**AQA Computer Science**

**NEA 2020**

**Wesley OSAYANDE**

**Basketball Statistics System**

Logo

Description automatically generated with low confidence

**Contents**

[**Analysis** 3](#_Toc95150588)

[**Background to and identification of the problem** 3](#_Toc95150589)

[**Description of Current System** 4](#_Toc95150590)

[**Identification of prospective users** 7](#_Toc95150591)

[**Specific Requirements of the User and Acceptable Limitations** 7](#_Toc95150592)

[**Numbered Objectives of the Project** 10](#_Toc95150593)

[**Proposed Method of Solution** 12](#_Toc95150594)

[**Design** 13](#_Toc95150595)

[**Overview of the System** 13](#_Toc95150596)

[**Design Data Dictionary** 13](#_Toc95150597)

[**User Interface and Algorithms** 15](#_Toc95150598)

[21](#_Toc95150599)

[**System Security and Integrity** 21](#_Toc95150600)

[**Technical Solution** 22](#_Toc95150601)

[**List of Methods** 22](#_Toc95150602)

[Program.cs 23](#_Toc95150603)

[MainMenu 24](#_Toc95150604)

[CreateAccount 36](#_Toc95150605)

[**Testing** 45](#_Toc95150606)

[**Test 1:** 45](#_Toc95150607)

[**Test 2:** 47](#_Toc95150608)

[**Test 3:** 48](#_Toc95150609)

[**Test 4:** 50](#_Toc95150610)

[**Test 5:** 51](#_Toc95150611)

[**Test 6:** 56](#_Toc95150612)

[**Test 7:** 60](#_Toc95150613)

[**Test 8:** 62](#_Toc95150614)

[**Test 9:** 63](#_Toc95150615)

[**Test 10:** 65](#_Toc95150616)

[**Test 11:** 66](#_Toc95150617)

[**Test 12:** 68](#_Toc95150618)

[**Test 13:** 69](#_Toc95150619)

[**Test 1**4**:** 71](#_Toc95150620)

[**Test 15:** 73](#_Toc95150621)

[**Test 16:** 76](#_Toc95150622)

[**Test 17:** 78](#_Toc95150623)

[**Test 18:** 80](#_Toc95150624)

[**Test 19:** 80](#_Toc95150625)

[**Test 20:** 81](#_Toc95150626)

[**Evaluation** 84](#_Toc95150627)

[**User Feedback** 87](#_Toc95150628)

[**Analysis of User Feedback** 88](#_Toc95150629)

[**Possible Extensions and Improvements** 88](#_Toc95150630)

# **Analysis**

## **Background to and identification of the problem**

I really enjoy playing and watching basketball games, so I decided to create a system which allows the user to easily view the stats such as the points, rebounds and assists per game of individual current NBA players, the details of past fixtures and show the tables from both the eastern and western conferences. The aim of this system is to make these statistics easy to read and understand and navigate through. My program is orientated towards basketball fans, their ages can range from young children to older adults. The system will have to be suited for both audiences.

## **Description of Current System**

One of the most popular places to look for the most up to date basketball statistics is the official [NBA website](https://www.nba.com/stats/). The user is immediately shown the top four stat leaders for points, rebounds, assists, blocks, and fantasy points for the most recent games with the number one leader highlighted in bold with each player Additionally, there is another tab allocated for displaying stats for the entire NBA season. The different categories of stats can be clicked and accessed to show the top 223 players divided into 5 pages accordingly. At the top of the website there are numerous tabs the user can navigate through to find the official leaders, all-time leaders, and hustle leaders.

Graphical user interface, text, application

Description automatically generated

Figure 1

Below is a small section of a summary of the leaders in rebounds for the most recent games which is shown when clicking on the ‘rebounds’ subtitle. As shown along with the number of rebounds (subtly highlighted in blue at the top of the table) a myriad of other unrelated stats such as 3-point attempts and field goal makes for each player. While some important information is displayed such as the respective team (abbreviated), minutes played etc.

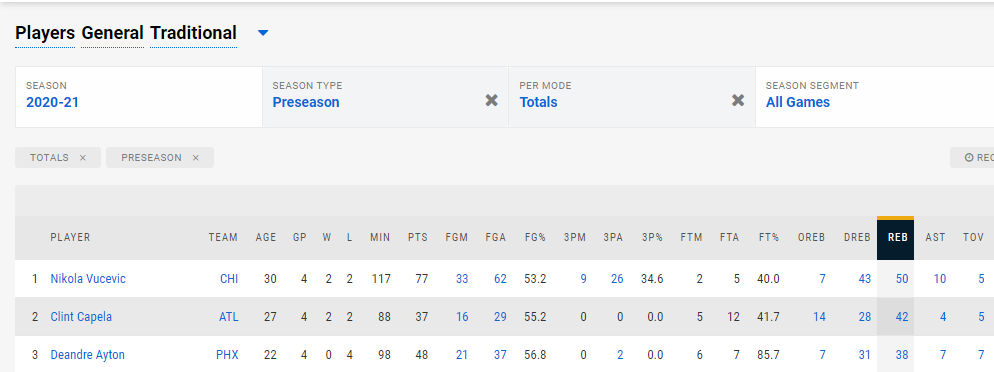
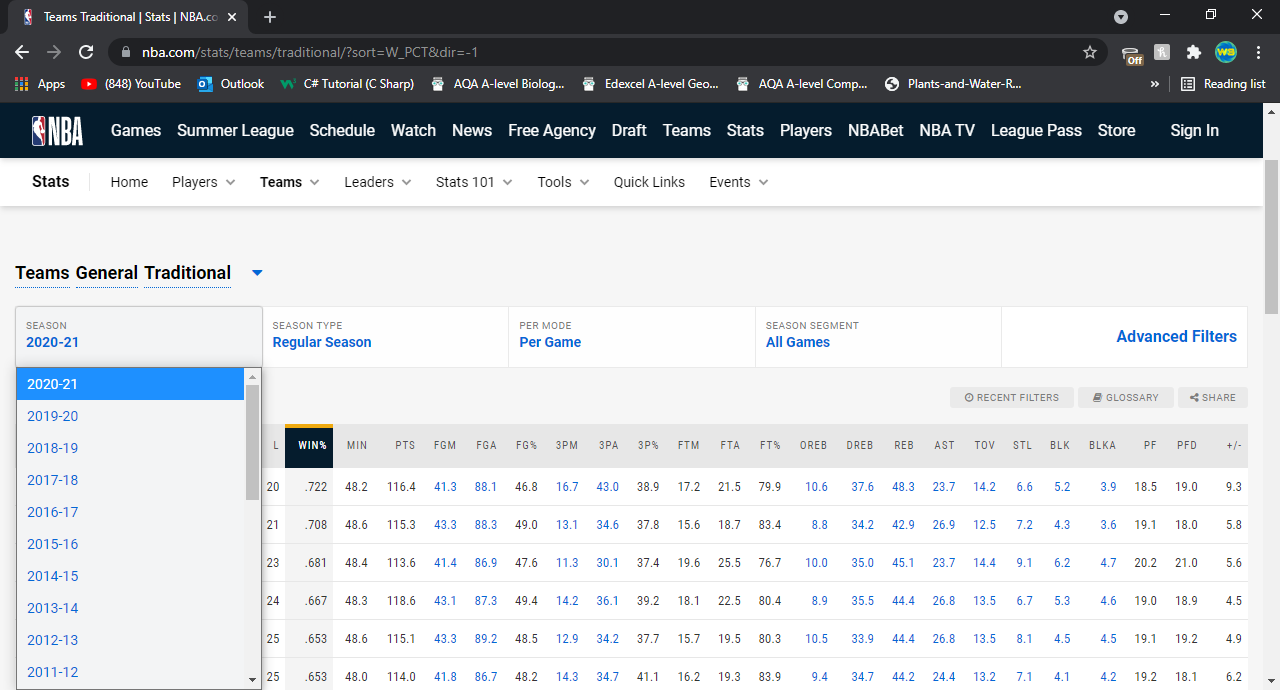


Figure 2

At the top of the table, users can decide to change the season, season type and what segment of each season of what stats are shown via a drop list which is expanded when clicked on displaying a list of the current and past NBA seasons. For whatever season is selected, the stat leaders for the previously chosen stat (Rebounding in this case) will be displayed in a ranked order.

The current system is very cohesive, containing a lot of data to be accessed as its targeted towards a

Figure 3

larger audience while my system isn’t so the overall UI and system will be more compact with possibly not as many features nor as complex.

Graphical user interface, text, application, chat or text message

Description automatically generated

Figure 3

In addition to all the other features mentioned above the website also allows for the creation of an NBA account in which they can use to log in.

Graphical user interface, application

Description automatically generated

Figure 4

When creating an account, the user is presented with an interface displaying all NBA teams in alphabetical order. In this screen, the user can select up to fifteen NBA teams as their favourite.

