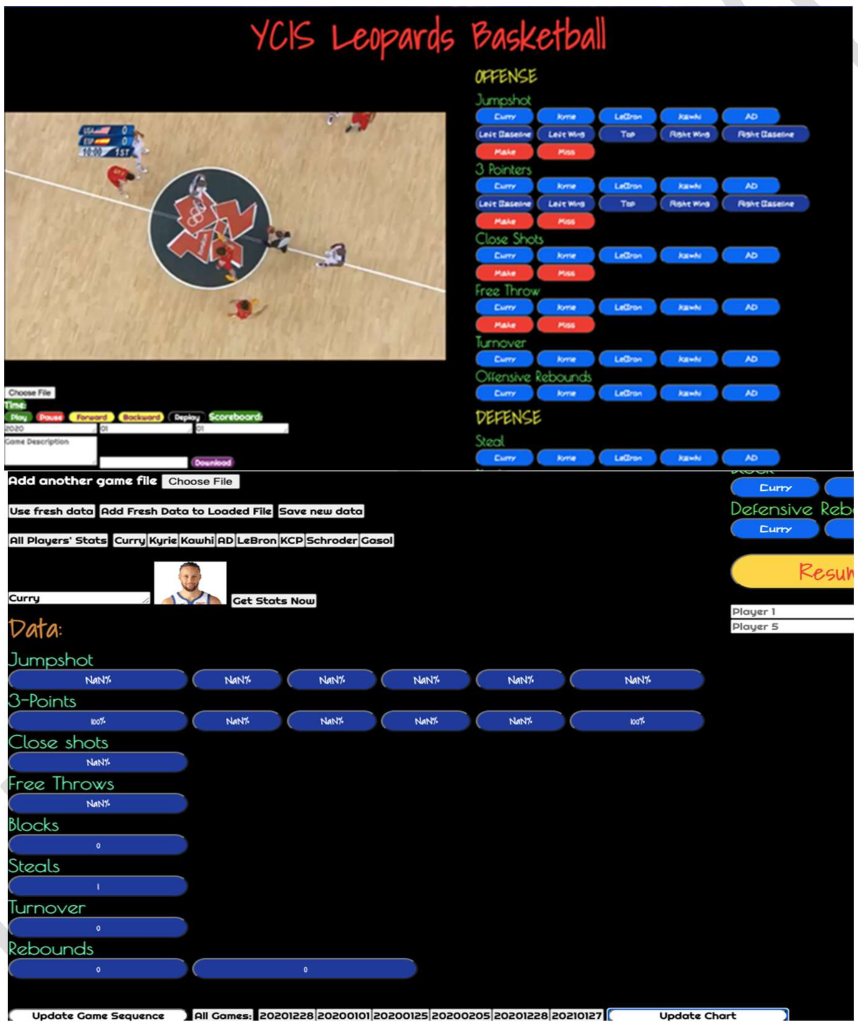
Data-Driven Basketball Performance Optimization Platform

Author : Yuchung (Troy) Wu
Date : November 2020



IB CS Internal Assessment Section Criteria A

The Scenario

The client, Mr. Yu, is the coach of my school's varsity basketball team. He encountered the problem of not enough data about his team for him to setup plays and game plans for his players. After consulting with my client, we agreed it will be more helpful if there's a tool that can give details on each player and the team's performance. I pitched him the idea of creating a program that provides a data performance analysis for the games of the season and practice. With this program, we would have access on each player's performance and consistency for every game.

Rationale for Solution

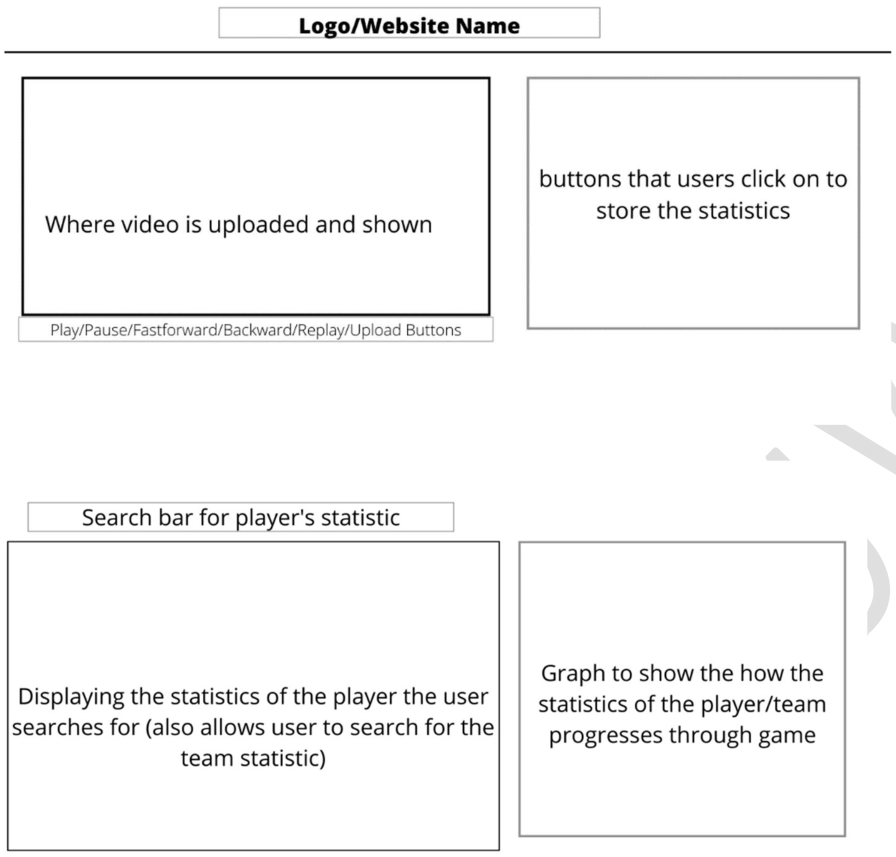
My product (only available to my client) will be a HTML file that stores and displays team's and players' data performance in a numerical and graphical display. It has the ability to upload the video of a whole game video onto the platform with some side functions of pausing and fast-forwarding. It'll store all the games' data onto the database (a txt file that can be stored on a computer) and does statistical calculation to display players' progressions in their performances numerically and graphically. I've made my decision to use HTML because it's more user friendly and allow me to decorate the website with great user interface as it contains CSS allowing me to structure layout and JavaScript helping the website runs various functions.

Success Criteria

1. The client can upload, play, pause, fast-forward, backward the mp4 file they have on the program
2. The client can watch the video and click buttons concurrently to store data of the game
3. Video can be converted into data by allowing the client to click the buttons on the program to store data into arrays in the form of .txt file
4. After finishing clicking the buttons, the client can name a file and download the file to the local computer
5. The data from the game can be added to the database, which stores all the data in all the games uploaded for every single player
6. The client can search for data of a particular day that the mp4 file was uploaded
7. The client can look at performance of their team, in numerical and graphical display

Computer Science Criteria B Design Layout, Key Algorithm, Flow chart

Design Layout


Logo/Website Name
Where video is uploaded and shown
Play/Pause/Fastforward/Backward/Replay/Upload Buttons
buttons that users click on to store the statistics
Search bar for player's statistic
Displaying the statistics of the player the user searches for (also allows user to search for the team statistic)
Graph to show the how the statistics of the player/team progresses through game

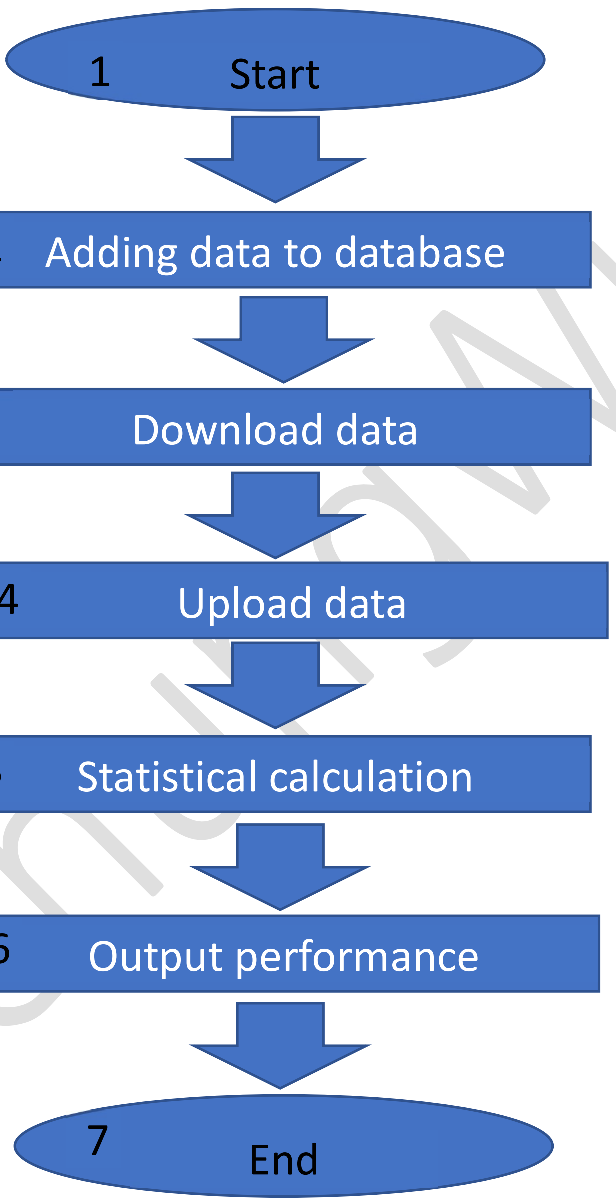
Tables of data/object: I used object-orientated JavaScript programming and store all the data into a database. After, all the games will be stored in the database and my program will carry about a function called statCalculation in which total percentage of shot will be calculated and presented numerically and graphically

Object	**eachGame: stores players data for each game (based on the video uploaded and button pressed by the users)**	**database: stores all data of the players and the team collected from all the games and videos uploaded**
time	[ ] (int): stores the time of the video when the player scores	[ ] (int): stores all the time of the video when the player scores
position	[ ] (str): stores the position of the video (ex. Left baseline) when the player scores	[ ] (str): stores all the position of the video (ex. Left baseline) when the player scores
name	[ ] (str): stores the player's name	[ ] (str): stores all the player's name
GameID	[ ] (str): the date (month, day, year) of the game/clip	[ ] (str): stores all the date (month, day, year) of the game/clip
result	[ ] (str): stores whether if the player makes a shot or not (either "make" or miss"	[ ] (str): stores all the times when the player makes a shot or not (either "make" or miss")
scoreboard	Int: stores and outputs the combined scores of all the players' points	

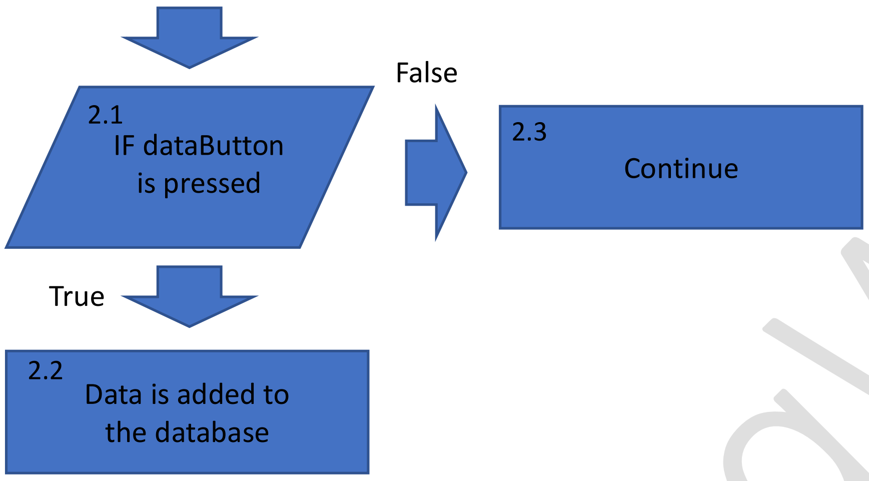
type	[ ] (str): stores the type of shot the player shot on (either layup, jumpshot, 3pointshot, freeThrow)	[ ] (str): stores all the type of shot the player shot on (either layup, jumpshot, 3pointshot, freeThrow)

