Final year project first deliverable – Nathan simcock

Basketball Management Simulator

De Montfort University

Computer Games Programming

Contents

[Literature Review 1](#_Toc61372762)

[Functional Requirements 4](#_Toc61372763)

[Base Components Functional Requirements 4](#_Toc61372764)

[Functional Requirements to Win the Game 5](#_Toc61372765)

[Use Cases 5](#_Toc61372766)

[System Design Documentation 7](#_Toc61372767)

[User Interface Designs 7](#_Toc61372768)

[Entity Relationship Diagram 11](#_Toc61372769)

[System Architecture 12](#_Toc61372770)

[Test Plan 14](#_Toc61372771)

[Test Cases 15](#_Toc61372772)

[Implementation Report 18](#_Toc61372773)

[References 19](#_Toc61372774)

# Literature Review

It is believed (Bristow, 1977, p.1) the first video game was developed in the “very early 1960’s” where Steve Russell created Space War. In 1972, the first consumer video game Odyssey was released by Magnavox (Kent, 2010), followed by the well-known, but very similar version from Atari called ‘Pong’. These games laid the groundworks for the development of video games to be sold to a consumer market, rather than simply for coin operated machines.

While it can be argued that Pong itself is a sports game as it reflects Tennis, Taito and Sega were the companies which blazed the trail for the Sports Video Games industry we have today. In 1973 Taito released several Sports games including Davis Cup for doubles tennis, Soccer, and Pro Hockey. In 1974 Taito introduced Basketball, an attempt at a “better concept of a team oriented video game” (Nachman, 2011).

In early Video Game history, a big problem for the industry was how you can reflect a sport with more than 20 athletes performing at once, in games which rarely have more than two players. That was before the now Video Game Giants Electronic Arts released Earl Weaver Baseball, a game with AI shaped by Baseball Hall of Fame member Earl Weaver. According to Nachman (2011) the game is “still considered to be one of the best baseball games ever created.” The game was not only innovative in its use of Artificial Intelligence, but it was also the first sports video game to allow the player to simulate an entire season without having to watch play-by-play. This is something which I intend to use in my own game, where the user simulates matches, rather than playing them.

According to Mutzu (2016) the concept of a Sports Simulation Game is “to recreate the game as close to reality as possible”. This genre of games has grown a huge following, especially in recent times as the realism continues to grow. This can be put down to the fact that the game provides an experience which is so rare, that of managing a sports team, and one of the best. As a sports fan myself, there are countless times you question the line-up choice of a Football manager, or the signing of a player in the National Basketball Association. A sports simulation game gives you the chance to make these choices yourself.

In fact, Sports Simulation Games are often used as ways of educating those who are trying to enter the Sports industry. Surdam (2009, p.138-149) wrote how they created their own Sports Simulation card game to teach Sports Economics students the basics of running a sports team, including “the dynamics of revenue sharing, television contracts, salary caps, and other concepts.”

To further the point of how useful Sports Video Games can be for learning, one of Basketball’s all-time greats, LeBron James (2017, quoted by McMillen, 2017), said that he uses the video game series NBA 2K to test his teams line-up for when an injured player returns, stating "It’s the most realistic basketball game you could ever play,".

In the games industry, it is rare that a game is released which is unlike any game which has come before. For example, Fornite, Apex Legends and PlayerUnknown’s Battlegrounds (PUBG) were three battle royale games which were all released in a two-year time frame, with Fortnite and PUBG just months apart. While all of these games have their own unique features, such as Fortnite’s building or PUBG’s more realistic approach compared to the other two’s more cartoon like graphics, they all share some core functions. All three follow the same win condition of be the last player remaining, all have similar looting systems, and all games begin with the player’s dropping out of some form of air vehicle.

From this knowledge, I knew that to gain an understanding of how to make the best possible Basketball Management Simulator possible, I must study those that already exist in the genre.

The creator of online Basketball Sim ‘Basketball Gm’ (So you want to write a sports simulation game, no date) wrote that the first thing required in a sports simulation game is “A database to store all the data (teams, players, game results, etc)”.

While my game lacks the licensing of a real Basketball league, meaning the database cannot be ‘accurate’, this helped me to realise that the quality of the database would be crucial in the quality of the overall game. The reasoning behind this is that some of the core features I want to include in my game, such as trading players, rely on the use of a database.

Also, in a way the database will define the difficulty of the game, which must be fair if there are to be no difficulty levels at first. This is because the game engine I am going to make will consider each player’s ratings when deciding match winners. Therefore, one team cannot be unfairly balanced with much higher rated players, as this team would be near impossible to beat, at the same time, the user’s team must be somewhere in the middle of the league in terms of rating, otherwise there would be no point in using the trade feature to try and build a better squad.

