDB extends SQLite3 {
    function __construct()
    {
        $this->open( filename: 'ncaa.db');
    }
}

?>
```

Figure 11 - Screenshot of MyDB Class

From there we created a ‘db.php’ file that creates a new instance of that class which is included in our ‘functions.php’ file which is where our queries are executed.



```php
<?php


include "classes.php";
$db = new MyDB();

$name = 'steph';

if(!$db) {
    echo $db->lastErrorMsg();
} else {
    echo "Opened database successfully\n";

}


?>
```

Figure 12 - Instantiating the MyDB Object

We then include the ‘functions.php’ file in all our other files to finalize that connection with the following code: `include "includes/db.php";`

Viewing the Database Through DB Browser for SQLite

There are many SQLite database viewers available for free to download. We’ve chosen to use DB Browser for SQLite as our database viewing platform. This has allowed us to ensure SQLite queries are being executed correctly. Upon opening DB Browser it allows you to create a new database or open an existing database.

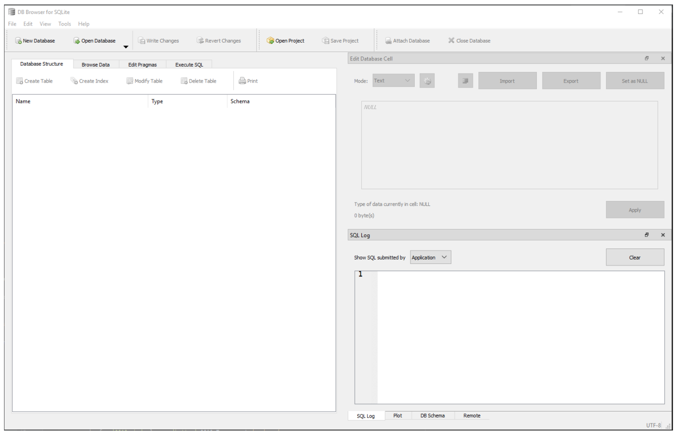
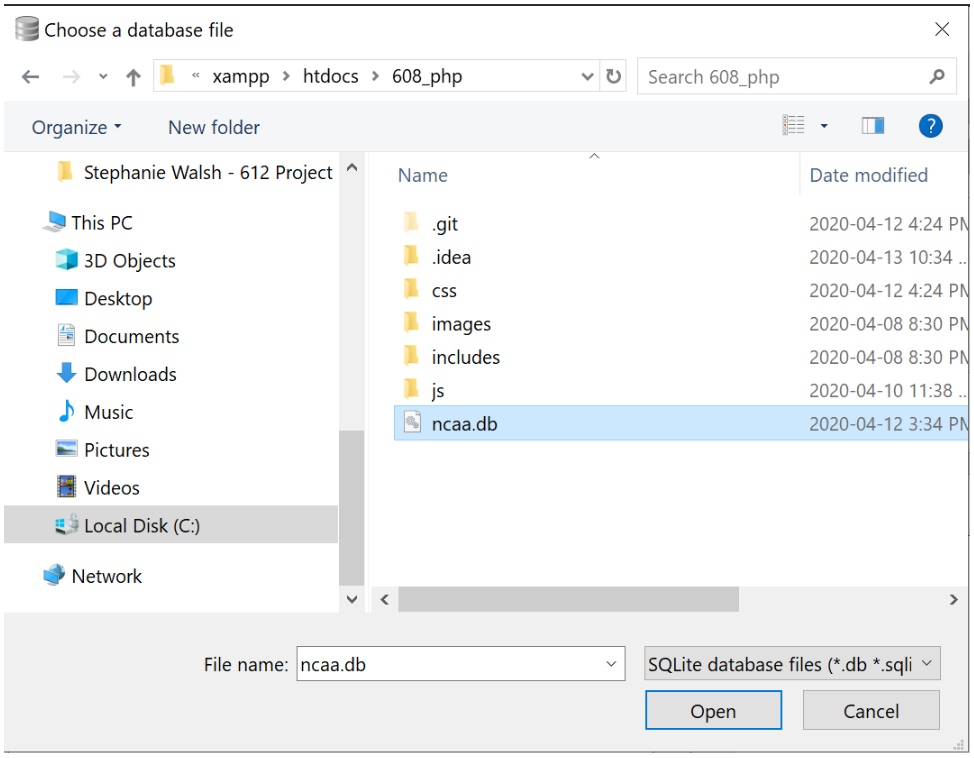