Flow Chart

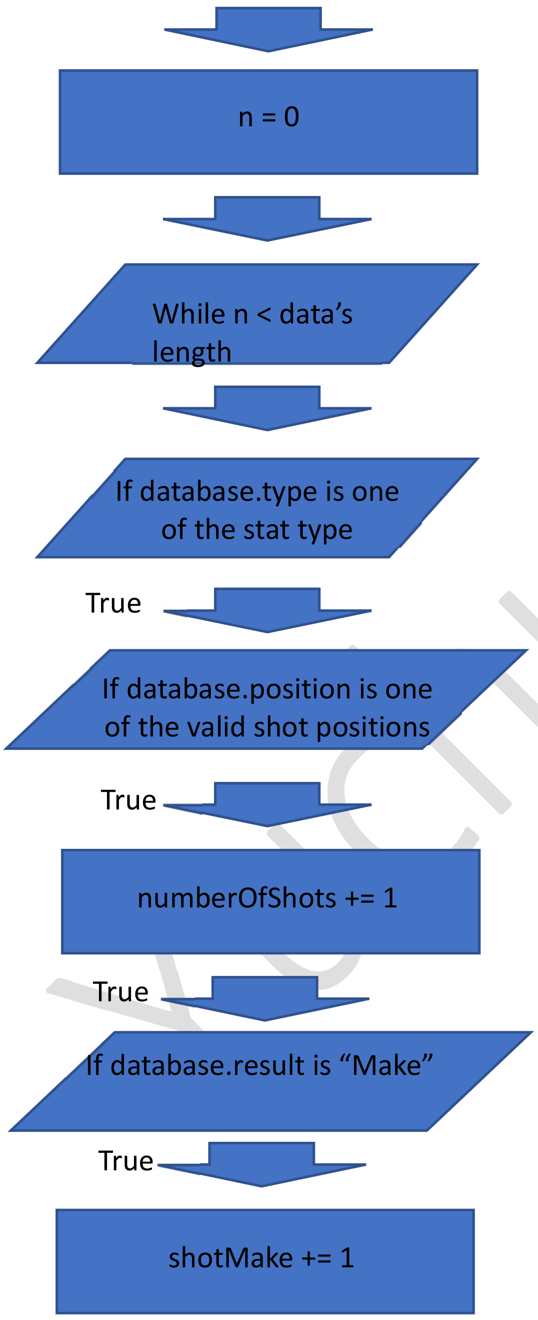
The whole process of coding



Combining games: This is necessary in situation such as combing last game's performance to today's game performance


False
2.1
IF dataButton is pressed
2.3
Continue
True
2.2
Data is added to the database

Statistical calculation and analysis: This is a function that calculates the stats of a player. There are many stats to calculate, I only display the most common one, calculating the shooting percentage.


n = 0
While n < data's length
If database.type is one of the stat type
True
If database.position is one of the valid shot positions
True
numberOfShots += 1
True
If database.result is "Make"
True
shotMake += 1

Test Plan

Test Type	**Nature of Test**	**Example**
The client can upload, play, pause, fast-forward, backward the mp4 file they have on the program	Check if video can be seen on the website after upload; Video should be played fluently (including ability to pause, replay, and fast/backward	Client can watch the video they just uploaded from their local computer
The client can watch the video and click buttons concurrently to store data of the game	Check if client can click on buttons of the data while watching the video	Client can do both tasks of watching the video and pressing the button at the same time
Video can be converted into data by allowing the client to click the buttons on the program to store the information into arrays in the form of a .txt file	Check if client can click on buttons, and check if error will show up if the user makes a mistake	Client can click on buttons on different players and they will be notified on their browser when a mistake is made

After finishing clicking the buttons, the client can name a file and download the file to the local computer	Check if client can type on the space where it requires you to name the file and type the date of the file, check if client can download the file onto whichever space the client prefers.	User can store all the data they just clicked (the database) onto their local computer
The data from the game is added to the database, which stores all the data in all the games uploaded for every single player	Check if you can download the data of the database, and check if you can upload the data of the database to add new data on the website	Client can easily download and upload file to overwrite the file
The client can search for data of a particular day that the mp4 file was uploaded	Check if user can search for stat from their most recent game	Client can access their performance on the game on March. 3rd.
The client can look at performance of their team,	Check if the numerical and graphical display of the	With a button to show the stat, client should be able to see the numerical and

in numerical and graphical display	team’s performance is visible on the website	graphical presentation of the team’s performance

Record of Task

	Candidate:				
Task Number	**Planned Action**	**Planned Outcome**	**Time Estimated**	**Target Completion Date**	**Criterion**
1	Initial Discussion	Client approved to be advisor and idea is approved	5-10 mins	June 18th , 2020	A
2	Criterion A: Situation, Rationale, Success Criterion	Complete description of situation and rationale, and listing of success criterion	1-1.5 hours	June 23rd, 2020	B
3	Learning HTML	Be fluent with the basics and some advances	5 hours	July 15th, 2020	C

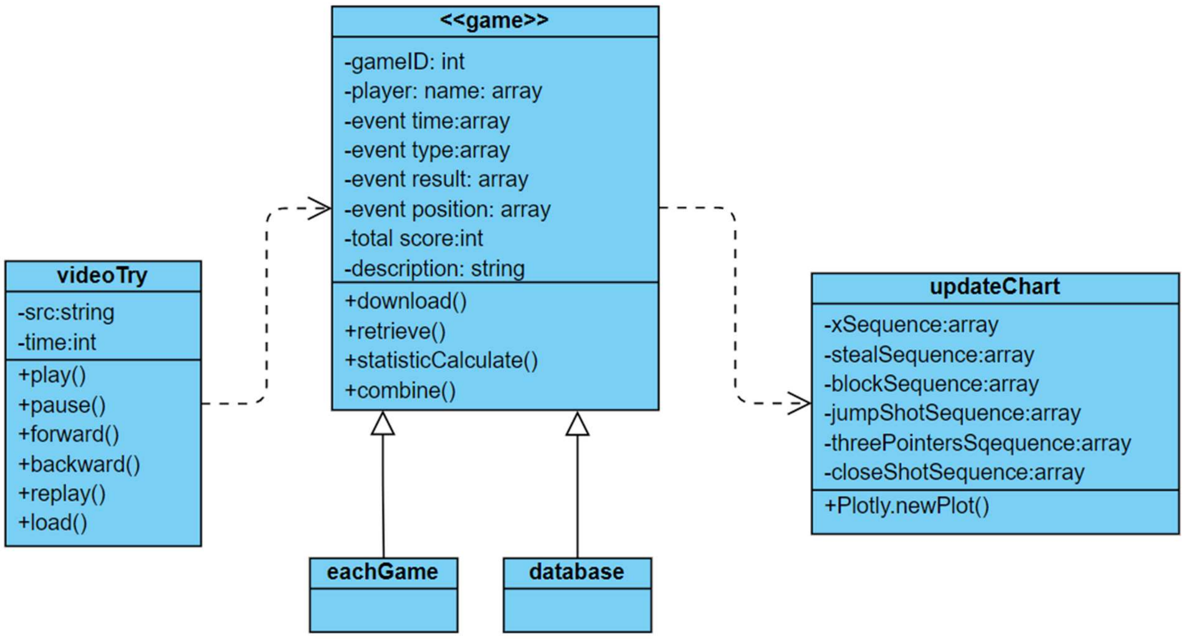
		tricks of HTML			
4	JavaScript Code Development 1	Allows user to upload mp4 file and able to perform the basic function of play, pause, replay	1.5 week	August 1st, 2020	C
5	Design Layout	A brief design that showcases what my website will look like	4 hours	Sep 5th, 2020	B
6	Flowchart	Shows how processes will run in my program, should have correct output	2 days	Sep 10th, 2020	B

5	JavaScript Code Development 2	Stores user input and allows calculation to get results	3 weeks	September 23rd, 2020	C
7	UML Diagram	A complete UML diagram that has different classes and subclasses	1 week	November 7th , 2020	C
8	JavaScript Drawing Graphs to evaluate players' performances	The website displays graphs of players performances through games	4 weeks	December 12th , 2020	C
9	Finish all coding	The website will exactly be what the client wanted	5 weeks	January 24th, 2020	C

		in the first place			
10	Polishing up Criteria A and B	The final document should tick off all the requirements stated in the marking of IA	1 day	January 26th, 2020	A, B

Criterion C: Explaining complex code

UML Class Diagram



List of technique used:

A. Object-Oriented Programming: JavaScript/CSS/HTML
B. Debugging by browser-inspection
C. Video handling
D. Object/String/Blob conversion and parsing
E. Saving a file in JavaScript
F. Reading a file in JavaScript
G. Dynamically adding/removing children to/from a DOM element
H. Canvas 2D image rendering
I. Chart-plotting with Plotly.js

Algorithms/functions

Algorithms	Purposes	Comments
downloadFile()	To save new game data as a file in local drive	The file includes all event information (class: time/player/action type/result/spot). The user shall input date as game ID and could optionally add game description. Then all the data would eb saved as text file in the file name decided by the user.
checkDataLength()	To make sure of data integrity before any new input	It examines whether all the needed data fields are filled; otherwise, it would infer the latest input to be problematic and remove it. The video will backtrack to the previous correct time stamp so that the user could replay/re-input.
toRetrieveData()	To load a game data file from local drives. This game data will be automatically combined as the database to calculate game/player statistic.	The string in text file will be read in, parsed and summed up to game class/object data.
useFreshData()	To use only the fresh input as database without combining it with any previous data.	The fresh input alone will be used as the database no matter it is downloaded.
addFreshData()	To add the fresh data to whatever is already retrieved from local drives	To include the fresh data in existing database
saveTheCombined()	To save the current database as a whole to a local drive	The database can cosist of fresh input only, previous game data from sequentially retrieving several files from a local drive or the combination of the fresh data and sequentially retrieved game data.
statCal()	To calculate individual player's statistics, at the user's pick, based on the existing database.	It reflects a player's performance on jump shots, three pointers, close shot, free throws, blocks, steals, turnovers and rebounds. For jump shots and three pointers, yields at different shooting spots will be displayed. It will inspire the user of training plans for individuals.

updateGameSequence()	To reorganize and list all games in the database in order of the game date	It is not required for the user to combine historical game data in order. It is also all right if the user add redundant games. This function will sort the game neatly and list them in chronological order.
updateChart()	To creates chart reflecting team performance over time (game date).	Total scores, jump shots, three pointers, close shots, steal, blocks, turnovers and rebounds will be plotted for the user to easily observe the team progress over time.