Although having a detailed database makes part of the core of a good sports simulation, this must be paired with an accurate match engine. Football Manager 2021, arguably the biggest sports simulation game with 93,775 concurrent players at its peak (stat from Steamcharts.com), is constantly praised for its match engine and how the players make decisions and react without any user input, with a Gamesradar reviewer (James, 2020) stating “match engine is the best it’s ever been.” While my game will not provide a view of the players in game, the match engine will be needed to take into account stats and ratings, player happiness and line-ups to generate a realistic outcome to every match.

For example, the effect of an unhappy player cannot be underestimated on a team, with their lack of commitment causing a severe drop in output. The game engine must find a balance where a great team with unhappy players is able to lose games.

The match engine must also find a balance between offense and defence within a match. The method I would like to use to find the result of a game is to compare the quality of each team’s offense against their opponent’s defence to generate a range for the team’s possible score. A generator will then pick a number within each team’s range, with the higher number resulting in the winning team.

When determining how this system is created, I must find the correct balance between offence and defence. If unbalanced, it would be easy for the user to play their strongest offensive team and simply outscore their opponents. However, in the study (Williams, 2019) of the well-known sports phrase ‘Offense wins games, defence wins championships’, it was found that Basketball statistics support the idea that a great defensive team can actually outperform a great offense.

Another thing to consider is the various types of offense and defence, with scoring coming mostly in two forms, close range dunks and layups, and longer-range jump shots. A team which is great at defending dunks and layups may not be great at defending jump shots, therefore the game must reflect the different types of scoring. This gives the player more ways of winning if they look more in depth at team stats. For instance, the player might choose to exploit a team with poor jump shooting defence by changing their starting line-up to include more skilled shooters, rather than their best overall team. This type of thinking must be rewarded by the game rather than overlooked, so is something I will look to include in the match engine.

(next part on Micro transactions)

# Functional Requirements

When creating a computer system, the functional requirements are the core components of that system. As reflected in the name, they are required for the system to be functional and meet the needs of the client. In a video game, functional requirements would be the components which are necessary for the game to be played and completed. For example, a 3D shooter would require a camera for the player to be able to move through the scene, however, a second camera for two alternative views would be non-functional as it is an added extra and is not necessary for the game to work.

To recognise the functional requirements of my game, I must first identify what the purpose or goal of the game is. Basketball Management Simulator is a menu-based sports simulation game. The overview of the game is that the player becomes the new general manager of a Basketball team and must oversee this team. The goal of the game is to try and guide the team to winning the league championship, first by winning enough games to reach the knockout-based playoffs, and then winning the playoffs.

As part of running the Basketball team, the player must perform various tasks while maintaining player and owner relationships to improve the team. Such tasks include making trades with other AI controlled teams to acquire new players and editing the starting line-up of the team to try and create the best outcome.

The functional requirements can be split into two parts, the base components necessary to physically play the game, and the components needed to allow the player to reach the end goal of winning.

## Base Components Functional Requirements

* The game must allow the player to start a new game.
* The game must allow the player to quit the game.
* The game must allow the player to switch between scenes when necessary, such as loading the main game scene, or loading the match scene.
* The game must allow the player to progress through the game, via the use of the continue button.
* The game must allow user input via the mouse to navigate through the game.