## **Identification of prospective users**

This statistics system is targeted towards basketball fans of all ages. The software must be simply comprehensible for older audiences, as they tend to be a little less computer literate than they’re younger counterparts. Therefore, the software must be easily accessible and understandable for younger audiences to use while also not being extremely orientated to younger or older audiences as this might discourage the opposite age group.

The system will also be able to display the fairly recent and relevant statistics first and foremost to allow enthusiasts and casual fans alike to get a quick and explanatory glance of the information they need.

## **Specific Requirements of the User and Acceptable Limitations**

My younger brother and a teammate at the Xaverian College basketball team both requested a software capable of displaying basketball stats however, both have differing requirements. My brother requires way to quickly view the most recent stats for his favourite basketball team. He has demanded for several features to be included in the software:

* A login system in which the user will be asked to create an account and select their favourite basketball team, allowing the system to prioritise stats from that team.
* An interface that allows the user to select a stat (ppg, reb etc.)
* An interface which is able to display stats from various players

On the other hand, when asked what features they would like to be present in the program, my teammate Arian demanded for these:

* A way of viewing information about players such as their height and weight
* An interface allowing him to view information about different teams while also being able to have a way to switch to viewing information about players and their stats.

He also desired a feature which would allow the system to simulate a match between two teams and predict the winner and number of points scored in each team. However, this would be very difficult to make and would possibly require machine learning to help predict the outcomes of these games. In addition, implementing such feature would be very time consuming which would only increase the difficulty of the task due to limited time constraints.

**Data Dictionary for Current System**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Data Type** | **Validation** | **Example Data** | **Comment** |
| PlayerName | String | - | Paul George | Name of basketball player |
| TeamName | String | - | Los Angeles Clippers | Unique Identifier of a basketball team name |
| PPG | Integer | - | 13 | The average number of Points Per Game for a player |
| REB | Integer | - | 6 | The average number of Rebounds Per Game for a player |
| Minutes Played | Double | - | 34.2 | The number of minutes played for a player |
| FieldGoal | Double | - | 42 | The ratio of field goals made to field goals attempted (as a percentage) |
| Season | Integer | Must be > 1996 | 2021 | The year a basketball season takes place in |
| Email | String | - | ball@bounce.com | The users email entered in the account creation screen. It’s also used to log in |
| Password | String | Must be at least 8 characters | Password123 | Text used to create/ sign into an account |
| Favourite Team | String | - | Boston Celtics | Can be up to 15 teams. Consists of the users’ favourite teams |
| First\_Name | String | - | Bob | The user’s first name. Not required when creating an account (optional) |
| Last\_Name | String | - | Ball | The user’s last name. Not required when creating an account (optional) |

**Data Volumes**

As my system is database oriented, a large number of data is going to be stored, including each NBA team and its players and each players stats along with it. These will be retrieved from an API and stored in separate tables. Additionally, as users will be allowed to create their own login details a table for this data to go in will also be implemented. The user’s login details are the main source of data which is going to be amended while a lot of statistical data will be added to various tables.

**ER Model for the Existing System**

Stat

Player

Team

Figure 5

As shown in the Entity-Relationship diagram above each team contains multiple players (one to many relationship) and can also have various stats which can be viewed as team averages (one to many relationship). On the other hand, each player can have many stats (one to many relationship).

## **Numbered Objectives of the Project**

(Objectives in red are optional)

1. Retrieve JSON string from an API with relevant player stats
   1. Store items from the JSON string into the player’s table.
   2. This should be done when the form is loaded.
2. Retrieve JSON string from an API with all NBA teams

**2.1** Store items from the JSON string into the team table.

**2.3** This should be done when the form is loaded.

1. When the system is run for the first time, the system must create a database.
   1. The system must check if the database has been created.
   2. The system must create a table for storing login information
   3. The system must create a table for storing player stats
   4. The system must create a table for storing team information
   5. If a database hasn’t been created, the system must create one.
2. Create a login interface.

**4.1** Create a button to be clicked to allow the user to create an account.

* 1. Allow the user to create a username and password.
  2. The system should save the login details in a database.
  3. The system should allow the user to login with their created details.
  4. Create a button to allow the user to enter the main menu if password and username entered are present in the database

1. Allow the user to select their favourite NBA team.

**5.1** Create a drop-down list of all current NBA teams.

**5.2** Store the user’s favourite NBA team on a database.

**5.3** Create a UI displaying the user’s profile and their favourite NBA team.

**5.4** Allow the user to change their favourite NBA team.

**5.5** Allow the user to change their password.

1. Create an opening interface used to display stats
   1. Display information about the user’s favourite team
   2. Create a button to allow the user to load team stats from the team table into a data grid view
   3. Create a button to allow the user to load player stats from the team table into a data grid view
   4. Create a combo box containing all NBA teams for the user to select from
   5. Using a combo box, the program should allow the user to filter/search which team they are looking for.
   6. The filtered team the user wants to search for should be displayed in the data grid view along with all other attributes regarding that team from the team table. (Conference, full name etc.)
   7. Create a button to allow the user to load players stats from the stats table into a data grid view.

**Extension Objectives**

1. Add validation to the username key stored in the **users** table.
   1. Username must be at least 5 characters
2. Add validation to the Pword key stored in the **users** table.
   1. Password cannot be less than 5 characters
   2. Password must contain at least one capital letter
3. Add a label displaying the date and time on the main menu
4. Cover/hide letters of password while being entered.
5. Program should create a table to store player information
   1. The table should be created when the form is first loaded
6. Retrieve JSON string from an API with all NBA teams
   1. Store items from the JSON string into a table used to store player information.
7. Allow the user to search for a specific player
   1. Display the stats for the player matching the search
   2. If the player can’t be found an error message should be displayed
8. If no team is selected from the combo box and the user attempts to search, an error message should be displayed.

## **Proposed Method of Solution**

To create this system, I will be using the C# Programming Language as I think it best suits my project and I am more familiar with this language. The application will be in Windows forms format as it’s the most appropriate for my intended use. Moreover, I will also utilise Microsoft Access to store, access and amend database tables such as log in details which is imperative for this system as large amounts of data are handled. On the other hand, web scraping data from websites such as Basketball References could be a viable alternative to obtain immense amounts of information which would typically be manually stored in a database, from an existing website. This method is a lot more time efficient than manually entering hundreds of entries into tables. While python would arguably be more advisable for web scraping with APIs etc however, due to time constraints, it would be difficult to learn an entirely new language to construct this project.

For the API, I decided to use [‘free NBA’](https://rapidapi.com/theapiguy/api/free-nba). This API is present in the Rapid API website. It was chosen as its one of the most accessible API for retrieving a wide range of NBA data. Additionally, it usually has a service level of 100% which helps ensure my program runs as reliably as possible, guaranteeing the relevant data is ready to be retrieved when needed. While the average latency of the API is quite high at 560 milliseconds (compared to the average latency of an API of 50ms), I doubt this will be much of an issue as the different in real usage is negligible.

# **Design**

## **Overview of the System**

The log in form shown in figure 6 is the first form displayed when the user opens the program and allows the user to enter their username and password in order to gain access to the main menu form. Alternatively, when the **Create Account** button is clicked, the user is directed to a sign-up form allowing for the creation of an account by creating a username, password and are given the option to select their favourite NBA team from a dropdown list. All these details are then stored on the **Users** table. Once the user has created an account they are taken back to the log in form, ready to enter the details from their newly created account or exit the program altogether by selecting the **Exit** button. At the Main Menu, the user will have the option to log out, by clicking this button the user will be logged out and the application will be closed.

Diagram

Description automatically generated

Figure 6

## **Design Data Dictionary**

Player stats database is created in Microsoft Access and consists of the login information for users accessing the system. When the form is loaded the form is created; new entries to the table can be created via the Sign In form when creating an account.

**Table Name –** “Users”

**Primary Key –** “Username”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Length** | **Example Data** | **Comment** |
| Username | VARCHAR | 20 | JaMorant01 | A unique identifier the user uses to either create or login to their account |
| Pword | VARCHAR | 20 | Password145 | An identifier the user enters to login and is created on the ‘create account’ form |
| Team | VARCHAR | 30 | Utah Jazz | Choice prompted when a user creates an account in the ‘create account’ form. |

The database is also used to store data retrieved from the API. When the form is loaded the table is created and the statistics are retrieved via an API.