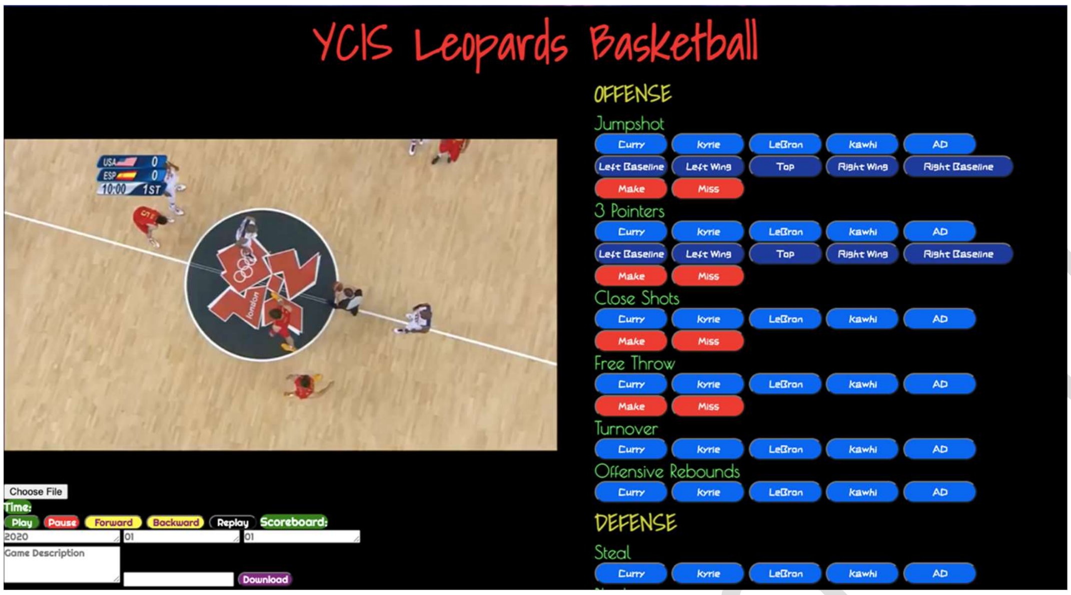
Tools used in coding

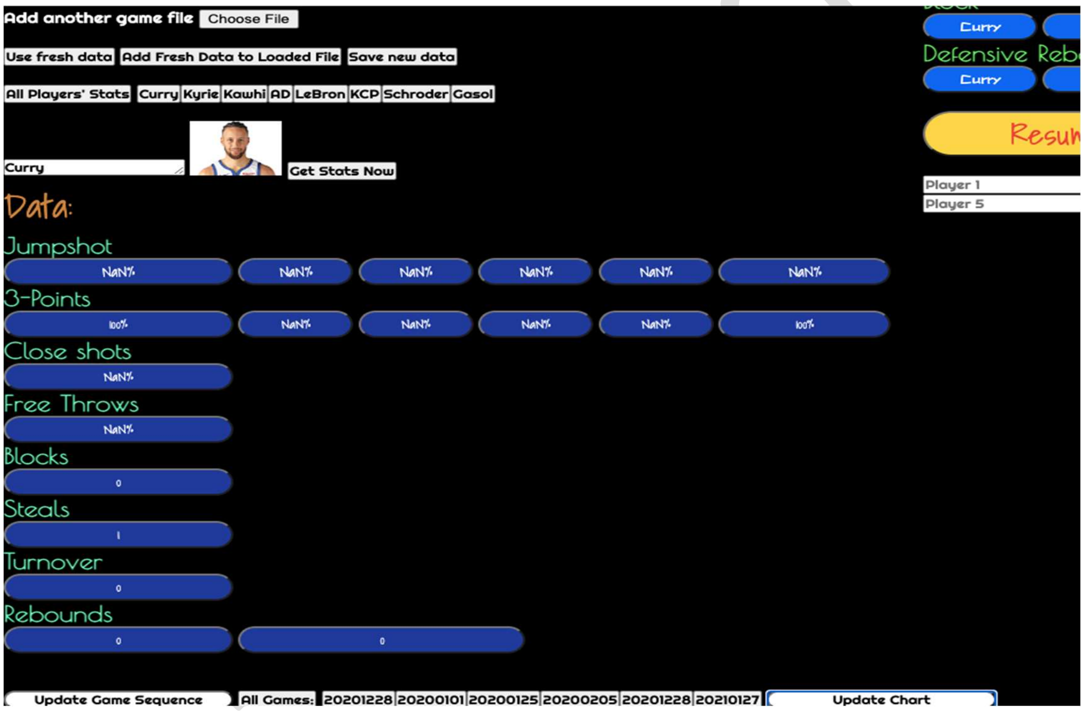
Tools used in coding	Reason for selection	What it does
Plotly.newPlot	Plotly.js is a high-level, declarative charting library with over 40 chart types, including 3D charts, statistical graphs, and SVG maps. plotly.js is free and open source	Plotly.newPlot draws a new plot in an <div> element to display team performance over time
HTML video	Interactive video handling and data input through webpage is one of the main features for application. In addition to pure play/resume, I build proprietary forward/backward operation; particularly, the video will backtrack itself to a correct timestamp when a user input is identified problematics, which calls for deep undersatdning and control in video.	HTML video provides methods, properties, and events for the <video> elements.
Canvas	once a player is picked for statistics analysis, the	.drawImage() method of the Canvas 2D API

	corresponding picture would be shown. We need API to render the 2D context for displaying.	provides sevral ways to draw an image onto the surface of a <canvas> element
Blob	Blobs are objects that are used to represent raw immutable data with the type and size of data information. It serves as the basis for bowser to store and work on file contents. In fact, the File object is a special extension of the Blob interface.	Game data is class/object data. API Blob organizes it in a per-defined way to make it an binary large object(blob) including type and size information for File API to process the data with
createObjectURL	When game data is made into a blob by API Blob. File API needs a path, URL, to reach it.	The URL.createObjectURL() static method creates a string containing a URL, called DOMstring, representing the object.
.dataset.downloadURL	The logic of downloading a file can be broken down as follows: 1.Create an object URL for the blob object 2.Create an anchor element (<a></a>) 3.Set the href attribute of the anchor element to the created object URL 4.Set the download attribute to the filename of the file to be downloaded. This forces the anchor element to trigger a file download when it is clicked	It creates a download link (<a></a> element) that can be clicked in order to download the content of the blob, just like a regular file download.
.readAsText	The FileReader interface has pretty good browser support in reading blob data as text by FileReader.readAsText()	Reads the content of a specified file for game data The FileReader object lets web applications asynchronously read the

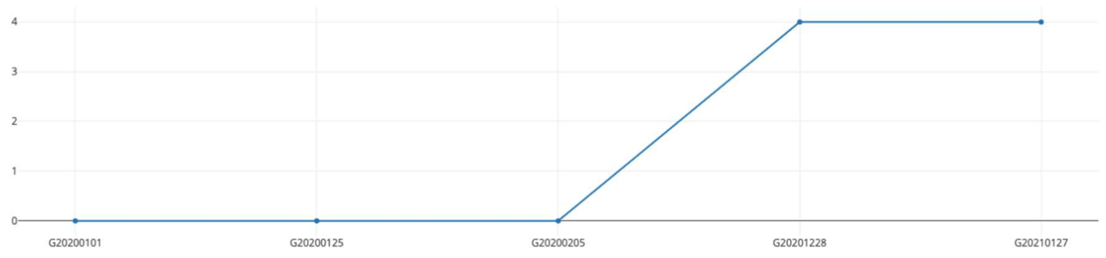
		contents of files (or raw data buffers) stored on the user's computer, using File or Blob objects to specify the file or data to read.
JSON.parse	The JSON.parse() method parses a JSON string, constructing the JavaScript value or object described by the string.	It parses string (the text content of game files) content into a javascript object (game class data)
JSON. stringify	JSON.stringify() method converts the properties of a JavaScript object to a JSON string, which consists of original strings with additional ones to describe object structure.	Converts a javaScript object (i.e.game data in this app) or value to a JSON string.
Math.round()	Unlike python, web browsers doesn’t have API in formatting digital result. I use Math.round() to manipulate the digital result and discard the unwanted spaces.	To format decimal result to fixed decimal spaces

Full layout

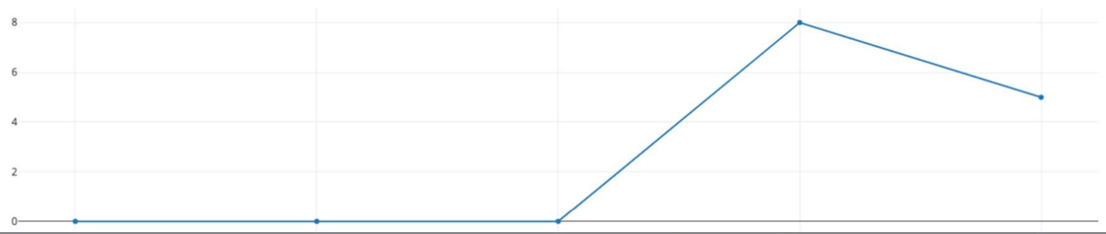

YCIS Leopards Basketball
OFFENSE
Jumpshot
Curry
Kyrie
LeBron
Kawhi
AD
Left Baseline
Left Wing
Top
Right Wing
Right Baseline
Make
Miss
3 Pointers
Close Shots
Free Throw
Turnover
Offensive Rebounds
DEFENSE
Steal
Choose File
Time:
Play
Pause
Forward
Backward
Replay
Scoreboard:
2020
01
01
Game Description
Download

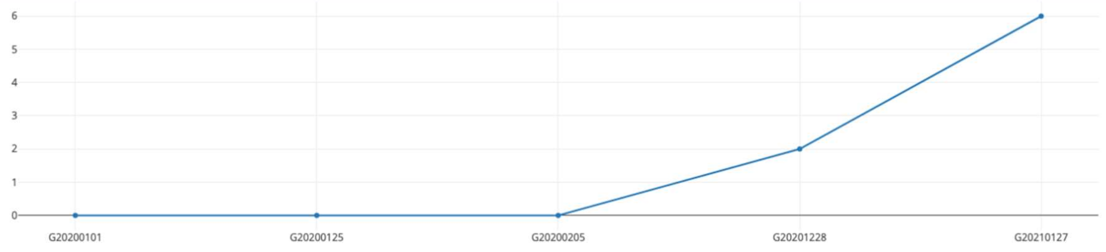

Add another game file
Choose File
Use fresh data
Add Fresh Data to Loaded File
Save new data
All Players' Stats
Curry
Kyrie
Kawhi
AD
LeBron
KCP
Schroder
Gasol
Curry
Get Stats Now
Data:
Jumpshot
NaN%
3-Points
100%
Close shots
Free Throws
Blocks
0
Steals
1
Turnover
Rebounds
Update Game Sequence
All Games:
20201228
20200101
20200125
20200205
20201228
20210127
Update Chart
Defensive Reb
Resu
Player 1
Player 5

Jumpshot

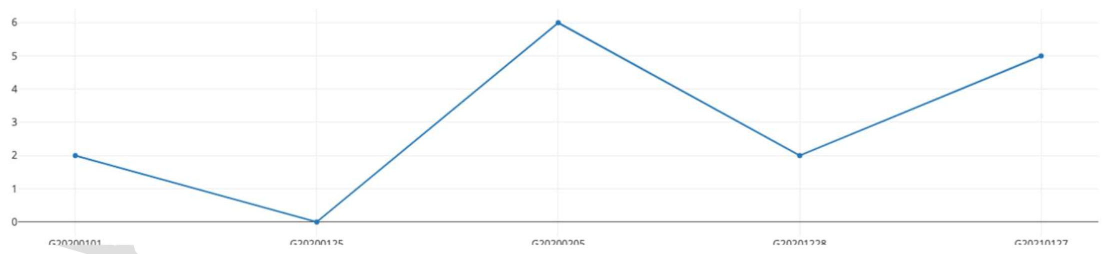


Three Points

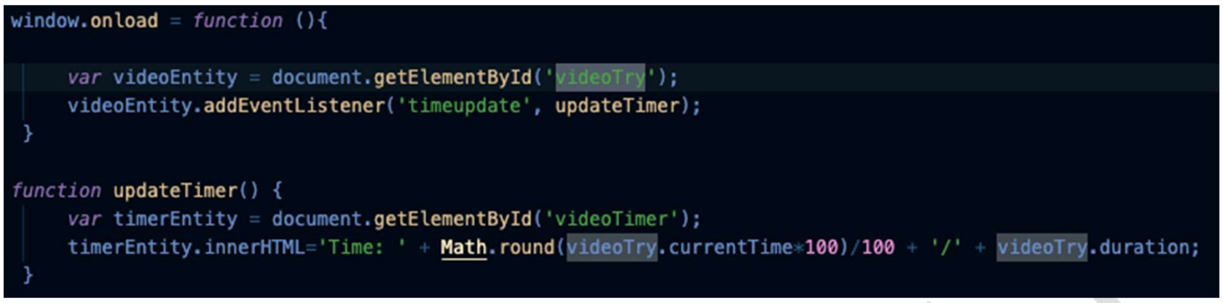




Steal



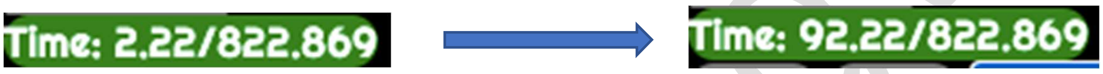
Video functions

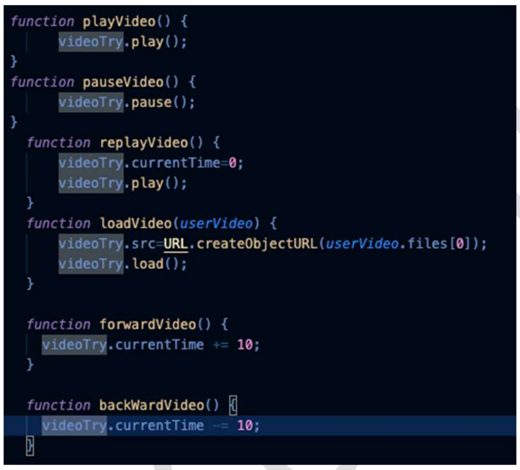


```
window.onload = function (){

    var videoEntity = document.getElementById('videoTry');
    videoEntity.addEventListener('timeupdate', updateTimer);
 }

function updateTimer() {
    var timerEntity = document.getElementById('videoTimer');
    timerEntity.innerHTML='Time: ' + Math.round(videoTry.currentTime*100)/100 + '/' + videoTry.duration;
 }
```

Window.onload: when the HTML file is loaded on your browser, this function is run immediately, and when the video on the website is played ('timeUpdate'), the updateTimer() is run. This is the conditions for addEventListener.

videoTimer (id) is where the webpage displays the time. This function updates the time and prints the updated resulted on 'videoTimer'.