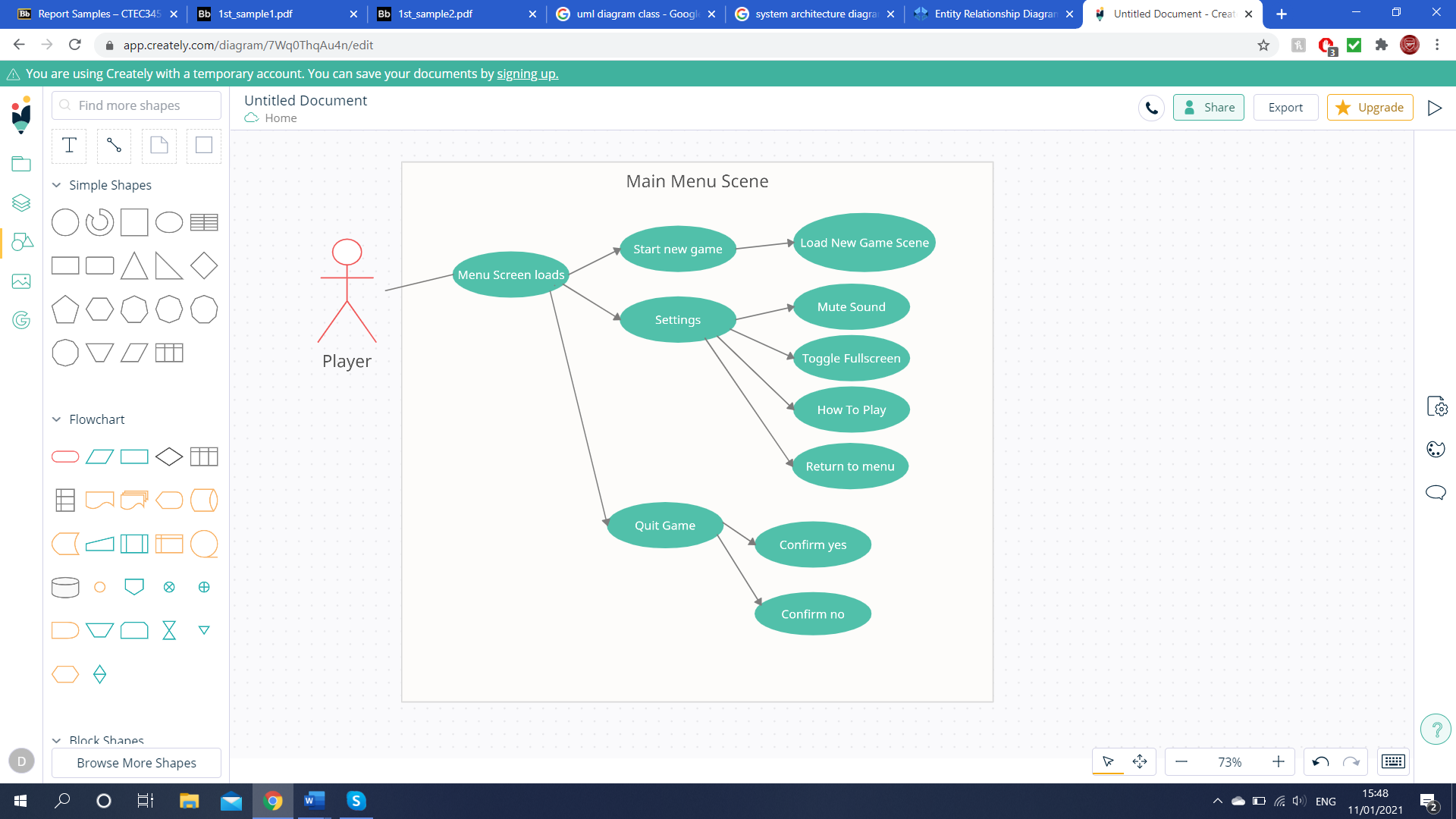
## Functional Requirements to Win the Game