**Table Name** – “Players”

**Primary Key –** “id”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Length** | **Example Data** | **Comment** |
| id | String | 200 | 1069008 | A unique 7-digit identifier for each player |
| ast | Integer | 200 | 0 | The number of assists |
| blk | Integer | 200 | 1 | The number of blocks |
| dreb | Integer | 200 | 2 | The number of defensive rebounds |
| fg3\_pct | Double | 200 | 0.2 | field goal 3-point percentage |
| fg3a | Integer | 200 | 5 | field goal 3-point attempts |
| fg3m | Integer | 200 | 1 | field goal 3 point makes |
| fg\_pct | Double | 200 | 0.333 | field goal percentage |
| fga | Integer | 200 | 9 | Field goal attempts |
| fgm | Integer | 200 | 3 | field goal makes |
| Fta | Integer | 200 | 0 | free-throw attempts |
| ftm | Integer | 200 | 0 | free-throw makes |
| oreb | Integer | 200 | 0 | The number of offensive rebounds |
| pf | Integer | 200 | 0 | number of personal fouls |
| reb | Integer | 200 | 2 | The number of rebounds |
| stl | Integer | 200 | 0 | The number of steals |
| turnover | Integer | 200 | 2 | The number of turnovers |

**Table Name** – “team”

**Primary Key –** “full\_name”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Length** | **Example Data** | **Comment** |
| Id | String | 200 | 2 | The unique id given to each team, value ranges from 1-30 |
| abbreviation | String | 200 | BOS | Three letter abbreviation of the full name of an NBA team |
| city | String | 200 | Boston | The city of the NBA team |
| conference | String | 200 | East | The conference of a team can either be ‘east’ or ‘west’ each consisting of 15 teams split into 3 divisions |
| division | String | 200 | Atlantic | One of 6 groups in the NBA consisting of 4 other teams in the same area these include: Atlantic, Mountain, Midwest, Pacific, Southwest and Southeast. |
| full\_name | String | 200 | Boston Celtics | The full name of the NBA team including the city name and name of the team |
| name | String | 200 | Celtics | The name of the NBA team |

## **User Interface and Algorithms**

**MainMenu form:**

Graphical user interface, application, Word

Description automatically generated

Buttons: btCreateAccount, btButton1, btExit

(Button1 is the Login button)

GroupBox: gbLogIn

Labels: lbUsername, lbPassword, Date\_Time

When loading the form, the program checks “Player Stats” database already exists, if not its created so the user’s login information can be stored, and the necessary tables are in place for the data retrieved from the API to be stored in when the Main Menu is loaded. The SQL statement below is used to create the respective tables in the player stats table are shown below:

players table:

**"CREATE TABLE Players (id VARCHAR(200), ast VARCHAR(200), blk VARCHAR(200), " +**

**"dreb VARCHAR(200), fg3\_pct VARCHAR(200), fg\_pct VARCHAR(200), fga VARCHAR(200), fgm VARCHAR(200), fta VARCHAR(200), ftm VARCHAR(200), " +**

**"oreb VARCHAR(200), pf VARCHAR(200), reb VARCHAR(200), stl VARCHAR(200), turnover VARCHAR(200), PRIMARY KEY (id))"**

Team table:

**"CREATE TABLE team (id VARCHAR(200), abbreviation VARCHAR(200), " +**

**"city VARCHAR(200), conference VARCHAR(200), division VARCHAR(200), full\_name VARCHAR(200), name VARCHAR(200),PRIMARY KEY (id))";**

Users table:

**"CREATE TABLE Users (Username CHAR(10), Pword VARCHAR(20), full\_name VARCHAR(30), PRIMARY KEY (Username))"**

After the user has entered their username and password in the respective fields and click the login button is clicked the system checks if the username and password combination is present in the Users table. If not, a message box is displayed with one of the following messages: “Wrong Password”, if the password entered is not present on the database or “Username not found” if the username entered is not present. If the login information matches one of the records in the database, upon clicking login, a message box with the message “Log-in successful” is displayed and the items such as labels and textboxes are hidden from the group box while items from the main menu are shown.

If Password entered = Password in Database & Username = Username in Database

Show message (“Log-in successful”)

Displays Main Menu Interface

Else if password entered != Password in Database

Show message (“Wrong Password”)

Else

Show message (“Username not found”)

When the Exit button is clicked the form is closed and the application ends.

When the Create Account button is clicked the form used to create an account is form (SignIn) is opened.

**MainMenu form (when logged in):**

Combo box: cbTeam

Graphical user interface, application

Description automatically generated

GroupBox: gbLogIn

Labels: Date\_Time, lbteam

DataGridView: dataGrid

Buttons:btExit, button2(search), btPlayerStats, btPlayerStats(team stats), btViewAccount.

After the user is successfully logged in, they are greeted with a main menu screen. The datagridview is used to display team or player stats when the respective button is clicked. When player stats button is clicked, all 30 NBA teams are loaded onto the datagridview. SQL for displaying team table:

"SELECT \* FROM team"

**Example of JSON for a team**

{

"data": [

{

"id":14,

"abbreviation":"LAL",

"city":"Los Angeles",

"conference":"West",

"division":"Pacific",

"full\_name":"Los Angeles Lakers",

"name":"Lakers"

},

...

],

"meta": {

"total\_pages": 1,

"current\_page": 1,

"next\_page": null,

"per\_page": 30,

"total\_count": 30

}

}

This data is first taken using an API and stored in the team table in the access database. On the other hand, When the player stats button is clicked, all 100 player stats currently available in the program are displayed. This data is retrieved just like the tam data was. SQL for displaying players table:

"SELECT \* FROM Players"

**Example of JSON for a player’s stats**

{

"data": [

{

"id":29,

"ast":2,

"blk":2,

"dreb":8,

"fg3\_pct":0.25,

"fg3a":4,

"fg3m":1,

"fg\_pct":0.429,

"fga":21,

"fgm":9,

"ft\_pct":0.8,

"fta":5,

"ftm":4,

"game":{

"id":1,

"date":"2018-10-16T00:00:00.000Z",

"home\_team\_id":2,

"home\_team\_score":105,

"season":2018,

"visitor\_team\_id":23,

"visitor\_team\_score":87

},

"min":"36:49",

"oreb":2,

"pf":3,

"player":{

"id":145,

"first\_name":"Joel",

"last\_name":"Embiid",

"position":"F-C",

"team\_id":23

},

"pts":23,

"reb":10,

"stl":1,

"team":{

"id":23,

"abbreviation":"PHI",

"city":"Philadelphia",

"conference":"East",

"division":"Atlantic",

"full\_name":"Philadelphia 76ers",

"name":"76ers"

},

"turnover":5

},

...

],

"meta": {

"total\_pages": 2042,

"current\_page": 1,

"next\_page": 2,

"per\_page": 25,

"total\_count": 51045

}

}

Above the data grid a combo box can be found. It contains all teams for the user to select from once the user has selected the team they want to search for in the database, The search button can be selected. This leads to relevant information about the team selected being displayed on the data grid. SQL for displaying selected team:

"SELECT \* FROM team " + " WHERE full\_name = ('" + ChooseTeam + "')"

However, if no team is selected on the combo box before selecting the search button an error message is displayed prompting the user to select a team.

When the view button, present in the login combo box is selected, a data grid is displayed containing the current user’s login information.

**MainMenu form (View Account and Change Details are clicked):**

Graphical user interface

Description automatically generated

Buttons: btHide, btSaveDetails, btChangeDetails, lbViewPassword, txViewPassword

Datagridview: dgvAccount

The data grid view dvgAccount displays the user’s login details. When the empty cell under each attribute is selected the value is shown and the rest are hidden for security purposes.

The SQL statement used to display such information is below:

"SELECT \* FROM Users WHERE Username = '" + txUsername.Text + "'"

When the ‘change details’ button is clicked, lbViewPassword and txViewPassword are displayed. This button allows the user to change their password. If the password entered: is less than 5 characters, doesn’t contain at least on upper case character or digit an error message is displayed, and the user’s password isn’t updated. If the entered password meets all the criteria mentioned above the password is updated.

When the hide button is clicked all contents in the group box, including the hide button and excluding ‘view account’, are hidden. The SQL statement used to update the password is below:

"UPDATE Users SET Pword = '" + txViewPassword.Text WHERE Username = txUsername.Text"'"

After the user’s password has been updated all contents in the group box, including the hide button and excluding ‘view account’, are hidden.

Pressing the exit button closes the form

**CreateAccount (Account Creation Form)**

In this form, the user can enter a username and password in the corresponding text boxes and select their favourite NBA team from the dropdown list. On the other hand, if any of the three fields (username, password, and favourite team) have been left blank an error message is displayed printing out the message: “Details entered incorrectly” and the user is prompted to enter their details again. Moreover . After the fields have been correctly entered A message box displaying the message “Account created successfully”

Is displayed. The create account button can be clicked and the login information is inserted into the Users table in the Accounts Database.

"INSERT INTO Users VALUES('" + txUsername.Text + "', '" + txPassword.Text + "', '" + FavTeam + "')"

If password entered = “ “ or username entered = “ “ or FavTeam = “ “

Show message (“Details entered incorrectly”)

Else

Insert username && password in USERS table

Show message (“Account Created Successfully”)

## Graphical user interface, diagram Description automatically generated

## **System Security and Integrity**

Due to its type of program, usually sensitive data such as usernames and passwords aren’t an upmost priority in ensuring they are extremely secure. This is because no sensitive information is being input into the program’s database and the program would only be used locally meaning those without access to the user’s personal computer won’t be able to view or amend these details. The only data stored in an individual’s account is their favourite team. However, measures were still taken to ensure the usernames and passwords of the accounts created were not too simplistic. When creating an account, the program requires the username and password to be at least 5 characters. Additionally, to make the user’s password less predictable one of the characters have to be upper case and must contain a number. If these conditions can’t be met an error message is displayed telling the user what needs to be changed.