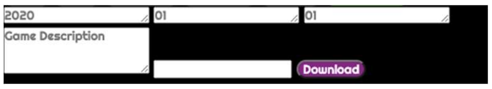


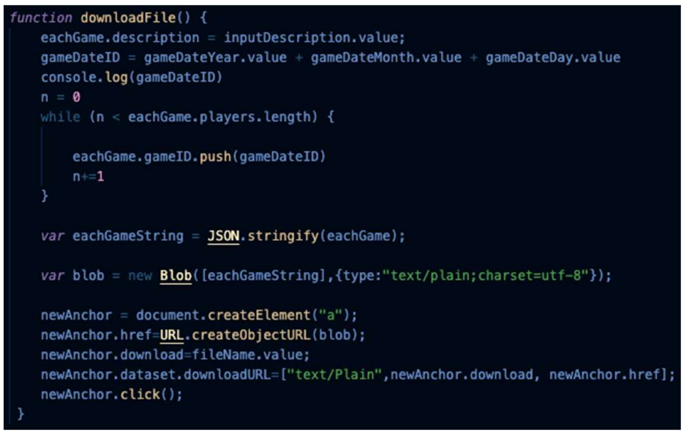


```
function playVideo() {
        videoTry.play();
}
function pauseVideo() {
        videoTry.pause();
}
  function replayVideo() {
        videoTry.currentTime=0;
        videoTry.play();
  }
  function loadVideo(userVideo) {
        videoTry.src=URL.createObjectURL(userVideo.files[0]);
        videoTry.load();
  }

  function forwardVideo() {
    videoTry.currentTime += 10;
  }

  function backWardVideo() {
    videoTry.currentTime -= 10;
  }
```

Video and audio are HTMLMediaElements, so there are special application programming interfaces available like .play() and .pause() for them

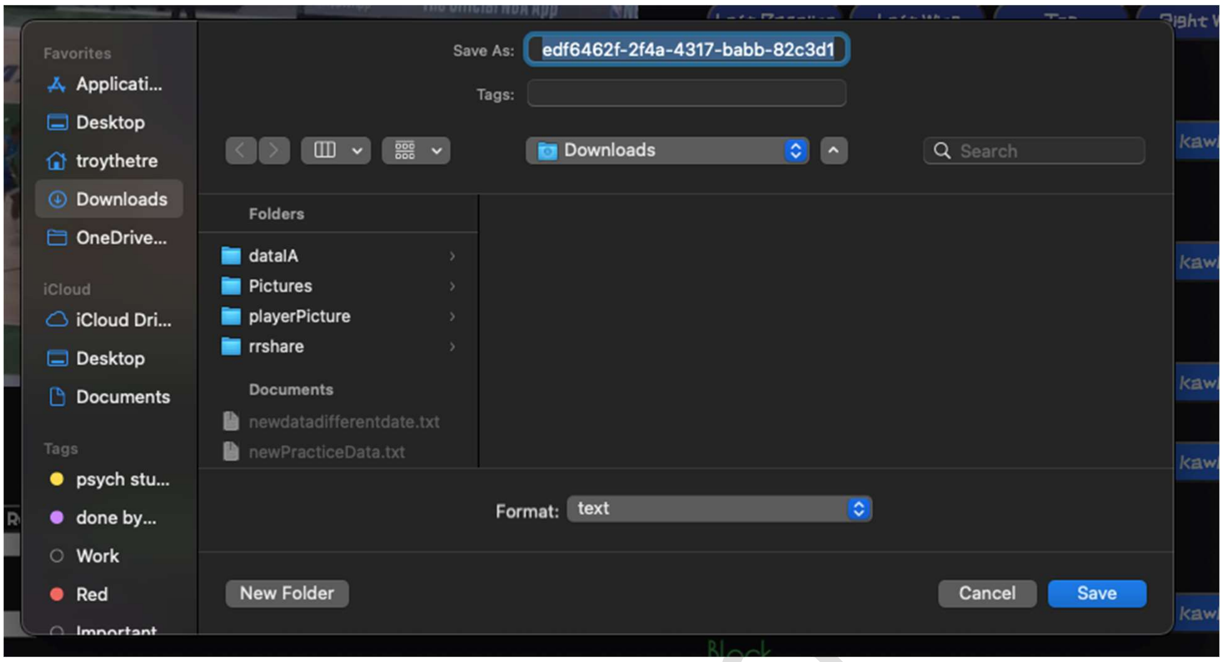
The supplementary parameters to make a game data unique (gameID, gameDescription, file name) are required to put down in text area elements before the user clicks “Download”.





```
function downloadFile() {
    eachGame.description = inputDescription.value;
    gameDateID = gameDateYear.value + gameDateMonth.value + gameDateDay.value
    console.log(gameDateID)
    n = 0
    while (n < eachGame.players.length) {

        eachGame.gameID.push(gameDateID)
        n+=1
    }

    var eachGameString = JSON.stringify(eachGame);

    var blob = new Blob([eachGameString],{type:"text/plain;charset=utf-8"});

    newAnchor = document.createElement("a");
    newAnchor.href=URL.createObjectURL(blob);
    newAnchor.download=fileName.value;
    newAnchor.dataset.downloadURL=["text/Plain",newAnchor.download, newAnchor.href];
    newAnchor.click();
 }
```

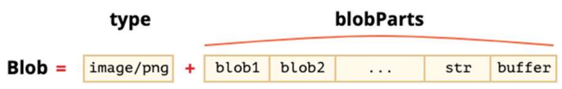
eachGame is an object which stores stats of a game and it contains properties like the gameDateID, which stores the date of the game data. JSON.stringify converts a javaScript object into a string; then the string, together with type/size information, is further converted to a “blob”, binary data chunk. createURL() creates the URL for this blob so that downloadURL() could reach this blob and save it into a local drive as a text file.


Save As: edf6462f-2f4a-4317-babb-82c3d1

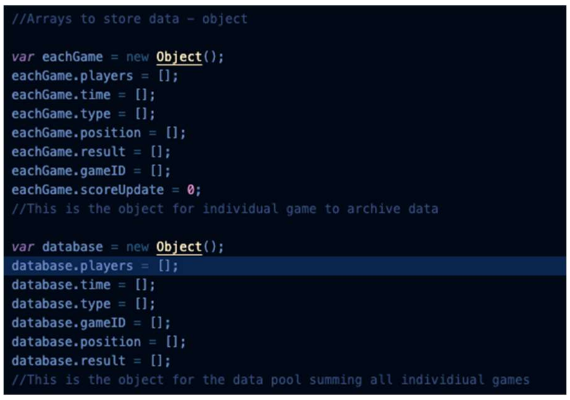
Tags:

Format: text

The Blob object represents a blob, which is a file-like object of immutable, raw data; they can be read as text or binary data

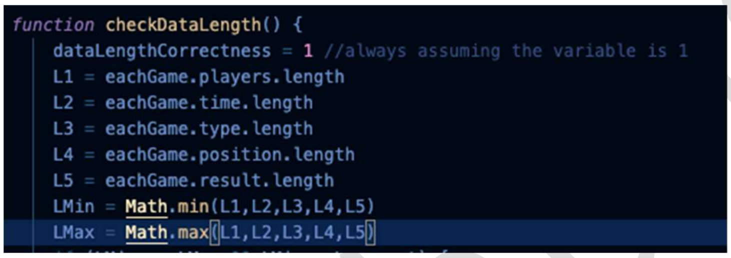


The database: (linked to criterion B)



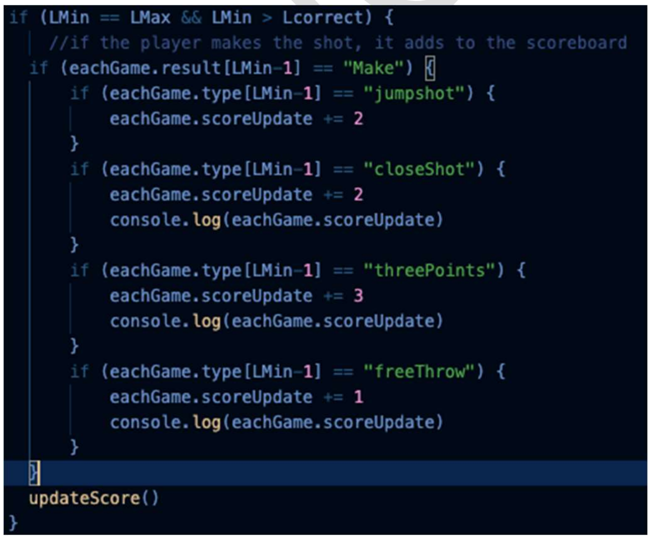
```
//Arrays to store data - object

var eachGame = new Object();
eachGame.players = [];
eachGame.time = [];
eachGame.type = [];
eachGame.position = [];
eachGame.result = [];
eachGame.gameID = [];
eachGame.scoreUpdate = 0;
//This is the object for individual game to archive data

var database = new Object();
database.players = [];
database.time = [];
database.type = [];
database.gameID = [];
database.position = [];
database.result = [];
//This is the object for the data pool summing all individiual games
```

Sanity check for data: This app makes sure there's no previous error before a new input by using checkDataLength(): All the event properties of the object should have the same length. If not, error message will be delivered through "alert".



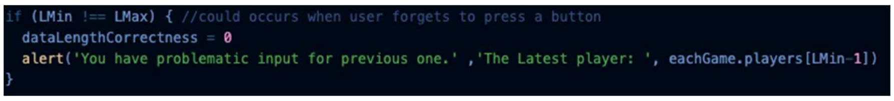
```
function checkDataLength() {
    dataLengthCorrectness = 1 //always assuming the variable is 1
    L1 = eachGame.players.length
    L2 = eachGame.time.length
    L3 = eachGame.type.length
    L4 = eachGame.position.length
    L5 = eachGame.result.length
    LMin = Math.min(L1,L2,L3,L4,L5)
    LMax = Math.max(L1,L2,L3,L4,L5)
```

There's a variable, dataLengthCorrectness, set to 1 (assumes it's correct | 1 = correct, 0 = wrong). And L1-5 are just the length of the properties of the object. By comparing the length of the properties, we will know if the user over-presses and forgot to press button which would make the data ineligible for statistical calculation.

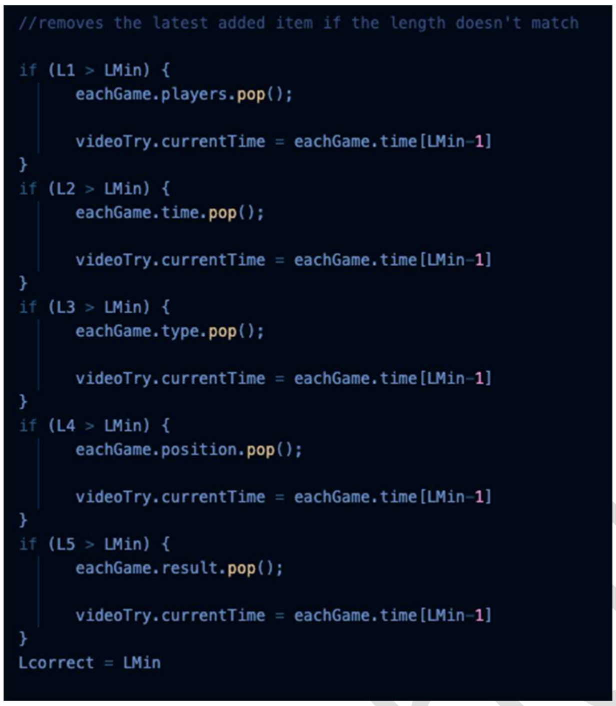


```
if (LMin == LMax && LMin > Lcorrect) {
    //if the player makes the shot, it adds to the scoreboard
    if (eachGame.result[LMin-1] == "Make") {
        if (eachGame.type[LMin-1] == "jumpshot") {
            eachGame.scoreUpdate += 2
        }
        if (eachGame.type[LMin-1] == "closeShot") {
            eachGame.scoreUpdate += 2
            console.log(eachGame.scoreUpdate)
        }
        if (eachGame.type[LMin-1] == "threePoints") {
            eachGame.scoreUpdate += 3
            console.log(eachGame.scoreUpdate)
        }
        if (eachGame.type[LMin-1] == "freeThrow") {
            eachGame.scoreUpdate += 1
            console.log(eachGame.scoreUpdate)
        }
    }
    updateScore()
}
```

LMin == LMax reflects the data integrity (dataLengthCorrectness = 1)
The score will only be updated if with positive sanity check