Figure 13 - DB Browser for SQLite Homepage

As we've already created our database, it needs to be opened. The database file is included in the submission and named 'ncaa.db'.

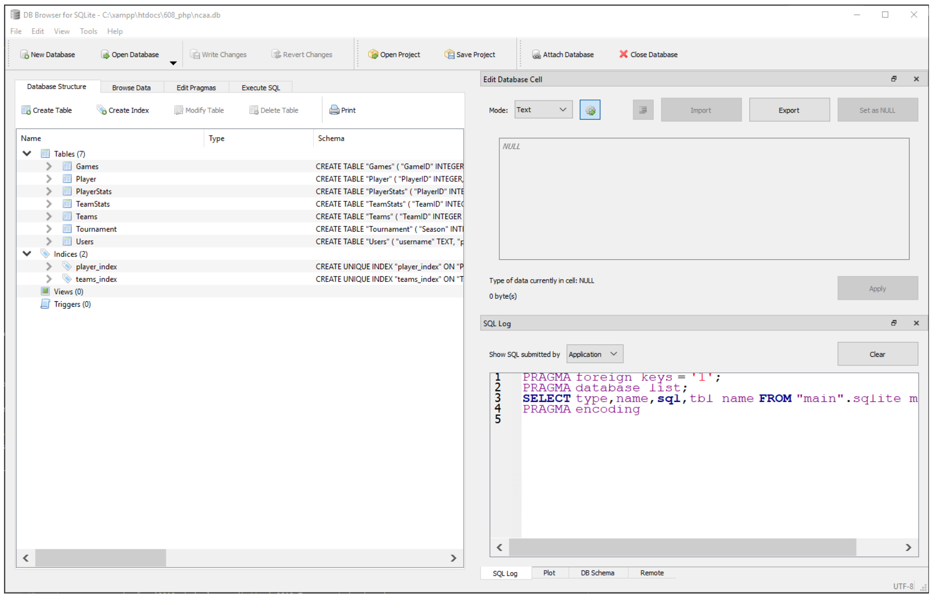


Name	Date modified
.git	2020-04-12 4:24 PM
.idea	2020-04-13 10:34 ..
css	2020-04-12 4:24 PM
images	2020-04-08 8:30 PM
includes	2020-04-08 8:30 PM
js	2020-04-10 11:38 ..
ncaa.db	2020-04-12 3:34 PM

File name: ncaa.db
SQLite database files (*.db *.sqli

Figure 14 - Opening the DB ncaa.db

Once opened it can be viewed.



Name	Type	Schema
Tables (7)		
Games		CREATE TABLE "Games" ("GameID" INTEGER
Player		CREATE TABLE "Player" ("PlayerID" INTEGER,
PlayerStats		CREATE TABLE "PlayerStats" ("PlayerID" INTE
TeamStats		CREATE TABLE "TeamStats" ("TeamID" INTE(
Teams		CREATE TABLE "Teams" ("TeamID" INTEGER
Tournament		CREATE TABLE "Tournament" ("Season" INTI
Users		CREATE TABLE "Users" ("username" TEXT, "p
Indices (2)		
player_index		CREATE UNIQUE INDEX "player_index" ON "P
teams_index		CREATE UNIQUE INDEX "teams_index" ON "T
Views (0)		
Triggers (0)		

Mode: Text

NULL

Type of data currently in cell: NULL
0 byte(s)

Show SQL submitted by: Application