# **Technical Solution**

**FmCreateAccount**

|  |  |
| --- | --- |
| btCreate\_Click | When clicked the system checks weather the username entered is already present in the database if it is, a message is displayed and the user is prompted to try again if its not already present, the users log-in details are stored in the Users table |
| btReturn\_Click\_1 | When clicked, the SignIn form is closed, and the user is returned to the Log in form (Form1) |
| comboBox1\_SelectedIndexChanged | Allows the user to select from a list of current NBA teams. |
| UpperCase | Used to check if the password contains any Upper case characters |
| NumberCheck | Used to check if the password contains any numbers |

**FmMainMenu**

|  |  |
| --- | --- |
| stats | Receives data from the API (free NBA) from Rapid API containing the header parameters for the host and key. |
| teamInfo | Receives team information data from the API (free NBA) containing the header parameters for the host and key. |
| MainMenu\_Load | When the form loads the Time Date label displays the current date and time.  The system also checks if the Player Stats database is present if it's not, the database is created with the Players, Users and team table. |
| Button2\_Click | Button used to display selected team from the combo box cbteam onto the datagridview ‘dataGrid’ |
| ChooseTeam | Stores the string selected in the combo box cbteam |
| cbTeam\_SelectedIndexChanged | Allows the user to select from a list of current NBA teams. |
| btPlayerStats\_Click | Displays all player stats into the datagridview dataGrid |
| button1\_Click | Button used to display all team information onto the datagridview dataGrid |
| txPassword\_Click | Masks the password |
| btCreateAccount\_Click | Opens the createaccount form to allow the user to create an account |
| button3\_Click | Logs the user into the program |
| btExit\_Click | Closes the program |
| btViewAccount\_Click | Allows the user to view their account details by opening the datagridview dgvAccount |
| NumberCheck | Used to check if the password contains any numbers |
| UpperCase | Used to check if the password contains any Upper case characters |
| btChangeDetails\_Click | Displays a text box to allow the user to enter an updated password and a button to save these details |
| btHide\_Click | Hides all account information |
| btSaveDetails\_Click | Updates the users password to the one entered in the text box ‘txViewPassword’ if criteria is met |

# Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Basketball\_Statistics\_System

{

static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

public const String connString1 = "Provider=Microsoft.ACE.OLEDB.12.0; Data Source = Player\_Stats\_Datbase Database.accdb";

}

}

# MainMenu

using System;

using Json.Net;

using System.Data.OleDb;

using System.IO;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Net.Http;

using Basketball\_Statistics\_System;

using Newtonsoft.Json;

using System.Web.Script.Serialization;

using Newtonsoft.Json.Linq;

namespace Basketball\_Statistics\_System

{

public partial class MainMenu : Form

{

public const String connString1 = "Provider=Microsoft.ACE.OLEDB.12.0; Data Source = Player\_Stats\_Datbase Database.accdb";

private string statsresult;

//string used to store json containing player stats

private string teamresult;

//string used to store json containing team information

public MainMenu()

{

InitializeComponent();

}

public async Task teamInfo()

{

var client = new HttpClient();

var request = new HttpRequestMessage

{

Method = HttpMethod.Get,

RequestUri = new Uri("https://free-nba.p.rapidapi.com/teams?page=0"),

Headers =

{

{ "x-rapidapi-host", "free-nba.p.rapidapi.com" },

//Host site where data is being retriened from

{ "x-rapidapi-key", "d7d5dd3c27msh40f0fac5d835020p184785jsn5664c35fb0ad" },

//Key to access data from the API

},

};

using (var response = await client.SendAsync(request))

{

bool empty = true;

var body = await response.Content.ReadAsStringAsync();

//checks if json is received by the API

while (empty)

{

if (body.Length > 0)

{

empty = false;

break;

}

response.EnsureSuccessStatusCode();

body = await response.Content.ReadAsStringAsync();

}

teamresult = body;

//json stored as a string

//calls the playerinfo class to deserialise json 'teamresult' and stores it in a database

PlayerInfo playerinfo = new PlayerInfo(teamresult);

}

}

public async Task stats()

{

var client = new HttpClient();

var request = new HttpRequestMessage

{

Method = HttpMethod.Get,

RequestUri = new Uri("https://free-nba.p.rapidapi.com/stats?page=0&per\_page=100"),

Headers =

{

//Host site where data is being retriened from

{ "x-rapidapi-host", "free-nba.p.rapidapi.com" },

{ "x-rapidapi-key", "d7d5dd3c27msh40f0fac5d835020p184785jsn5664c35fb0ad" },

//Key to access data from the API

},

};

using (var response = await client.SendAsync(request))

{

bool empty = true;

var body = await response.Content.ReadAsStringAsync();

//checks if json is received by the API

while (empty)

{

if (body.Length > 0)

{

empty = false;

break;

}

response.EnsureSuccessStatusCode();

body = await response.Content.ReadAsStringAsync();

}

//json stored as a string

statsresult = body;

//calls the playerinfo class to deserialise json 'teamresult' and stores it in a database

playerStats playerstats = new playerStats(statsresult);

}

}

public async void MainMenu\_LoadAsync(object sender, EventArgs e)

{

//await stats();

await teamInfo();

await stats();

//displays the current date and time

lbTimeDate.Text = DateTime.Now.ToString();

}

private void button1\_Click(object sender, EventArgs e)

{

OleDbConnection Conn = new OleDbConnection(Program.connString1);

Conn.Open();

OleDbCommand Cmd = new OleDbCommand();

Cmd.Connection = Conn;

string SQL = "";

SQL = "SELECT \* FROM team";

//selects all contents from the team table

Cmd.CommandText = SQL;

OleDbDataAdapter da = new OleDbDataAdapter(Cmd);

DataTable table = new DataTable();

da.Fill(table);

dataGrid.DataSource = table;

if (table.Rows.Count > 0)

{

dataGrid.AutoResizeColumns();

dataGrid.ReadOnly = true;

//prevents the table from being edited by the user

dataGrid.Show();

//Displays records from the table onto the data grid view

}

else

{

MessageBox.Show("Nothing found.");

//Error message displayed if no data is found in the database to be displayed

}

Conn.Close();

}

private void MainMenu\_Load(object sender, EventArgs e)

{

// creates the 'Players' database if it does not already exist

if (File.Exists("Player\_Stats\_Datbase Database.accdb") == false)

{

ADOX.Catalog cat = new ADOX.Catalog();

cat.Create(Program.connString1);

OleDbConnection Conn = new OleDbConnection(Program.connString1);

Conn.Open();

OleDbCommand Cmd = new OleDbCommand();

Cmd.Connection = Conn;

Cmd.CommandText = "CREATE TABLE Players (id VARCHAR(200), ast VARCHAR(200), blk VARCHAR(200), " +

"dreb VARCHAR(200), fg3\_pct VARCHAR(200), fg\_pct VARCHAR(200), fga VARCHAR(200), fgm VARCHAR(200), fta VARCHAR(200), ftm VARCHAR(200), " +

"oreb VARCHAR(200), pf VARCHAR(200), reb VARCHAR(200), stl VARCHAR(200), turnover VARCHAR(200), PRIMARY KEY (id))";

//Creates the players table in the database

Cmd.ExecuteNonQuery();

Cmd.CommandText = "CREATE TABLE team (id VARCHAR(200), abbreviation VARCHAR(200), " +

"city VARCHAR(200), conference VARCHAR(200), division VARCHAR(200), full\_name VARCHAR(200), name VARCHAR(200),PRIMARY KEY (id))";

//Creates the team table in the database

Cmd.ExecuteNonQuery();

Cmd.CommandText = "CREATE TABLE Users (Username CHAR(20), Pword VARCHAR(20), full\_name VARCHAR(30), PRIMARY KEY (Username))";

//Creates the Users table in the database

Cmd.ExecuteNonQuery();

Conn.Close();

}

lbteam.Hide();

cbTeam.Hide();

button2.Hide();

dataGrid.Hide();

button1.Hide();

btPlayerStats.Hide();

btViewAccount.Hide();

btHide.Hide();

dgvAccount.Hide();

btChangeDetails.Hide();

btSaveDetails.Hide();

//Main menu buttons hidden until user has sucessfully logged in

txViewUser.Hide();

txViewPassword.Hide();

lbViewPassword.Hide();

lbViewUser.Hide();

//Buttons and labels which appear when View Account has been clicked are hidden also

stats();

//retrieves player stats data from API free NBA

teamInfo();

//retrieves team information from API free NBA

//displays the current date and time

lbTimeDate.Text = DateTime.Now.ToString();

}

public static string ChooseTeam = "";

//When an NBA team is selected from the combo box this string is changed to the chosen

//team ready to be entered into the Users table

private void button2\_Click(object sender, EventArgs e)

{

if (ChooseTeam == "")

//If no favourite team is selected

{

MessageBox.Show("Please Select a Team");

//Error message displayed if no team is selected from the drop down list

}

OleDbConnection Conn = new OleDbConnection(Program.connString1);