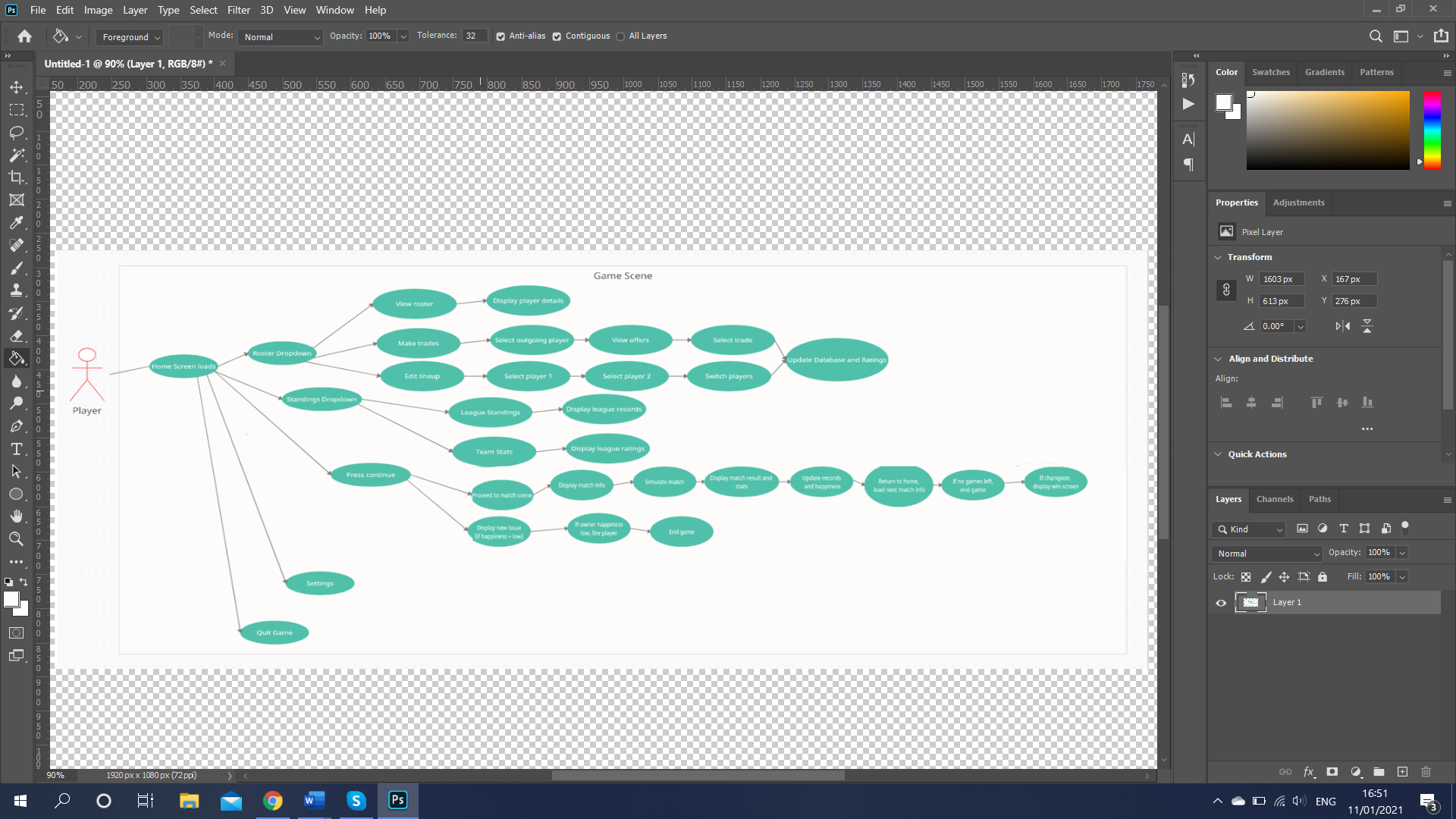
* After each match, the game must allow progression by loading the next opponent.
* The game must allow the player to make trades, by loading offers when the player searches for a trade.
* The game must end after all matches have been played.
* If the player wins the championship, the winning screen must be displayed.
* If the owner happiness drops too low, the losing screen must be displayed.
* After each database change such as trade or line-up change, the game must update the database to reflect these details.
* The match script must take the game database into account when deciding the result of matches.

## Use Cases

A use case creates a visual representation of the expected result of the system, given input from the user. In the case of a video game, it would show the interaction of the player and the result of what would happen upon their interaction. The purpose of a use case diagram is to show how the system can meet the requirements set out. For example, the main menu use case will show how the user is able to start a new game, and the actions required to do so.

1. Main Menu scene



1. New Game scene

Settings and quit game already detailed in main menu screen

(Creately.com)