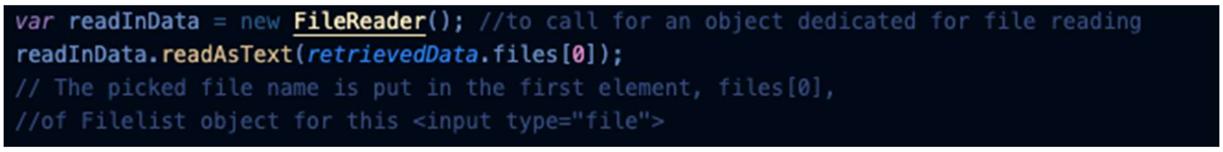


```
if (LMin !== LMax) { //could occurs when user forgets to press a button
  dataLengthCorrectness = 0
  alert('You have problematic input for previous one.' ,'The Latest player: ', eachGame.players[LMin-1])
}
```

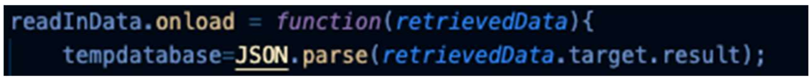


```
//removes the latest added item if the length doesn't match

if (L1 > LMin) {
     eachGame.players.pop();

     videoTry.currentTime = eachGame.time[LMin-1]
}
if (L2 > LMin) {
     eachGame.time.pop();

     videoTry.currentTime = eachGame.time[LMin-1]
}
if (L3 > LMin) {
     eachGame.type.pop();

     videoTry.currentTime = eachGame.time[LMin-1]
}
if (L4 > LMin) {
     eachGame.position.pop();

     videoTry.currentTime = eachGame.time[LMin-1]
}
if (L5 > LMin) {
     eachGame.result.pop();

     videoTry.currentTime = eachGame.time[LMin-1]
}
Lcorrect = LMin
```

Retrieving data



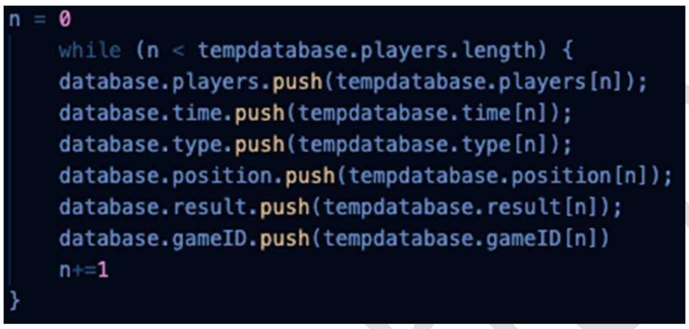
```
var readInData = new FileReader(); //to call for an object dedicated for file reading
readInData.readAsText(retrievedData.files[0]);
// The picked file name is put in the first element, files[0],
//of Filelist object for this <input type="file">
```



```
readInData.onload = function(retrievedData){
    tempdatabase=JSON.parse(retrievedData.target.result);
```

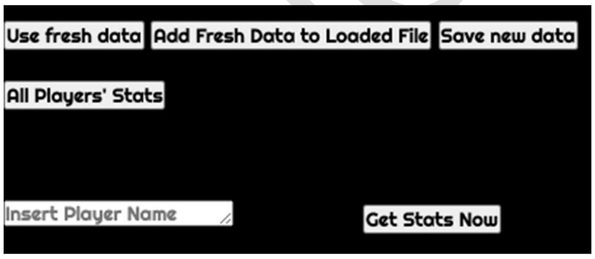
FileReader reads the data uploaded and and uses JSON.parse to turn it into a javaScript object, calling it a tempdatabase (temporary Database)

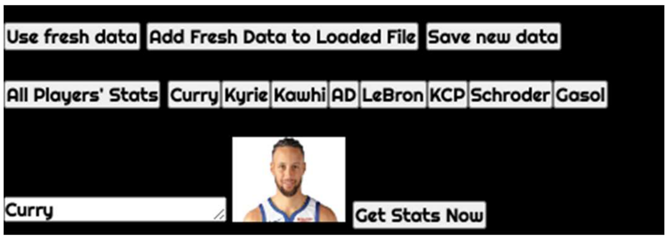
Adding the properties of the object of the database from the tempDataBase (adding the data of the uploaded file into the data – achieving the success criteria of adding data to the database):

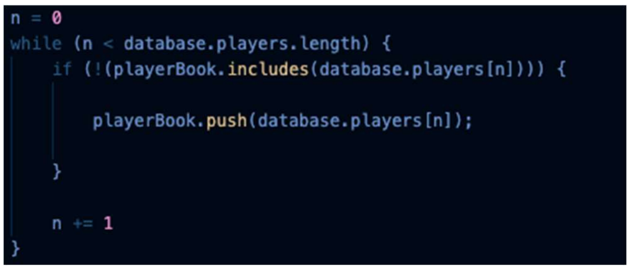


```
n = 0
    while (n < tempdatabase.players.length) {
    database.players.push(tempdatabase.players[n]);
    database.time.push(tempdatabase.time[n]);
    database.type.push(tempdatabase.type[n]);
    database.position.push(tempdatabase.position[n]);
    database.result.push(tempdatabase.result[n]);
    database.gameID.push(tempdatabase.gameID[n])
    n+=1
}
```

Next, the program will add shortcut buttons to search for the player's statistic (if confused, look at the criterion D video first)

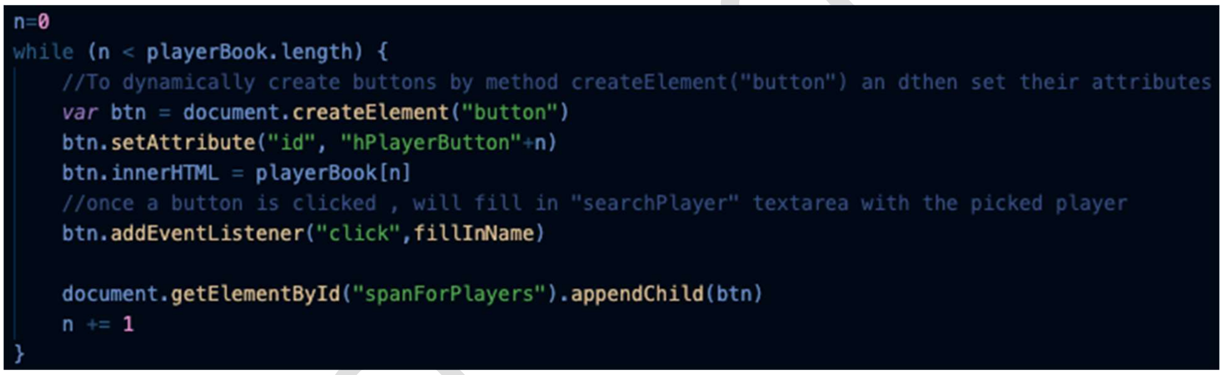






```
n = 0
while (n < database.players.length) {
    if (!(playerBook.includes(database.players[n]))) {

        playerBook.push(database.players[n]);

    }

    n += 1
}
```

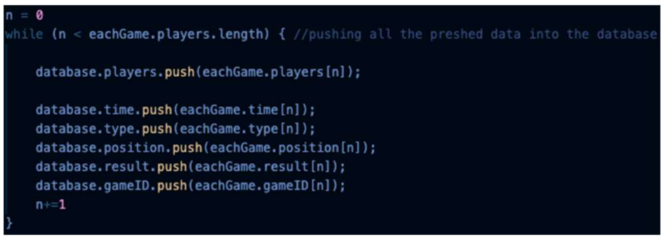
playerBook is an empty array storing all the players’ names, it adds the players’ names if they are not in the playerBook.



```
n=0
while (n < playerBook.length) {
    //To dynamically create buttons by method createElement("button") an dthen set their attributes
    var btn = document.createElement("button")
    btn.setAttribute("id", "hPlayerButton"+n)
    btn.innerHTML = playerBook[n]
    //once a button is clicked , will fill in "searchPlayer" textarea with the picked player
    btn.addEventListener("click",fillInName)

    document.getElementById("spanForPlayers").appendChild(btn)
    n += 1
}
```

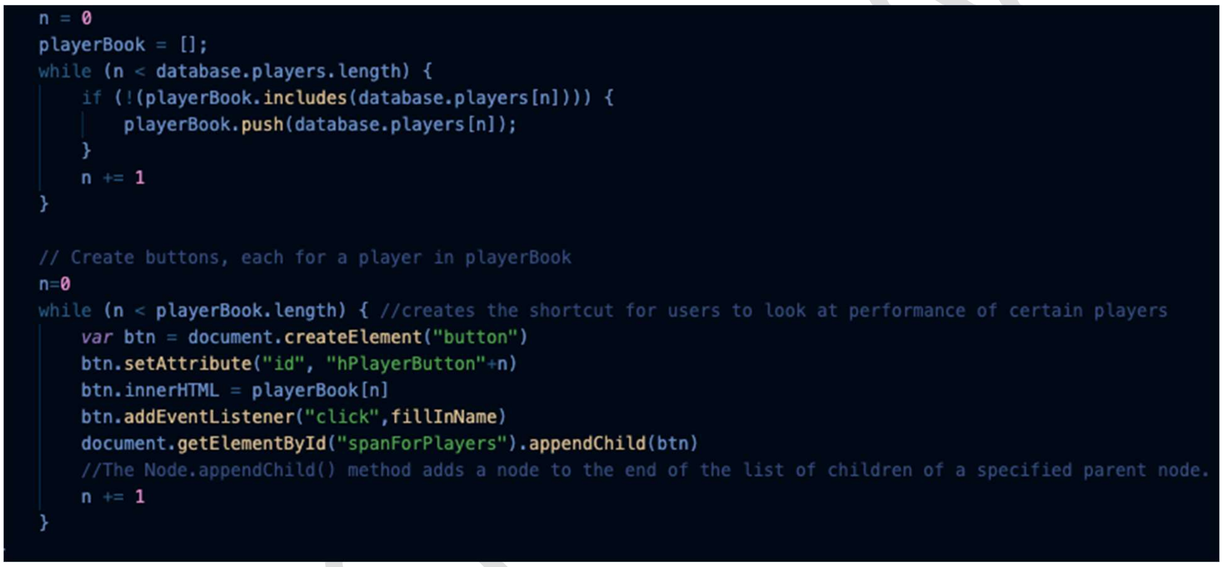
In the while loop, new buttons are created, and the buttons will show the names of the new players. When the button was clicked, it can then be used in statCalculation later on (will explain more few pages down).

useFreshData(): this function will store the data that the user just clicked on into the database for statistical analysis



```javascript
n = 0
while (n < eachGame.players.length) { //pushing all the preshed data into the database

    database.players.push(eachGame.players[n]);

    database.time.push(eachGame.time[n]);
    database.type.push(eachGame.type[n]);
    database.position.push(eachGame.position[n]);
    database.result.push(eachGame.result[n]);
    database.gameID.push(eachGame.gameID[n]);
    n+=1
}
```

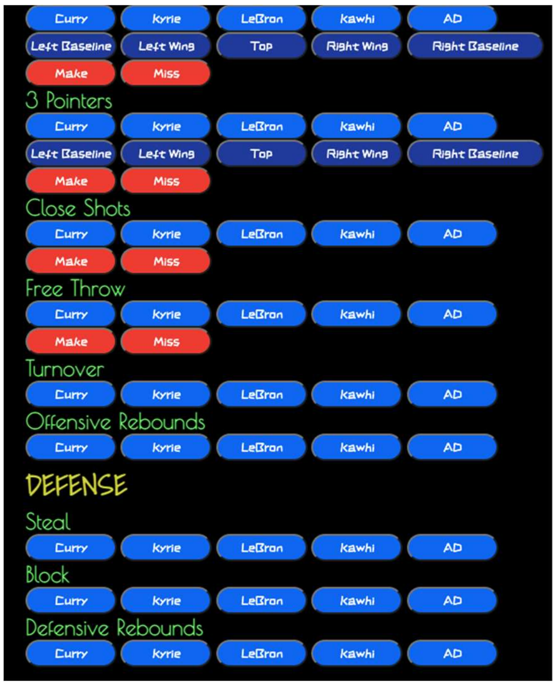
The while loop will continue to run until n is equals to the length of the array of the object, as long as we are in the loop, all the data on the program will be stored into the database