Conn.Open();

OleDbCommand Cmd = new OleDbCommand();

Cmd.Connection = Conn;

string SQL = "";

SQL = "SELECT \* FROM team " + " WHERE full\_name = ('" + ChooseTeam + "')";

//Searches for data only from the chosen team selected from the drop down list

Cmd.CommandText = SQL;

OleDbDataAdapter da = new OleDbDataAdapter(Cmd);

DataTable table = new DataTable();

da.Fill(table);

dataGrid.DataSource = table;

if (table.Rows.Count > 0)

{

dataGrid.AutoResizeColumns();

dataGrid.ReadOnly = true;

dataGrid.Show();

//Displays records from the table onto the data grid view

}

else

{

MessageBox.Show("Nothing found.");

//Error message displayed if no data is found in the database to be displayed

}

Conn.Close();

//Close connection

}

private void cbTeam\_SelectedIndexChanged(object sender, EventArgs e)

{

ChooseTeam = cbTeam.Text;

//Contains a list of all 30 NBA teams from the user to pick from

//when team is selected and search button is clicked, choice is saved along the users login details

switch (cbTeam.SelectedIndex)

{

case 0:

ChooseTeam = "Atlanta Hawks";

break;

case 1:

ChooseTeam = "Boston Celtics";

break;

case 2:

ChooseTeam = "Brooklyn Nets";

break;

case 3:

ChooseTeam = "Charlotte Hornets";

break;

case 4:

ChooseTeam = "Chicago Bulls";

break;

case 5:

ChooseTeam = "Cleveland Cavaliers";

break;

case 6:

ChooseTeam = "Dallas Mavericks";

break;

case 7:

ChooseTeam = "Denver Nuggets";

break;

case 8:

ChooseTeam = "Detroit Pistons";

break;

case 9:

ChooseTeam = "Golden State Warriors";

break;

case 10:

ChooseTeam = "Houston Rockets";

break;

case 11:

ChooseTeam = "Indiana Pacers";

break;

case 12:

ChooseTeam = "Los Angeles Clippers";

break;

case 13:

ChooseTeam = "Los Angeles Lakers";

break;

case 14:

ChooseTeam = "Memphis Grizzlies";

break;

case 15:

ChooseTeam = "Miami Heat";

break;

case 16:

ChooseTeam = "Milwaukee Bucks";

break;

case 17:

ChooseTeam = "Minnesota Timberwolves";

break;

case 18:

ChooseTeam = "New Orleans Pelicans";

break;

case 19:

ChooseTeam = "New York Knicks";

break;

case 20:

ChooseTeam = "Oklahoma City Thunder";

break;

case 21:

ChooseTeam = "Orlando Magic";

break;

case 22:

ChooseTeam = "Philadelphia 76ers";

break;

case 23:

ChooseTeam = "Phoenix Suns";

break;

case 24:

ChooseTeam = "Portland Trail Blazers";

break;

case 25:

ChooseTeam = "Sacramento Kings";

break;

case 26:

ChooseTeam = "San Antonio Spurs";

break;

case 27:

ChooseTeam = "Toronto Raptors";

break;

case 28:

ChooseTeam = "Utah Jazz";

break;

case 29:

ChooseTeam = "Washington Wizards";

break;

}

}

private void btPlayerStats\_Click(object sender, EventArgs e)

{

OleDbConnection Conn = new OleDbConnection(Program.connString1);

Conn.Open();

OleDbCommand Cmd = new OleDbCommand();

Cmd.Connection = Conn;

string SQL = "";

SQL = "SELECT \* FROM Players";

Cmd.CommandText = SQL;

OleDbDataAdapter da = new OleDbDataAdapter(Cmd);

DataTable table = new DataTable();

da.Fill(table);

dataGrid.DataSource = table;

if (table.Rows.Count > 0)

{

dataGrid.AutoResizeColumns();

dataGrid.ReadOnly = true;

dataGrid.Show();

}

else

{

MessageBox.Show("Nothing found.");

}

Conn.Close();

}

private void txPassword\_Click(object sender, EventArgs e)

{

txPassword.Text = "";

txPassword.ForeColor = Color.Black;

txPassword.PasswordChar = '●'; // Password masking for added security

}

private void btCreateAccount\_Click(object sender, EventArgs e)

{

CreateAccount fmSignIn = new CreateAccount();

fmSignIn.ShowDialog();

//Opens the CreateAccount form to allow the user to create an account

}

private void button3\_Click(object sender, EventArgs e)

{

OleDbConnection Conn = new OleDbConnection(Program.connString1);

// Uses constant connString to create a database

Conn.Open();

// Used to open a connection with the database

OleDbCommand Cmd = new OleDbCommand();

//A database command object Cmd is created

Cmd.Connection = Conn;

Cmd.CommandText = "SELECT \* FROM Users WHERE Username ='" + txUsername.Text + "'";

//Checks the users table if the username entered is present

OleDbDataReader reader = Cmd.ExecuteReader();

if (reader.HasRows)

{

reader.Read();

//The records are read if they are found in the Users table

if (txPassword.Text == reader["Pword"].ToString() || txUsername.Text == reader["Username"].ToString())

//If the password entered in the text box is present in the database

{

MessageBox.Show("Log-in successful!");

// Message box displayed if the password and username entered is present in the datase

button3.Hide();

txPassword.Hide();

txUsername.Hide();

lbPassword.Hide();

lbUsername.Hide();

logo.Hide();

btCreateAccount.Hide();

//Hides contents of groupbox when the user has logged in successfully

lbteam.Show();

cbTeam.Show();

button2.Show();

dataGrid.Show();

button1.Show();

btPlayerStats.Show();

btViewAccount.Show();

//Displays Mainmenu when logged in successfully

}

else

{

MessageBox.Show("Wrong Password");

//error message displayed if the password is not present in the database

}

}

else

{

MessageBox.Show("User name not found.");

//error message displayed if the username is not present in the database

}

Conn.Close();

//Close connection

}

private void btExit\_Click(object sender, EventArgs e)

{

Close();

}

private void btViewAccount\_Click(object sender, EventArgs e)

{

dgvAccount.Show();

btChangeDetails.Show();

btHide.Show();

OleDbConnection Conn = new OleDbConnection(Program.connString1);

Conn.Open();

OleDbCommand Cmd = new OleDbCommand();

Cmd.Connection = Conn;

string SQL = "";

SQL = "SELECT \* FROM Users WHERE Username = '" + txUsername.Text + "'";

//searches for the users login details

Cmd.CommandText = SQL;

OleDbDataAdapter da = new OleDbDataAdapter(Cmd);

DataTable table = new DataTable();

da.Fill(table);

dgvAccount.DataSource = table;

if (table.Rows.Count > 0)

{

dgvAccount.AutoResizeColumns();

dgvAccount.ReadOnly = true;

//prevents the table from being edited by the user

dgvAccount.Show();

//Displays records from the table onto the data grid view

}

else

{

MessageBox.Show("Nothing found.");

//Error message displayed if no data is found in the database to be displayed

}

Conn.Close();

//shows buttons and labels to view the users account information

}

//Used to check if the password contains any numbers

private int NumberCheck(string password)

{

int number = 0;

foreach (char ch in password)

{

if (char.IsDigit(ch))

{

number++;

}

}

return number;

}

//Used to check if the password contains any Upper case characters

private int UpperCase(string password)

{

int uppercheck = 0;

foreach (char upp in password)

{

if (char.IsUpper(upp))

{

uppercheck++;

}

}

return uppercheck;

}

//Minimum value for the password

const int minvalue = 5;

private void btChangeDetails\_Click(object sender, EventArgs e)

{

txViewPassword.Show();

lbPassword.Show();

btSaveDetails.Show();

//shows a textbox to allow the user to change their password

}

private void btHide\_Click(object sender, EventArgs e)

{

txViewUser.Hide();

txViewPassword.Hide();

lbPassword.Hide();

lbViewUser.Hide();

btSaveDetails.Hide();

dgvAccount.Hide();

btChangeDetails.Hide();

btHide.Hide();

}

private void btSaveDetails\_Click(object sender, EventArgs e)

{

if (txViewPassword.Text == "")

{

MessageBox.Show("Details entered incorrectly");

//Error message displayed if the password text box is left blank

}

else if (txViewPassword.TextLength < minvalue)

//If the password entered is less than 5 characters an eror message is displayed

{

MessageBox.Show("Password must be atleast 5 characters");

}

else if (UpperCase(txViewPassword.Text) < 1 || NumberCheck(txViewPassword.Text) < 1)

{

//If the password entered doesnt contain an upper case character or a digit

MessageBox.Show("Password must contain at least one number and upper case character");

}

else

{

OleDbConnection Conn = new OleDbConnection(Program.connString1);

Conn.Open();

OleDbCommand Cmd = new OleDbCommand();

Cmd.Connection = Conn;

String SQL = "UPDATE Users " +

" SET Pword = '" + txViewPassword.Text +

"' WHERE Username = '" + txUsername.Text + "'";

//Updates only the account with the username loggen in with.