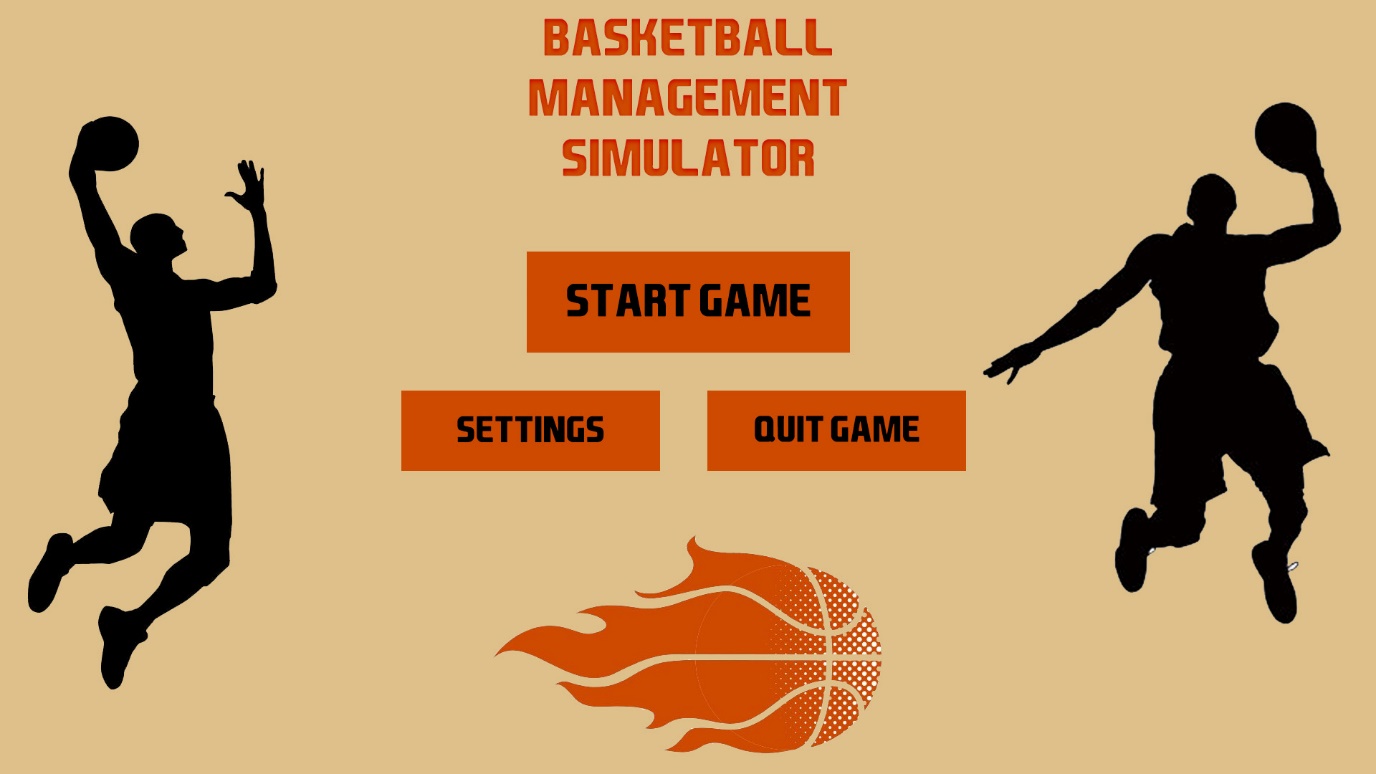
# System Design Documentation

## User Interface Designs

Due to the nature of the game being heavily UI focused, it is crucial the UI is well designed and clear, as this will really dictate the quality of the game. The general idea in mind for each screen is to display the most important information in the clearest way possible and make it easy for the user to play.

I intend to keep the theme of colours, text and designs similar across the game to create consistency and help provide player immersion.

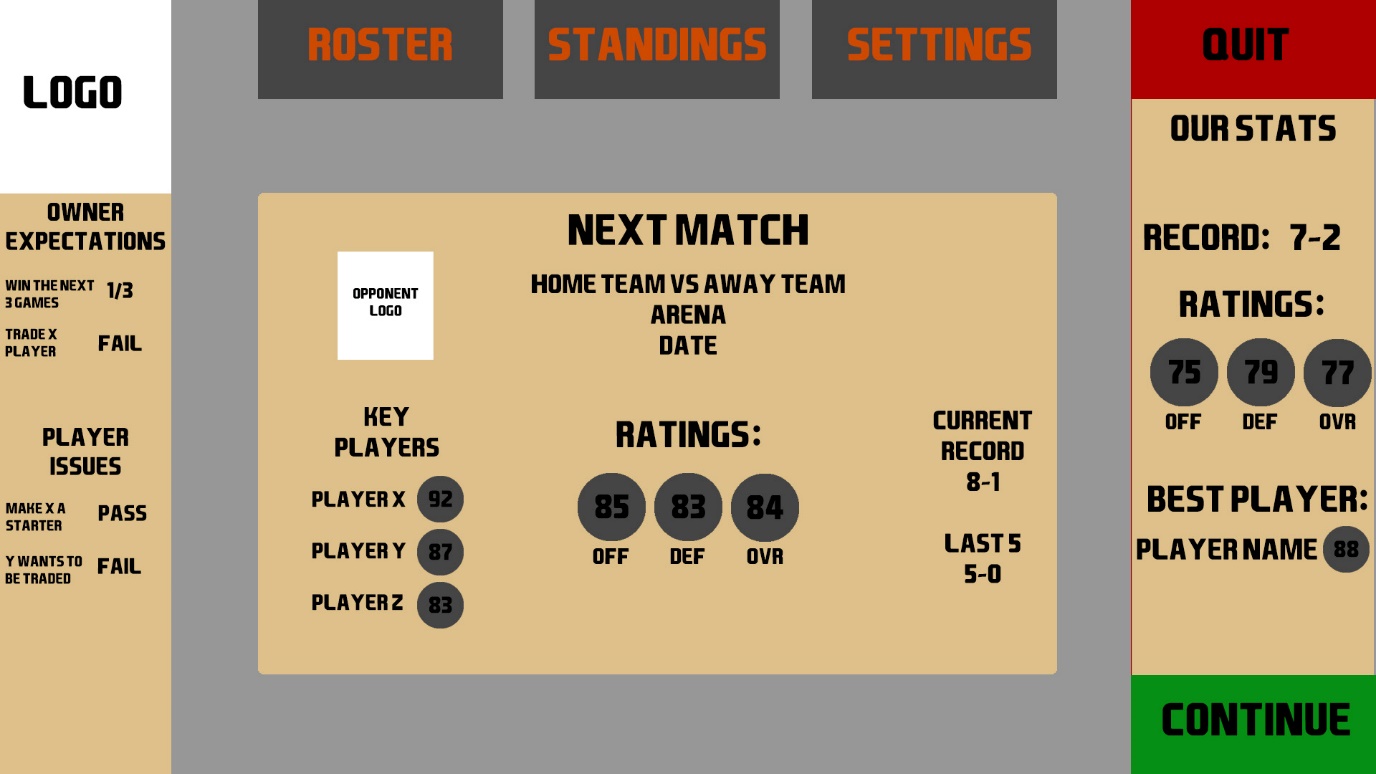
Main Menu:



An initial design made with Photoshop.

The purpose of the main menu is to be simple and straight to the point, it allows the user to either play, quit, or change settings and makes it easy to do so. I have used some images to fill up what would otherwise be empty space and made the buttons the focal point of the screen. On press of quit game, a pop-up confirmation will appear. Clicking settings will present a new screen to allow the user to edit sound, screen size etc. There will also be a how to play guide included in the settings screen.

Home screen after loading into game:



This is the main screen that the user will come across as soon as they load into the game and after each match.

In the centre of the screen will be information the player might want to know before advancing. This will include things like their next match, the rating and record of the team they face. How many games are remaining, whether the match is home or away etc.

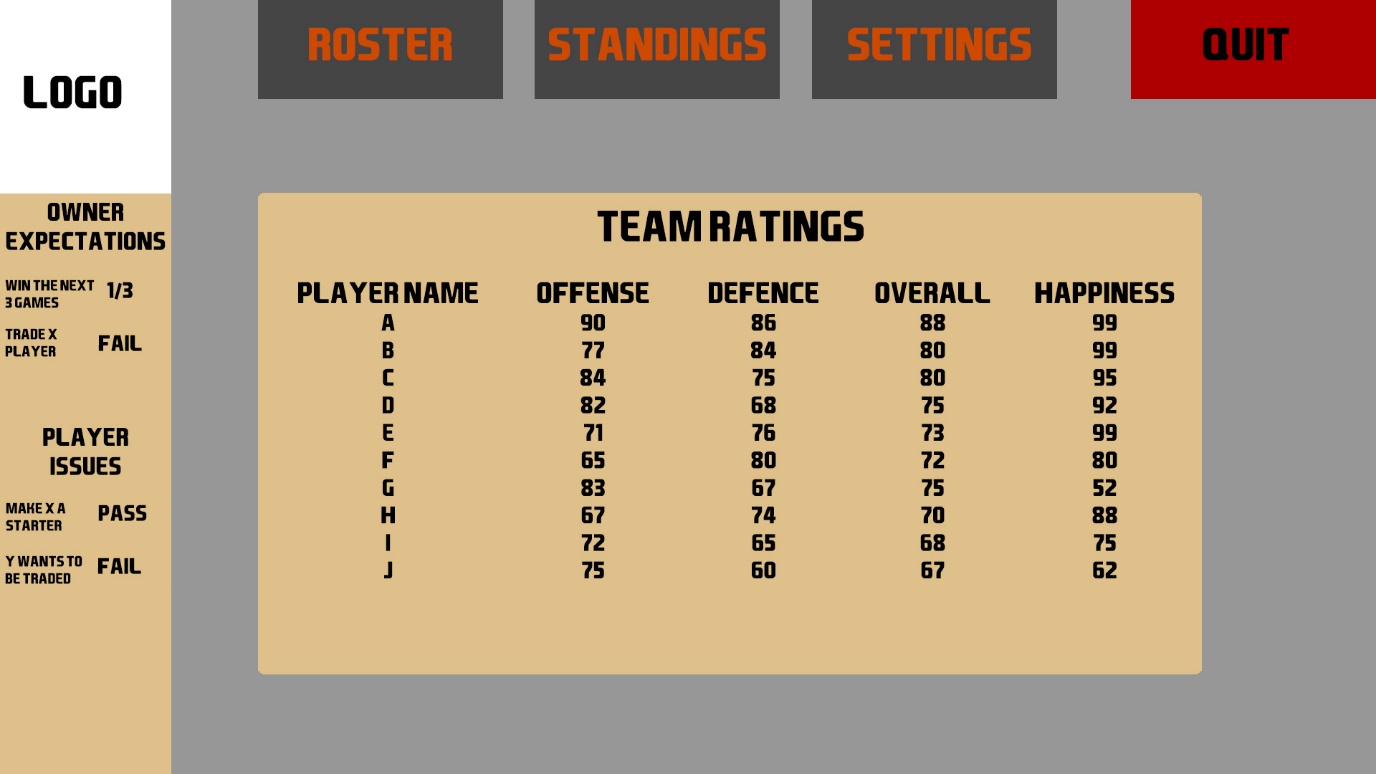
Across the top of the screen is a series of buttons. Each one will take the user to a new page. Each button will have a hover and clicked animation, and some may have dropdown options. For example, hovering over roster will give you the option to manage line-up, make a trade or look in detail at your team, this will show all ratings and happiness etc.