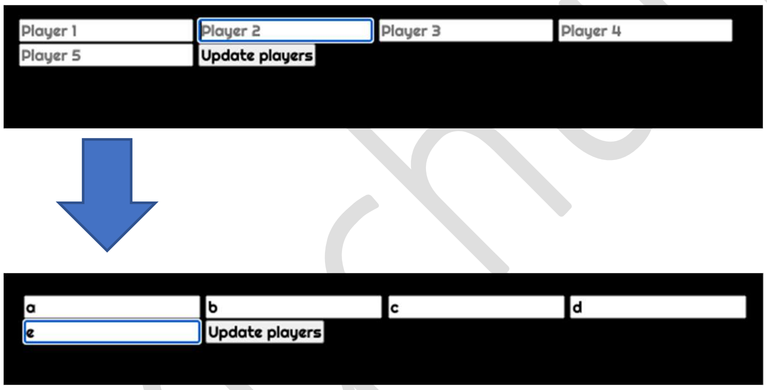


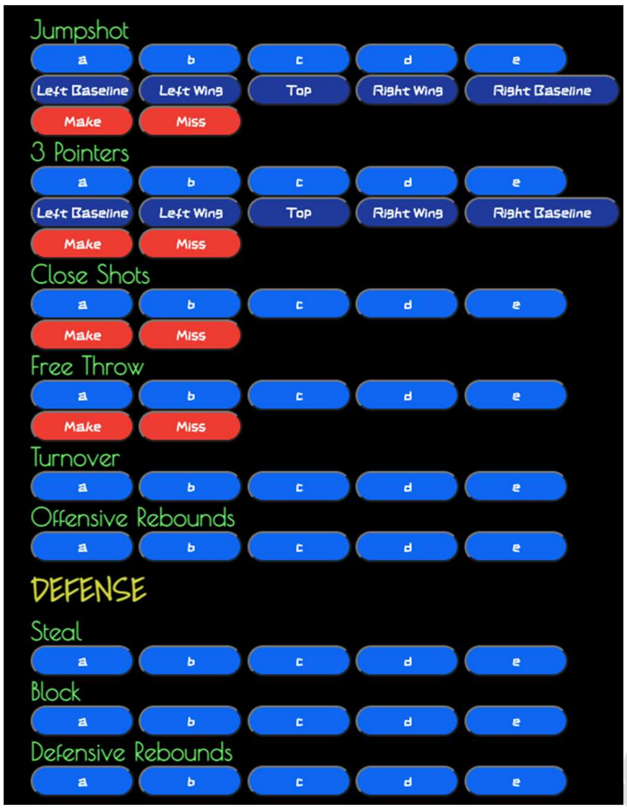
```javascript
n = 0
playerBook = [];
while (n < database.players.length) {
    if (!(playerBook.includes(database.players[n]))) {
        playerBook.push(database.players[n]);
    }
    n += 1
}

// Create buttons, each for a player in playerBook
n=0
while (n < playerBook.length) { //creates the shortcut for users to look at performance of certain players
    var btn = document.createElement("button")
    btn.setAttribute("id", "hPlayerButton"+n)
    btn.innerHTML = playerBook[n]
    btn.addEventListener("click",fillInName)
    document.getElementById("spanForPlayers").appendChild(btn)
    //The Node.appendChild() method adds a node to the end of the list of children of a specified parent node.
    n += 1
}
```

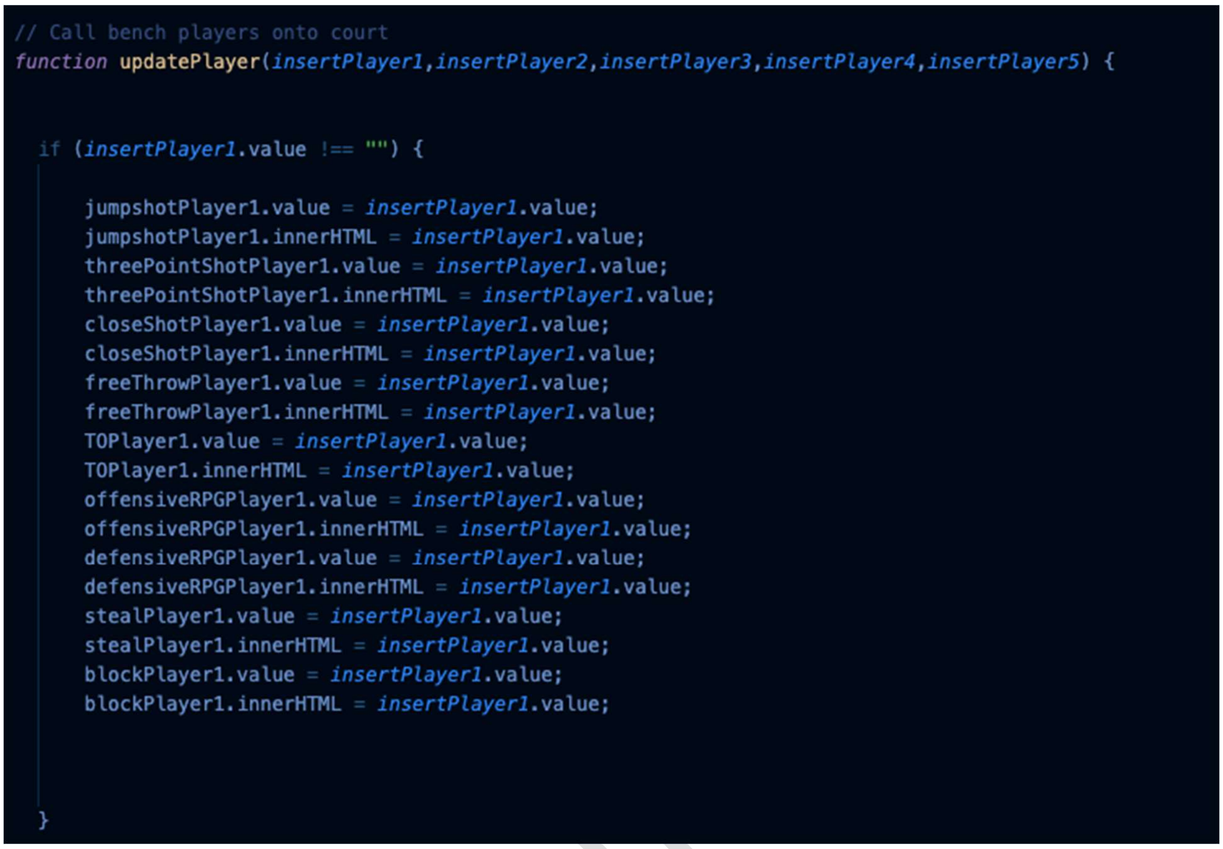
This, like the function of to retrieve data, uses the array, playerBook, and make use of while loop .createElements to create buttons for the players shortcut for the players' performances.

Updating the player for substitution: This makes use of .value and .innerHTML, which were both something we learned in our CS class.





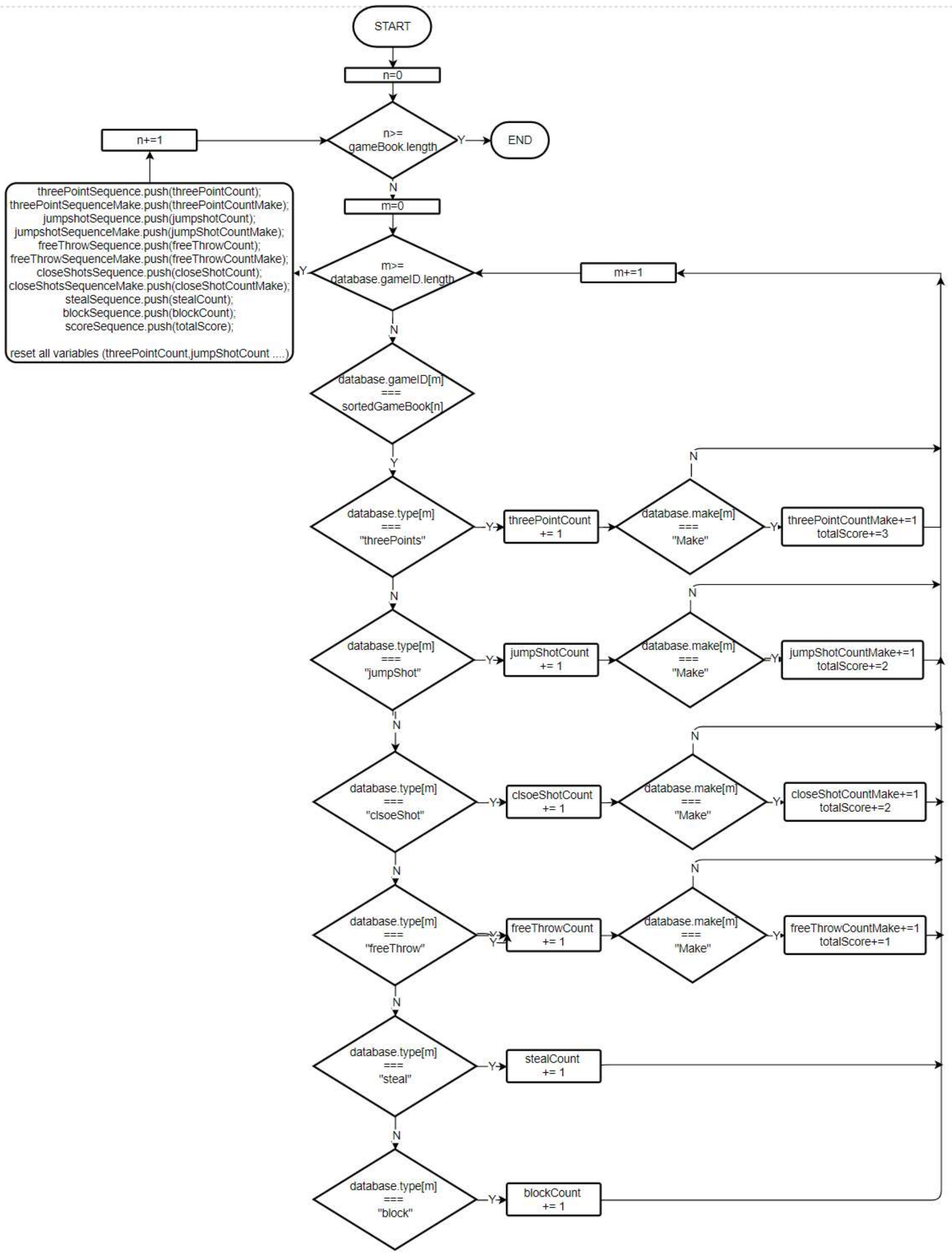

Jumpshot
a b c d e
Left Baseline Left Wing Top Right Wing Right Baseline
Make Miss
3 Pointers
a b c d e
Left Baseline Left Wing Top Right Wing Right Baseline
Make Miss
Close Shots
a b c d e
Make Miss
Free Throw
a b c d e
Make Miss
Turnover
a b c d e
Offensive Rebounds
a b c d e
DEFENSE
Steal
a b c d e
Block
a b c d e
Defensive Rebounds
a b c d e