Cmd.CommandText = SQL;

Cmd.ExecuteNonQuery();

Conn.Close();

MessageBox.Show("Password has been changed!");

//message box shown if password has successfully been changed

txViewUser.Hide();

txViewPassword.Hide();

lbPassword.Hide();

lbViewUser.Hide();

btSaveDetails.Hide();

dgvAccount.Hide();

btChangeDetails.Hide();

btHide.Hide();

//Buttons hidden after password has been successfully changed

}

}

}

}

# CreateAccount

using System;

using System.IO;

using System.Data.OleDb;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Basketball\_Statistics\_System

{

public partial class CreateAccount : Form

{

public CreateAccount()

{

InitializeComponent();

}

private void btReturn\_Click\_1(object sender, EventArgs e)

{

Close();

}

//Used to check if the password contains any Upper case characters

private int UpperCase(string password)

{

int uppercheck = 0;

foreach(char upp in password)

{

if (char.IsUpper(upp))

{

uppercheck++;

}

}

return uppercheck;

}

//Used to check if the password contains any numbers

private int NumberCheck(string password)

{

int number = 0;

foreach (char ch in password)

{

if (char.IsDigit(ch))

{

number++;

}

}

return number;

}

//Minimum value for the username and password

const int minvalue = 5;

private void btCreate\_Click\_1(object sender, EventArgs e)

{

OleDbConnection Conn = new OleDbConnection(Program.connString1);

Conn.Open();

//connection is opened

OleDbCommand Cmd = new OleDbCommand();

Cmd.Connection = Conn;

OleDbDataReader reader = null; // This is OleDb Reader

OleDbCommand cmd = new OleDbCommand("SELECT \* FROM Users WHERE Username ='" + txUsername.Text + "'", Conn);

reader = cmd.ExecuteReader();

if (reader.HasRows) //if a record is found

{

reader.Read();

if (txUsername.Text == reader["Username"].ToString())

{

Console.WriteLine("Username already exists");

}

}

if (txUsername.Text == "" || txPassword.Text == "" || FavTeam == "")

{

MessageBox.Show("Details entered incorrectly");

//Error message displayed if the username, password of combo box are left blank

}

else if (txPassword.TextLength < minvalue || txUsername.TextLength < minvalue)

//If the username or password entered is less than 5 characters an eror message is displated

{

MessageBox.Show("Password and Username must be atleast 5 characters");

}

else if (UpperCase(txPassword.Text) < 1 || NumberCheck(txPassword.Text) < 1)

{

//If the password entered doesnt contain an upper case character or a digit

MessageBox.Show("Password must contain at least one number and upper case character");

}

else

//If all conditions have been met the entered details are stored in the Users table

{

Cmd.Connection = Conn;

Cmd.CommandText = "INSERT INTO Users VALUES('" + txUsername.Text + "', '" + txPassword.Text + "', '" + FavTeam + "')";

// Text entered in the username textbox and password textbox are

//inserted in the users table in addition to the favourite team chosen

Cmd.ExecuteNonQuery();

Conn.Close();

//Connection is closed

MessageBox.Show("Account Created Successfully");

//Message box displayed if the accaount details were entered correctly

Close();

//Once the account has been created the form is closed and the user is returned to

//the main menu to log in with their details

}

}

public static string FavTeam = "";

//When an NBA team is selected from the combo box this string is changed to the chosen

//team ready to be entered into the Users table

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

FavTeam = comboBox1.Text;

//Contains a list of all 30 NBA teams from the user to pick from

switch (comboBox1.SelectedIndex)

{

case 0:

FavTeam = "Atlanta Hawks";

break;

case 1:

FavTeam = "Boston Celtics";

break;

case 2:

FavTeam = "Brooklyn Nets";

break;

case 3:

FavTeam = "Charlotte Hornets";

break;

case 4:

FavTeam = "Chicago Bulls";

break;

case 5:

FavTeam = "Cleveland Cavaliers";

break;

case 6:

FavTeam = "Dallas Mavericks";

break;

case 7:

FavTeam = "Denver Nuggets";

break;

case 8:

FavTeam = "Detroit Pistons";

break;

case 9:

FavTeam = "Golden State Warriors";

break;

case 10:

FavTeam = "Houston Rockets";

break;

case 11:

FavTeam = "Indiana Pacers";

break;

case 12:

FavTeam = "Los Angeles Clippers";

break;

case 13:

FavTeam = "Los Angeles Lakers";

break;

case 14:

FavTeam = "Memphis Grizzlies";

break;

case 15:

FavTeam = "Miami Heat";

break;

case 16:

FavTeam = "Milwaukee Bucks";

break;

case 17:

FavTeam = "Minnesota Timberwolves";

break;

case 18:

FavTeam = "New Orleans Pelicans";

break;

case 19:

FavTeam = "New York Knicks";

break;

case 20:

FavTeam = "Oklahoma City Thunder";

break;

case 21:

FavTeam = "Orlando Magic";

break;

case 22:

FavTeam = "Philadelphia 76ers";

break;

case 23:

FavTeam = "Phoenix Suns";

break;

case 24:

FavTeam = "Portland Trail Blazers";

break;

case 25:

FavTeam = "Sacramento Kings";

break;

case 26:

FavTeam = "San Antonio Spurs";

break;

case 27:

FavTeam = "Toronto Raptors";

break;

case 28:

FavTeam = "Utah Jazz";

break;

case 29:

FavTeam = "Washington Wizards";

break;

}

}

}

}

**Player Stats (class)**

using Newtonsoft.Json;

using Newtonsoft.Json.Linq;

using System;

using System.Data;

using System.Collections.Generic;

using System.Data.OleDb;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.ComponentModel;

namespace Basketball\_Statistics\_System

{

public class playerStats

{

///public class playerStats

public playerStats(string json)

{

JObject jObject = JObject.Parse(json);

JToken jUser = jObject["data"];

var info = JObject.Parse(json);

JArray items = (JArray)info["data"];

//Used to represent the json array data containing all the other values

int length = items.Count;

for (int i = 0; i <= length - 1; i++)

//loops through the json array to represent data from each team

{

id = (string)jUser[i]["id"];

ast = (string)jUser[i]["ast"];

blk = (string)jUser[i]["blk"];

dreb = (string)jUser[i]["dreb"];

fg3\_pct = (string)jUser[i]["fg3\_pct"];

fg3a = (string)jUser[i]["fg3a"];

fg3m = (string)jUser[i]["fg3m"];

fg\_pct = (string)jUser[i]["fg\_pct"];

fga = (string)jUser[i]["fga"];

fgm = (string)jUser[i]["fgm"];

fta = (string)jUser[i]["fta"];

ftm = (string)jUser[i]["ftm"];

//game = (IList<string>)jUser[i]["game"];

oreb = (string)jUser[i]["oreb"];

pf = (string)jUser[i]["pf"];

//player = (IList<string>)jUser[i]["player"];

reb = (string)jUser[i]["reb"];

stl = (string)jUser[i]["stl"];

//team = (IList<string>)jUser[i]["team"];

turnover = (string)jUser[i]["turnover"];

OleDbConnection Conn1 = new OleDbConnection(Program.connString1);

Conn1.Open();

OleDbCommand Cmd1 = new OleDbCommand();

Cmd1.Connection = Conn1;

Cmd1.CommandText = "INSERT INTO Players VALUES('" + id + "','" + ast + "','" + blk + "', '" + dreb + "','"

+ fg3\_pct + "','" + fg\_pct + "','" + fga + "','" + fgm + "','" + fta + "','" + ftm + "','" + oreb + "','" + pf + "','"

+ reb + "','" + stl + "','" + turnover + "')";

OleDbDataReader reader = Cmd1.ExecuteReader();

Conn1.Close();

}

}

// Unique ID number for each player

[JsonProperty("id")]

public string id { get; set; }

//The number of assists

[JsonProperty("ast")]

public string ast { get; set; }

//The number of blocks

[JsonProperty("blk")]

public string blk { get; set; }

//The number of defensive rebounds

[JsonProperty("dreb")]

public string dreb { get; set; }

//field goal 3 point percentage

[JsonProperty("fg3\_pct")]

public string fg3\_pct { get; set; }

//field goal 3 point attempts

[JsonProperty("fg3a")]

public string fg3a { get; set; }

//field goal 3 point makes

[JsonProperty("fg3m")]

public string fg3m { get; set; }

//field goal percentage

[JsonProperty("fg\_pct")]

public string fg\_pct { get; set; }

//Field goal attempts

[JsonProperty("fga")]

public string fga { get; set; }

//field goal makes

[JsonProperty("fgm")]

public string fgm { get; set; }

//free-throw attempts

[JsonProperty("fta")]

public string fta { get; set; }

//free-throw makes

[JsonProperty("ftm")]

public string ftm { get; set; }

//The number of offensive rebounds

[JsonProperty("oreb")]

public string oreb { get; set; }

//number of personal fouls

[JsonProperty("pf")]

public string pf { get; set; }

//The number of rebounds

[JsonProperty("reb")]

public string reb { get; set; }

//The number of steals

[JsonProperty("stl")]

public string stl { get; set; }

//The number of turnovers

[JsonProperty("turnover")]

public string turnover { get; set; }

}

}

**Player Info (class)**

using Newtonsoft.Json.Linq;

using System;

using System.Collections.Generic;

using System.Data.OleDb;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Basketball\_Statistics\_System