Another button is positioned in the bottom right to continue to the next screen. This may be advancing to a match or to a new issue with player or owner.

In the bottom left of the screen are issues and goals that the player must handle. These will affect the happiness of the person involved if fulfilled or failed.

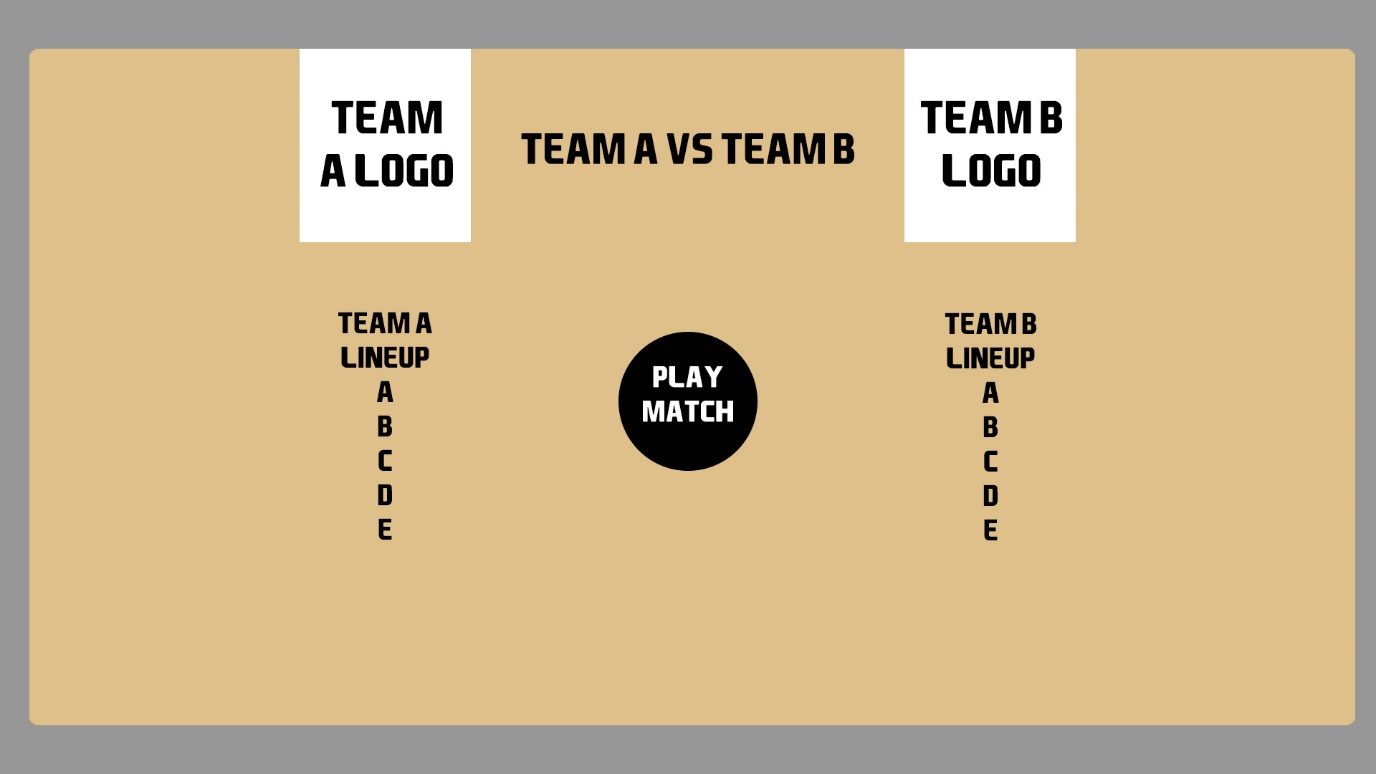
The purpose of this home screen is to provide everything which may be of immediate concern to the player.