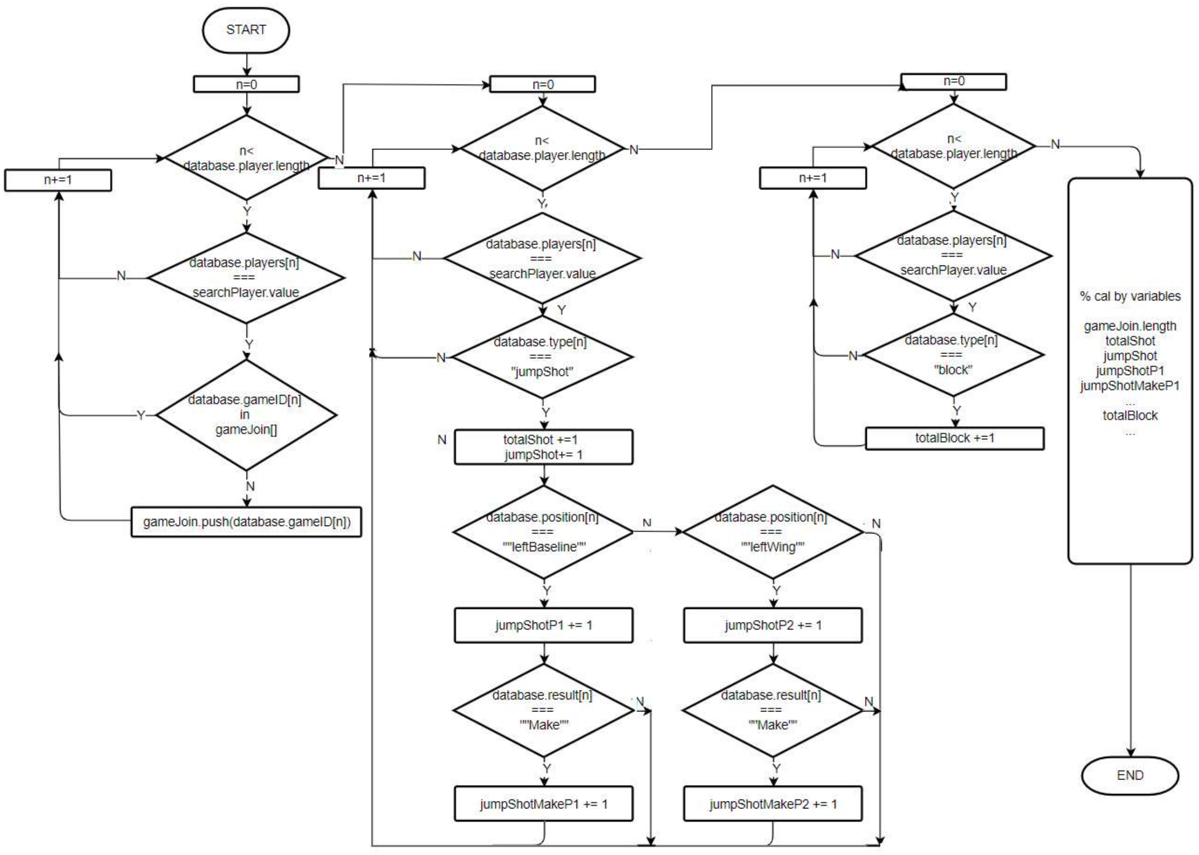
```
// Call bench players onto court
function updatePlayer(insertPlayer1,insertPlayer2,insertPlayer3,insertPlayer4,insertPlayer5) {


  if (insertPlayer1.value !== "") {

      jumpshotPlayer1.value = insertPlayer1.value;
      jumpshotPlayer1.innerHTML = insertPlayer1.value;
      threePointShotPlayer1.value = insertPlayer1.value;
      threePointShotPlayer1.innerHTML = insertPlayer1.value;
      closeShotPlayer1.value = insertPlayer1.value;
      closeShotPlayer1.innerHTML = insertPlayer1.value;
      freeThrowPlayer1.value = insertPlayer1.value;
      freeThrowPlayer1.innerHTML = insertPlayer1.value;
      TOPlayer1.value = insertPlayer1.value;
      TOPlayer1.innerHTML = insertPlayer1.value;
      offensiveRPGPlayer1.value = insertPlayer1.value;
      offensiveRPGPlayer1.innerHTML = insertPlayer1.value;
      defensiveRPGPlayer1.value = insertPlayer1.value;
      defensiveRPGPlayer1.innerHTML = insertPlayer1.value;
      stealPlayer1.value = insertPlayer1.value;
      stealPlayer1.innerHTML = insertPlayer1.value;
      blockPlayer1.value = insertPlayer1.value;
      blockPlayer1.innerHTML = insertPlayer1.value;



  }
```

When someone types sometime in the textAreas (insertPlayer1, insertPlayer2, insertPlayer3, insertPlayer4, insertPlayer5), the function updatePlayer will be activated, and the values and the innerHTML will change, which influence the appearance of the button relating to the players' names.

Flow Chart to Get Team Statistic and Plot Data


START
n=0
n>= gameBook.length
Y
END
N
m=0
n+=1
threePointSequence.push(threePointCount);
threePointSequenceMake.push(threePointCountMake);
jumpshotSequence.push(jumpshotCount);
jumpshotSequenceMake.push(jumpShotCountMake);
freeThrowSequence.push(freeThrowCount);
freeThrowSequenceMake.push(freeThrowCountMake);
closeShotsSequence.push(closeShotCount);
closeShotsSequenceMake.push(closeShotCountMake);
stealSequence.push(stealCount);
blockSequence.push(blockCount);
scoreSequence.push(totalScore);
reset all variables (threePointCount,jumpShotCount ....)
m>= database.gameID.length
Y
m+=1
N
database.gameID[m] === sortedGameBook[n]
Y
database.type[m] === "threePoints"
Y
threePointCount += 1
database.make[m] === "Make"
Y
N
threePointCountMake+=1 totalScore+=3
N
database.type[m] === "jumpShot"
Y
jumpShotCount += 1
database.make[m] === "Make"
Y
N
jumpShotCountMake+=1 totalScore+=2
N
database.type[m] === "clsoeShot"
Y
clsoeShotCount += 1
database.make[m] === "Make"
Y
N
closeShotCountMake+=1 totalScore+=2
N
database.type[m] === "freeThrow"
Y
freeThrowCount += 1
database.make[m] === "Make"
Y
N
freeThrowCountMake+=1 totalScore+=1
N
database.type[m] === "steal"
Y
stealCount += 1
N
database.type[m] === "block"
Y
blockCount += 1

Flow Chart to Get Individual Player's Statistics


START
n=0
n< database.player.length
n+=1
database.players[n] === searchPlayer.value
database.gameID[n] in gameJoin[]
gameJoin.push(database.gameID[n])
database.type[n] === "jumpShot"
totalShot +=1 jumpShot+= 1
database.position[n] === ""leftBaseline""
database.position[n] === ""leftWing""
jumpShotP1 += 1
jumpShotP2 += 1
database.result[n] === ""Make""
jumpShotMakeP1 += 1
jumpShotMakeP2 += 1
database.type[n] === "block"
totalBlock +=1
% cal by variables
gameJoin.length
totalShot
jumpShot
jumpShotP1
jumpShotMakeP1
...
totalBlock
...
END
Y
N

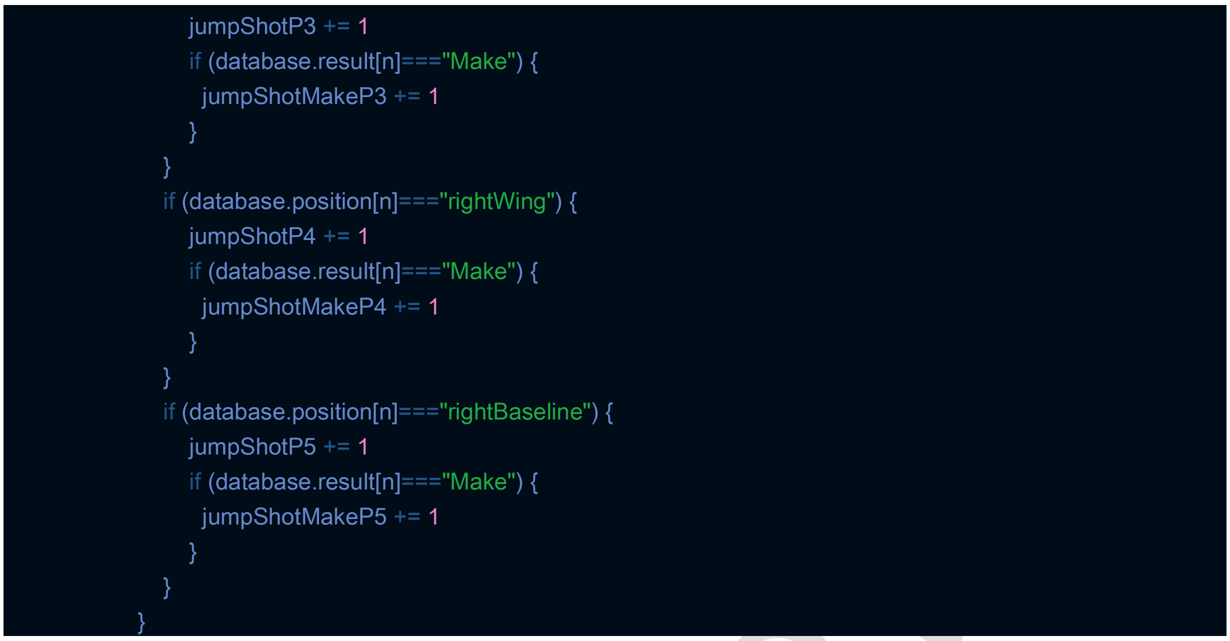
Function: Statistical Calculation (Because there are too many stats that were calculated in the program, for the simplistic explanation, I'll use the example of jumpshot)

P1-5 represents the 5 positions on a basketball court: left baseline, left wing, top, right wing, right baseline

```
totalShot = 0;
   totalMake = 0;
   jumpShot=0;
   jumpShotP1 = 0;
   jumpShotP2 = 0;
   jumpShotP3 = 0;
   jumpShotP4 = 0;
   jumpShotP5 = 0;
   jumpShotMakeP1 = 0
   jumpShotMakeP2 = 0
   jumpShotMakeP3 = 0
   jumpShotMakeP4 = 0
   jumpShotMakeP5 = 0
```

Then the program will use a while loop to run multiple IF conditions to all the data in the database. As long as n is not as big as the length of the data in the object, the program won't stop calculating.

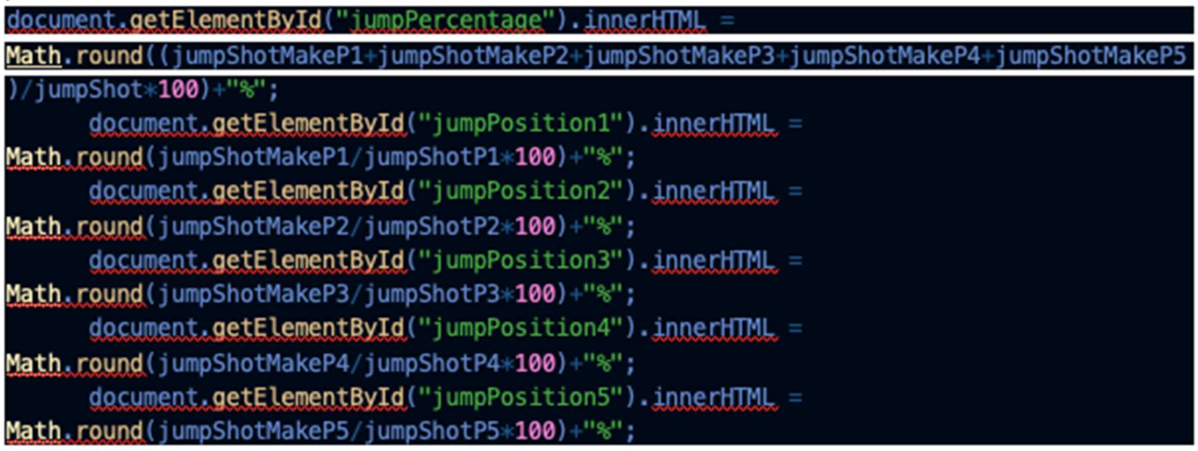
```
n=0
while (n<database.players.length) {


     if (database.players[n] === searchPlayer.value) {
       if (database.type[n] !== "freeThrow") {
          totalShot+=1;

          if (database.type[n]==="jumpshot") {
            jumpShot += 1;
            if (database.position[n]==="leftBaseline") {
              jumpShotP1 += 1
              if (database.result[n]==="Make") {
               jumpShotMakeP1 += 1
              }
            }
            if (database.position[n]==="leftWing") {
              jumpShotP2 += 1
              if (database.result[n]==="Make") {
               jumpShotMakeP2 += 1
              }
            }
            if (database.position[n]==="top") {
```



```
            jumpShotP3 += 1
            if (database.result[n]==="Make") {
              jumpShotMakeP3 += 1
            }
          }
          if (database.position[n]==="rightWing") {
            jumpShotP4 += 1
            if (database.result[n]==="Make") {
              jumpShotMakeP4 += 1
            }
          }
          if (database.position[n]==="rightBaseline") {
            jumpShotP5 += 1
            if (database.result[n]==="Make") {
              jumpShotMakeP5 += 1
            }
          }
        }
```

After finishing the calculation and that each variable has stored the numbers after running the loop, we will now print out the results on the website. getElementByID is used so computer knows where the results of the data should be at.

This was done in repetition for all other stats.



```
document.getElementById("jumpPercentage").innerHTML =
Math.round((jumpShotMakeP1+jumpShotMakeP2+jumpShotMakeP3+jumpShotMakeP4+jumpShotMakeP5
)/jumpShot*100)+"%";
      document.getElementById("jumpPosition1").innerHTML =
Math.round(jumpShotMakeP1/jumpShotP1*100)+"%";
      document.getElementById("jumpPosition2").innerHTML =
Math.round(jumpShotMakeP2/jumpShotP2*100)+"%";
      document.getElementById("jumpPosition3").innerHTML =
Math.round(jumpShotMakeP3/jumpShotP3*100)+"%";
      document.getElementById("jumpPosition4").innerHTML =
Math.round(jumpShotMakeP4/jumpShotP4*100)+"%";
      document.getElementById("jumpPosition5").innerHTML =
Math.round(jumpShotMakeP5/jumpShotP5*100)+"%";
```

This is the code that prints the result, the data number only shows to 2 decimal place using math.round so users may find it easier to look at.