{

public class PlayerInfo

{

public PlayerInfo(string json)

{

//string json = @"{""id"":""abbreviation"":""city"":""conference"":""division"":""full\_name"":""name"":}";

JObject jObject = JObject.Parse(json);

//Used to represent a json object

JToken jUser = jObject["data"];

var info = JObject.Parse(json);

JArray items = (JArray)info["data"];

//Used to represent the json array data containing all the other values

int length = items.Count;

for (int i = 0; i < length; i++)

//loops through the json array to represent data from each team

{

var item = (JObject)items[i];

id = (string)jUser[i]["id"];

abbreviation = (string)jUser[i]["abbreviation"];

city = (string)jUser[i]["city"];

conference = (string)jUser[i]["conference"];

division = (string)jUser[i]["division"];

full\_name = (string)jUser[i]["full\_name"];

name = (string)jUser[i]["name"];

OleDbConnection Conn1 = new OleDbConnection(Program.connString1);

Conn1.Open();

OleDbCommand Cmd1 = new OleDbCommand();

Cmd1.Connection = Conn1;

//Inserting the Json values into the team table in the playerStats database

Cmd1.CommandText = "INSERT INTO team VALUES('" + id + "', '" + abbreviation + "','"

+ city + "','" + conference + "','" + division + "','" + full\_name + "','" + name + "')";

Cmd1.ExecuteReader();

Conn1.Close();

}

}

public string id { get; set; }

public string abbreviation { get; set; }

public string city { get; set; }

public string conference { get; set; }

public string division { get; set; }

public string full\_name { get; set; }

public string name { get; set; }

}

}

## 

# **Testing**

|  |
| --- |
| **Test 1:** |
| **Description of the test:** When the Main Menu form is loaded for the first time, the system must create the player stats database.  **Test Data and Reason:** The system should check if a database exists when its run if the database isn’t present, one should be created.  **Expected Outcome:** The creation of the player stats database when the program is first run.  **Actual Outcome:** The first time the program is run, the accounts database is created and all tables are loaded properly |
| **Evidence:** |

|  |
| --- |
| **Test 2:** |
| **Description of the test:** When the Main Menu form is loaded for the first time, the team information should be retrieved from the API and stored in the team table.  **Test Data and Reason:** Program is opened  **Expected Outcome:** All 30 NBA teams appearing on the team table when the program is first run  **Actual Outcome:** As expected |
| **Evidence:** |

|  |
| --- |
| **Test 3:** |
| **Description of the test:** When the Main Menu form is loaded for the first time, the player stats data should be retrieved from the API and stored in the team table.  **Test Data and Reason:** none  **Expected Outcome:** 100 players appear on the players’ table  **Actual Outcome:** As expected |
| **Evidence:** |

|  |
| --- |
| **Test 4:** |
| **Description of the test:** Create a button to be clicked to allow the user to create an account.  **Test Data and Reason:** Clicking the Create Account button  **Expected Outcome:** The CreateAccount form opens  **Actual Outcome:** The CreateAccount form opens |
| **Evidence:** |

|  |
| --- |
| **Test 5:** |
| **Description of the test:** The create account form should allow the user to create an account which is stored in the users table. If data is input incorrectly an error message should be displayed  **Test Data and Reason:**  Typical data:  ‘bob098’  ‘ball123’  ‘Boston Celtics’  Erroneous data:  ‘bob’  ‘ball123’  ‘Atlanta Hawks’  Erroneous data:  ‘bob123’  ‘ball’  ‘Atlanta Hawks’  **Expected Outcome:** The CreateAccount form opens  **Actual Outcome:** The CreateAccount form opens |
| **Evidence:**  **Test details 1:**    **When the login details are entered correctly, they are saved in the users table.**    **Test details 2:**    **Error message displayed as username is less than three characters; data isn’t put in the Users table and the user can attempt to create an account again.**    **Test details 3:**    **Error message displayed as password is less than three characters; data isn’t put in the Users table and the user can attempt to create an account again.**    **Test details 4:**    **Error message displayed as Username textbox was left blank; data isn’t put in the Users table and the user can attempt to create an account again.**    **Test details 5:**    **Error message displayed as Password textbox was left blank; data isn’t put in the Users table and the user can attempt to create an account again.**    **Test details 6:**    **Error message displayed as favourite team combo box was left blank; data isn’t put in the Users table and the user can attempt to create an account again.** |

|  |
| --- |
| **Test 6:** |
| **Description of the test:** When creating an account, the username and password can’t be less than 5 characters  **Test Data and Reason:**  Typical data:  WoofWoof  Dog123  ‘Brooklyn Nets’  Erroneous data:  Woof  Dog1  ‘Brooklyn Nets’  Erroneous data:  MeowMeow  Cat1  ‘Brooklyn Nets’  Erroneous data:  Meow  Cat123  ‘Brooklyn Nets’  **Expected Outcome:** When the search button is clicked the team selected from the combo box will be displayed in the data grid view.  **Actual Outcome:** As expected |
| **Evidence:**  Typical Data:      Entry added to the table  Erroneous data:      Entry isn’t added to the table  Erroneous data:      Entry isn’t added to the table  Erroneous data:      Entry isn’t added to the table |

|  |
| --- |
| **Test 7:** |
| **Description of the test:** When creating an account, the password must contain an Upper-case letter  **Test Data and Reason:**  Typical data:  ‘BarkBark’  Canine123  ‘Atlanta Hawks’  Erroneous Data:  ‘Animal1’  canine123  ‘Atlanta Hawks’  **Expected Outcome:** An error message is displayed if no upper-case letters are present in the password.  **Actual Outcome:** As expected |
| **Evidence:**  Typical Data:    Entry added to the database  Erroneous Data: |

|  |
| --- |
| **Test 8:** |
| **Description of the test:** When clicking the return to menu button the user should betaken back to the main menu form  **Test Data and Reason:** Clicking the return to menu button.  **Expected Outcome:** The CreateAccount form is closed, and the suer is returned to the main menu  **Actual Outcome:** As expected |
| **Evidence:**      **Form was closed** |

|  |
| --- |
| **Test 9:** |
| **Description of the test:** When creating an account, the password must contain at least one digit  **Test Data and Reason:**  Typical data:  ‘Lamelo1’  Canine23  ‘Atlanta Hawks’  Erroneous Data:  ‘Green2’  Canine  ‘Atlanta Hawks’  **Expected Outcome:** An error message is displayed if no digits are present in the password.  **Actual Outcome:** As expected |
| **Evidence:**  Typical Data:    Entry added to the database    Erroneous Data:    Entry isn’t added to the table |

|  |
| --- |
| **Test 10:** |
| **Description of the test:** When clicking the Exit button, the program should close  **Test Data and Reason:** Clicking the exit button  **Expected Outcome:** The program closes  **Actual Outcome:** As expected |
| **Evidence:**      **Program closed afterwards** |

|  |
| --- |
| **Test 11:** |
| **Description of the test:** When clicking the login button, the program should check if the details entered in the username and password textboxes match any records in the users table, if they do the user is successfully logged in, if they don’t then an error message is displayed.  **Test Data and Reason:**  ‘bob098’  ‘ball123’  Erroneous data:  ‘bob098’  ‘football135’  Erroneous data:  ‘Ronaldo567’  ‘ball123’  **Expected Outcome:** User is logged in when correct details are entered and a respective error message is displayed if details are incorrect.  **Actual Outcome:** As expected |
| **Evidence:**  **Current account stored in the Users table**    **Test details 1:**    **Detail matched the record on the table, so the user was successfully logged in**  **Test details 2:**    **Program found the username matches one of the records in the table however the password didn’t, resulting in an error message.**  **Test details 3:**    **Program found the password matches one of the records in the table however the username didn’t, resulting in an error message.** |

|  |
| --- |
| **Test 12:** |
| **Description of the test:** If the user attempts to click the search button, next to the combo box containing NBA teams, without first selecting a team to search for an error message is displayed.  **Test Data and Reason:**  None  **Expected Outcome:** Clicking search before selecting a team from the combo box results in an error message being displayed  **Actual Outcome:** As expected |
| **Evidence:** |

|  |
| --- |
| **Test 13:** |
| **Description of the test:** The combo box aside the ‘select team’ label should contain all NBA teams; if the user selects one and clicks the search button the program should display the team information for the selected team on the data grid view. If no record of the team is found on the team table an error message is displayed.  **Test Data and Reason:**  Teams searched for:   1. Charlotte Hornets 2. Utah Jazz 3. Golden State Warriors   **Expected Outcome:** When the search button is clicked the team selected from the combo box will be displayed in the data grid view.  **Actual Outcome:** As expected |
| **Evidence:**    **After the search button was selected:**    **After the search button was selected:**    **After the search button was selected:** |