```
PRAGMA foreign_keys = '1';
PRAGMA database_list;
SELECT type,name,sql,tbl_name FROM "main".sqlite_m
PRAGMA encoding
```

Figure 15 - Viewing the Database in DB Browser

To edit the database, we can either use the forms created on our site, or we do have the option to use DB Browser for other types of queries as well.

User Manual

Video of How to use Our System

We have recorded a video of how to use our system. The video is titled FullDemo.mp4 file and has been included in the submission of this report.

List of Included Files

- css folder:
  - tournament.css: includes the CSS needed to format our pages.
- includes folder:
  - classes.php: this is our DB class which opens the ncaa.db file
  - db.php: this is the function that instantiates a new DB object and checks if the database opens successfully or not.
- js folder: this folder holds the Javascript files that our templates are built on.
- 2015.php, 2016.php, 2017.php, 2018.php, 2019.php: these are all of our tournament bracket pages.
- AdminPage.php: this page is only available when you login, and has links to all our admin forms to manipulate the database.
- form_process.php: this page is the landing page after a form is submitted. It reads the forms and executes the appropriate query.
- functions.php: this is where all of our SQL queries exists to be used on our various pages.
- Game_Amendments.php: this page contains the form for Insert, Update and Delete for the Games table.
- index.php: this is our homepage that contains links to the tournament brackets and Admin Login page.
- LoginPage.php: This is where admins can login to manipulate the database.
- LoginProcess.php: this page reads the info submitted from the login form and either gives a link to the Admin page, or prompts the user to try again if the login is unsuccessful.
- ncaa.db: this is our SQLite database file.
- Player_Amendments.php: this page contains the form for Insert, Update and Delete for the Player table.
- PlayerPage.php: this page is where the user lands when they click on a player of a team. It holds all their stats from 2015 – 2019 (if they played in those seasons).
- PlayerStats_Amendments.php: this page contains the form for Insert, Update and Delete for the Player tats table.
- Team.php: this page is where the user lands when they click on a team from a tournament page. It shows the team stats for that year, as well as the player roster for that year where all of their stats are displayed as well.
- Team_Amendments.php: this page contains the form for Insert, Update and Delete for the Teams table.

- TeamStats_Amendments.php: this page contains the form for Insert, Update and Delete for the TeamStats table.
- Tournament_Amendments.php: this page contains the form for Insert, Update and Delete for the Tournaments table.

Acknowledgments

The page layouts and styles used were taken from templates provided by CodePen.[5] The original UI implementation was taken from a PHP course on Udemy.[6] From that, we were able modify the content provided by the course to a working product for our system.

References

[1] Kaggle. “Google Cloud & NCAA® ML Competition 2020-NCAAM.” Internet: https://www.kaggle.com/c/google-cloud-ncaa-march-madness-2020-division-1-mens-tournament/data, Mar. 13, 2020 [Apr. 15, 2020]
[2] NCAA. “The absurd odds of a perfect NCAA bracket.” Internet: https://www.ncaa.com/news/basketball-men/bracketiq/2020-01-15/perfect-ncaa-bracket-absurd-odds-march-madness-dream, Mar. 26, 2020 [Apr.15, 2020]
[3] NCAA. “Capital One® NCAA® March Madness Bracket Challenge.” Internet: https://bracketchallenge.ncaa.com/, [Apr. 15, 2020]
[4] USA Today. “Warren Buffett's NCAA tournament bracket challenge: Perfection earns $1 million a year for life.” Internet: https://www.usatoday.com/story/sports/ncaab/tourney/2018/03/12/warren-buffetts-ncaa-tournament-bracket-challenge-perfection-earns-1-million-year-life/415748002/, Mar. 12, 2018 [Apr.15, 2020]
[5] CodePen. “Need to Make a Tournament Bracket?” Internet: https://blog.codepen.io/2018/02/16/need-make-tournament-bracket/, Feb 16, 2018 [Apr. 15, 2020]
[6] Udemy. “PHP for Beginners - Become a PHP Master - CMS Project.” Internet: https://www.udemy.com/course/php-for-complete-beginners-includes-msql-object-oriented/, Oct. 2019 [Apr. 15, 2020]