Update chart (There are 8 stats in total, for the purpose of simplistic explanation, I'll use the example of jumpshot throughout the code)

To present the charts, I first had to create a space to put the charts on the website

```
var jumpshotChartPlot = document.getElementById("jumpshotChartPlace")
```

Arrays are necessary so I can plot the data onto the graph. N is set to 0 for the while loop to run.

```
jumpshotSequence = []
jumpshotSequenceMake = []
n = 0
```

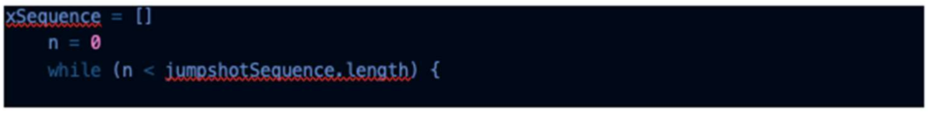
sortedGameBook is an array that stores all the game ID (date) but was already sorted based on time (e.g. [210101, 210105, 210117].

```
while (n < sortedGameBook.length) {
        m = 0

```

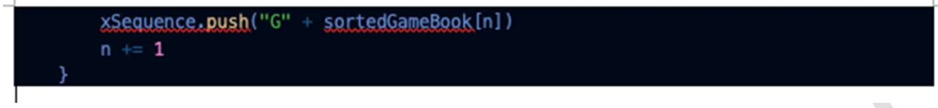
the while loop will store the number of jumpshot taken and the one made. The program will also make note of total score. All of which will be displayed at the end on the website.

```
        jumpshotCount = 0
        jumpShotCountMake = 0
        while (m < database.time.length) {
                if (database.type[m] === "jumpshot") {
           jumpshotCount += 1
           if (database.result[m]==="Make") {
             jumpShotCountMake += 1
             totalScore += 2
           }
                }
        m += 1
}
        jumpshotSequence.push(jumpshotCount);
    jumpshotSequenceMake.push(jumpShotCountMake);


n += 1
```

xSequence is an array of the gameID and eventually it will be used as the x-axis of the plot graph. The values will be the game date (game ID). While loop to make sure all the game ID will be included in the graph.

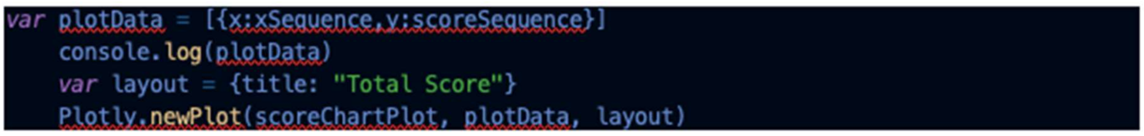


```
xSequence = []
    n = 0
    while (n < jumpshotSequence.length) {
```

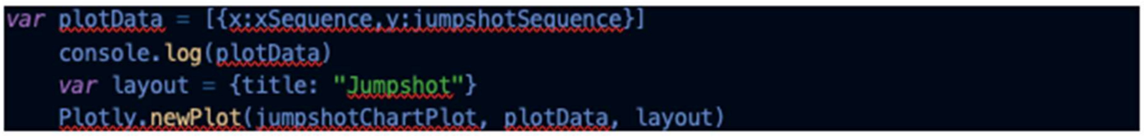


```
        xSequence.push("G" + sortedGameBook[n])
        n += 1
    }
```

The following lines help construct the graphs. This requires us importing additional library, Plotly, which contains functions like newPLot to help make up the graph.



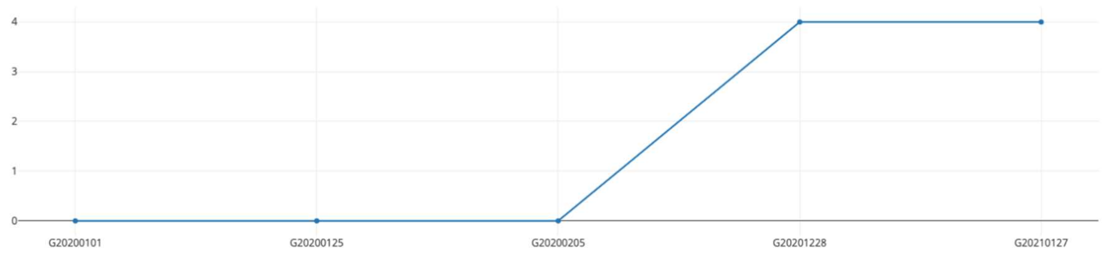
```
var plotData = [{x:xSequence,y:scoreSequence}]
    console.log(plotData)
    var layout = {title: "Total Score"}
    Plotly.newPlot(scoreChartPlot, plotData, layout)
```



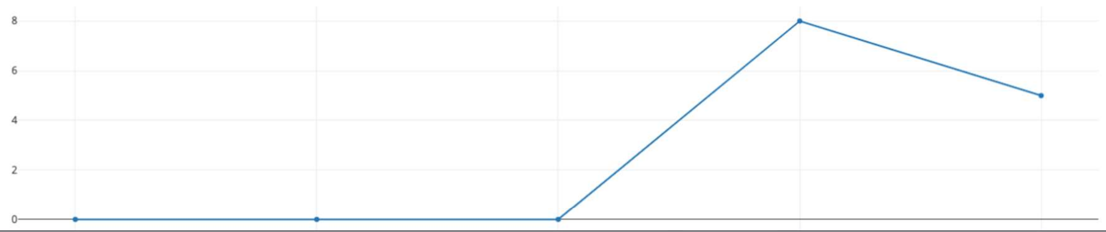
```
var plotData = [{x:xSequence,y:jumpshotSequence}]
    console.log(plotData)
    var layout = {title: "Jumpshot"}
    Plotly.newPlot(jumpshotChartPlot, plotData, layout)
```

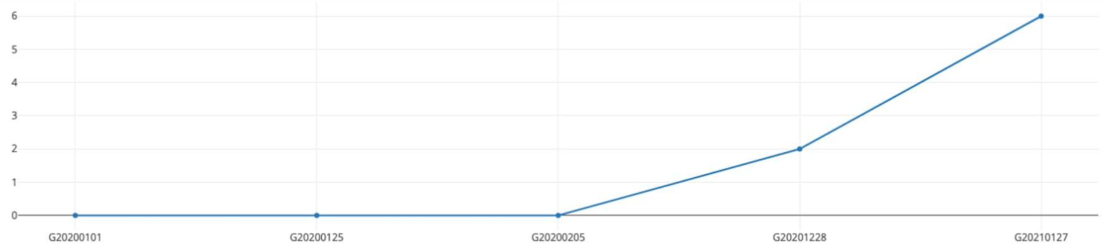
This is the result:

Jumpshot

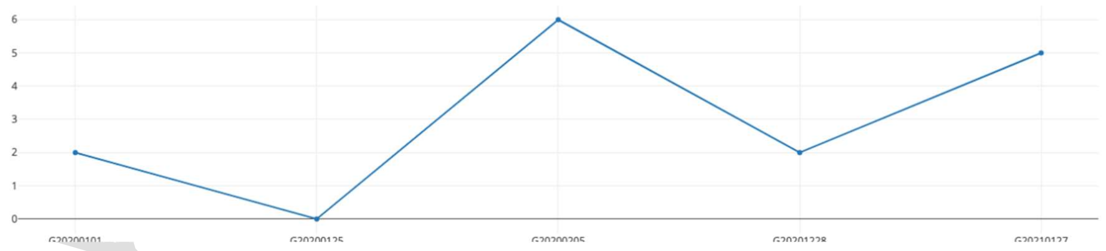


Three Points





Steal



IB Computer Science Criterion E

Word count: 418

Achieving success criteria

	Description of the success criteria	Comment
1	The client can upload, play, pause, fast-forward, backward the mp4 file they have on the program.	• The function uploadVideo() allows the video to be uploaded as the .createObjectURL allows the program to know where to get the file • The function of play(), pause() allow the video to be played and paused • .currentTime tells the time of the video and by adding and subtracting, it can fast-forward the video and back up the video
2	By clicking the event buttons as the video plays, the client could easily build the database collecting each event. Team score will be updated as well	• The buttons can be clicked while the video is watched. • All problematic input could be timely identified and automatically backtracked
3	The client can name and download the database to the client's specified location on the computer	• The client can type any eligible file name to save the fresh input to as a file in a local drive
4	When the client opens the file through any text edit, the client can see the data he has input	• The file shows the object, with the arrays properties, in string. • The data shown is exactly the same as the data the client has input
5	A game file can be uploaded from the client's computer to serve as the database.	• The client can upload a file from their computer
6	The database can be unlimitedly combined as the client upload as many historical files as he wants.	• The client can click to upload more even after a file is uploaded. • The redundant data, if any overlapped in different files, will be just counted once.
7	The combined database can be saved as a new file as specified by the user.	• The download button can be clicked, to allow function downloadFile() to save the combined data as a new file • The new file accommodates all the game data of several component files.

8	The app will screen to dynamically create name buttons for all the players included in combined database .	• Shortcut buttons showing player’s names are displayed after the client uploads a file. The client just clicks any button name to pick.
9	The client can pick a particular player, to display the player’s statistic. The player picture will be also shown provided that the client hasw it in the same folder.	• Conveniently and reliably, the client could click one of the name buttons to get the statistic of the click name. • The client can also type a name to get the player statistics.
10	The client clicks “update chart” to learn team progress, game by game, .	• When the client clicks “update chart”, the graphs are displayed with performance stat on top of each graph

Operating testing plan to validate the success criteria

Testing plan	Description	Time in the video
1	The client can upload, play, pause, fast-forward, backward the mp4 file they have on the program	01:34-02:35
2	By clicking the buttons next to the video, a database that contains all the different statistics is generated	02:35-02:51
3	The client can name and download the file onto the client’s selected location on the computer	02:51-04:44
4	When the client opens the file, the client can see the data he pressed	02:51-04:44
5	The database can be uploaded on the program from the client’s computer	04:44-05:08
6	The uploaded database can be modified as the client presses more buttons	06:35-06:52
7	The modified database can be downloaded onto the computer location selected by the client	05:08-06:52
8	After a database file is uploaded, shortcut buttons are created for the client to search a player’s data	
9	The client can search for data of a particular player, and the program will display the player’s statistic	
10	When the client presses “update chart”, the program will output graphs about the team’s progression on different basketball stats over a season of games	

Client feedback

My client was very happy with the program I have designed for him as it has fulfilled every requirement he has asked for. He loved how interactive and useful this can be. Before the meeting, he had a requirement of me using the statistics design graphs to show the players' progression. After showing my final product to him in the final meeting, he was glad he could see the progression of the players' statistics over long period of games. He thought this tracking tool will help his coaching and managing him team. Moreover, he has also suggested a couple of further updates that could make the program even better.

Potential Improvements

Potential improvement #1: Allowing the graph to be applied to a single player as well

Right now, the statistical graphical display is only available to the team's performance. I should add functions so that user can search for player's name and their graphical display will show up. This can be done by creating more while loop specific to the property (playerName) of the object (database).

Potential improvement #2: more accessibility

One of the problems my client has brought up was that how would this program be accessible to people other than him, like other coaches and players. Because this is a HTML file, the file will only be able to run to the person holding the file. To make it accessible to more people easier, I'll have to change this from a HTML file into a webpage on the website. It's possible for me to do it in a website called GitHub, which is a code hosting platform allowing you to create and host website. I'll create an account there and upload my HTML file to their repository folders. Then my website would be created allowing more people to have access to it.

Potential Improvement #3: Export into a spreadsheet file

Another suggestion my client has made was that instead of making the file downloading in the format of .txt, he would prefer if the data could be downloaded in the format of .xlsx (a spreadsheet). A spreadsheet would be easier to look at than a text file due to many designs, features, and functions of Microsoft Excel. To do so, I'll have to import JHXlSX, which is a dependent JavaScript library to download and create spreadsheet using JSON. The library is fully dynamic and flexible as we can merge and set styling on each cell. With this, I'll be able to export JSON data into an excel file.

Potential Improvement #4: Sharing function

My client was wondering if I can create a function on the website in which he can share the results of the players' statistics to his team directly. Instead of having to screenshot and send the picture to the basketball team group chat on a school messaging app, he wished the program could have a function that when he clicks on a button, an email with the results could be sent to his players. This can be done by installing a nodemailer module on my computer. Afterwards, I can include the module in the code to allow the program to share data in email.