|  |
| --- |
| **Test 1**4**:** |
| **Description of the test:** A button should display team stats on a datagridview when clicked and should be able to be clicked again  **Test Data and Reason:** Button is clicked consecutively to show it works after on attempt  **Expected Outcome:** The button displays team stats on data grid after being pressed  **Actual Outcome:** The button only displays team stats after being clicked for the first time; an error occurs if pressed more than once.  **After the fix:** The button displays team stats on data grid after being pressed even after being pressed more than once. |
| **Evidence (after first click):**    Team data is correctly displayed  **After second click:**    **Fix(highlighted line moved to public async Task stats()):**    **After fix (can be clicked continuously):** |

|  |
| --- |
| **Test 15:** |
| **Description of the test:** A button should display player stats on a datagridview when clicked and should be able to be clicked again  **Test Data and Reason:** Button is clicked consecutively to show it works after on attempt  **Expected Outcome:** The button displays player stats on data grid after being pressed  **Actual Outcome:** The button only displays team stats after being clicked for the first time; an error occurs if pressed more than once.  **After the fix:** The button displays player stats on data grid after being pressed even after being pressed more than once. |
| **Evidence (after first click):**    Player data is correctly displayed  **After second click:**    **Fix(highlighted line moved to public async Task stats()):**    **After fix (can be clicked continuously):** |

|  |
| --- |
| **Test 16:** |
| **Description of the test:** The program should hide the login groupbox after the user has logged in  **Test Data and Reason:** None  **Expected Outcome:** Login group box hidden when user is logged in  **Actual Outcome:** As expected |
| **Evidence:**      **If incorrect details are entered:**    **The user is given another attempt to enter the correct details** |

|  |
| --- |
| **Test 17:** |
| **Description of the test:** The view account button should display the login details of the account signed in on. When the cell on the data grid is clicked on the relevant details should be shown and hidden when another attribute is selected  **Test Data and Reason:** None  **Expected Outcome:** Program displays user’s account.  **Actual Outcome:** As expected |
| **Evidence:** |

|  |
| --- |
| **Test 18:** |
| **Description of the test:** User details and the hide button should be hidden once the hide button is clicked  **Test Data and Reason:** None  **Expected Outcome:** User details are hidden  **Actual Outcome:** As expected |
| **Evidence:** |
| **Test 19:** |
| **Description of the test:** Clicking change details should present the user with a textbox to allow them to change their password  **Test Data and Reason:** None  **Expected Outcome:** Textbox and label appear  **Actual Outcome:** As expected |
| **Evidence:** |
| **Test 20:** |
| **Description of the test:** The user can create a new password; an error message is displayed if the password is less than 5 characters, doesn’t contain an upper case character or doesn’t contain at least one digit.  **Test Data and Reason:**  Typical data:  “Lebron145”  Erroneous Data:  “ ”  Erroneous Data:  “curry123”  Erroneous Data:  “Curry”  **Expected Outcome:** The user’s password is updated if the password is entered correctly  **Actual Outcome:** As expected |
| **Evidence:**      **Password updated**  Erroneous Data:      **Details aren’t updated**  Erroneous Data:    **Details aren’t updated** |

# **Evaluation**

1. Retrieve JSON string from an API with relevant player stats
   1. Store items from the JSON string into the player’s table.
   2. This should be done when the form is loaded.

This objective was successfully met. This was achieved by calling the API, free NBA, and converting/deserialising JSON objects into C# objects then storing them in the players table. A model class for a player’s stats was created indicating the various attributes (e.g., reb and id) of a player. This is done when the form is loaded.

1. Retrieve JSON string from an API with all NBA teams

**2.1** Store items from the JSON string into the team table.

**2.3** This should be done when the form is loaded.

This objective was successfully met. This was achieved by calling the API, free NBA, and converting/deserialising JSON objects into C# objects then storing them in the team table. A model class for a team’s stats was created indicating the various attributes (e.g., conference, division). This is done when the form is loaded.

**3**. When the system is run for the first time, the system must create a database.

* 1. The system must check if the database has been created.
  2. The system must create a table for storing login information
  3. The system must create a table for storing player stats
  4. The system must create a table for storing team information
  5. If a database hasn’t been created, the system must create one.

This objective was successfully met. This was achieved through the MainMenu load method. An if statement checks is the database is already present if not, through the use of SQL commands, it created with the players, team and users tables.

1. Create a login interface.
   1. Create a button to be clicked to allow the user to create an account.
   2. Allow the user to create a username and password.
   3. The system should save the login details in a database.
   4. The system should allow the user to login with their created details.

**4.5** Create a button to allow the user to enter the main menu if password and username entered are present in the database

This objective was successfully met. The MainMenu form acts as a login interface as well as a main menu. It contains textboxes to allow the user to input their login information. It also has a ‘create account’ button. When clicked it opens the CreateAccount form which allows the user to create an account and saves these details on the users table.

1. Allow the user to select their favourite NBA team.

**5.1** Create a drop-down list of all current NBA teams.

**5.2** Store the user’s favourite NBA team on a database.

**5.3** Create a UI displaying the user’s profile and their favourite NBA team.

**5.4** Allow the user to change their favourite NBA team.

**5.5** Allow the user to change their password.

Most objectives were successfully met. The combo box combobox1 holds all current NBA teams. Upon reaching the CreateAccount form selecting a favourite NBA team is one of the required fields when creating an account. When the account is created, the users favourite team is stored alongside their other details in the Users table. Unfortunately, objective 5.4 wasn’t met. The user is able to change their password in the MainMenu form by clicking ‘View Account’ and ‘Change Details’. By doing this the user is able to update the password currently stored in the users table.

1. Create an opening interface used to display stats
   1. Display information about the user’s favourite team
   2. Create a button to allow the user to load team stats from the team table into a data grid view
   3. Create a combo box containing all NBA teams for the user to select from
   4. Using a combo box, the program should allow the user to filter/search which team they are looking for.
   5. The filtered team the user wants to search for should be displayed in the data grid view along with all other attributes regarding that team from the team table. (Conference, full name etc.)
   6. Create a button to allow the user to load players stats from the stats table into a data grid view.

The current system is unable to display information about the users favourite team. The other objectives on the other hand were successfully met. When clicked, button 1 displays team data onto the data grid view dataGrid. Conversely, when btPlayerStats is selected player data is displayed. The combo box btTeam contains a list of all current NBA teams when one is selected, and the search button is clicked only data from that team will be displayed on dataGrid.

**Extension Objectives**

1. Add validation to the username key stored in the **users** table.
   1. Username must be at least 5 characters

Objectives were successfully met. Validation was added to the CreateAccount form so that if the user attempts to create a username with less than 5 characters an error message is displayed.

1. Add validation to the Pword key stored in the **users** table.
   1. Password cannot be less than 5 characters
   2. Password must contain at least one capital letter

Objectives were successfully met. Validation was added to the CreateAccount form so that if the user attempts to create a password with less than 5 characters, no digits or no upper case characters an error message is displayed.

1. Add a label displaying the date and time on the main menu

Objective was successfully met. Date/time labels are present in the main menu.

1. Cover/hide letters of password while being entered.

Objective was successfully met. Date/time labels are present in the main menu.

1. Program should create a table to store player information
   1. The table should be created when the form is first loaded

Objective not met. There is no table for storing player information just stats.

1. Allow the user to search for a specific player
   1. Display the stats for the player matching the search
   2. If the player can’t be found an error message should be displayed

Objective not met. There is no way to search for a player

1. If no team is selected from the combo box and the user attempts to search, an error message should be displayed.

Objective was successfully met. If the search button (button2) if clicked before selecting a team the user is prompted to select a team.

## **User Feedback**

Anonymous user

**Was the program easy to use?**

The program was easy to use for the most part. Tasks such as viewing team information was very simple to do. However, as players in the player stats aren’t represented with names but rather IDs it was difficult to gather which player was who.

**Was the UI easy to navigate through?**

Yes, travelling through the main menu was very easy.

**Where the stats easy to understand?**

The players stats were understandable but figuring out which player is which is difficult. Team stats were easily accessible and readable and with the search functionality I could search for a specific team.

## **Analysis of User Feedback**

Overall, the user was pleased with the program. The comments on the player stats were understandable as I had trouble saving player names from the JSON string this meant he had to rely on IDs and had to refer to the free NBA site to figure out whose ID is whose. Originally the user requested a simulation feature on the program, however with my current knowledge on the C# programming language the request was far beyond my current skill set.

## **Possible Extensions and Improvements**

To improve the system, my program requires a fleshed out depiction of player stats and could possibly gain the functionality of displaying the data of players related to a specific team when a team is selected. With these improvements, the possibility to implement a searching algorithms for both players and teams can become more likely. Moreover, I believe more validation and exception handling could be added when creating an account as the current systems in place are very simplistic.

Currently, the program only retrieves 100 player entries from the API when there are actually 1169297 total entries available. To increase the number of entries pulled at once a loop could be added to loo through the 11693 pages.