In depth roster view screen, accessed via drop down menu of roster button.



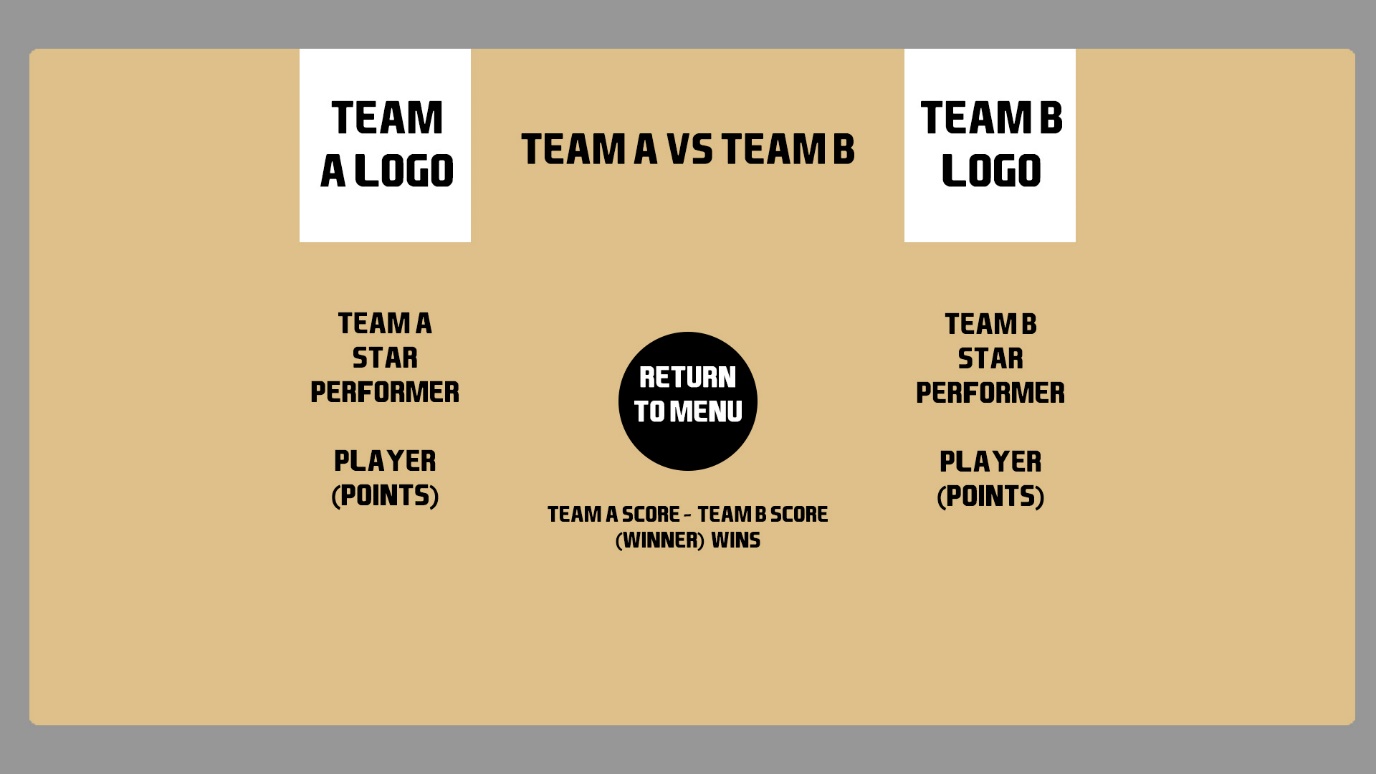
This is an example of one of the ‘sub-menus’ available in the game, which can be accessed by the buttons across the top of the screen. Most sub-menus will follow this design of removing the main menu content and provide in depth information. This screen keeps the information bar on the left as these issues can affect the player happiness shown in the central area. On these screens, the logo will work as a way of navigating back to the home page. At present, the planned sub-menus are team ratings, trades and edit line-up, all available in the roster drop-down. Standings will have drop downs for stats, where team ratings can be seen, and a league table to see team records. Settings is currently planned to have no dropdown and just take the user straight to the settings page on click.

Pre match screen design



This is a plan of the screen following the continue button being pressed on the home screen. The user is taken to a pre match screen where the lineups for the two sides are presented. The big button in the middle will then simulate the match, this is through the use of the match script.

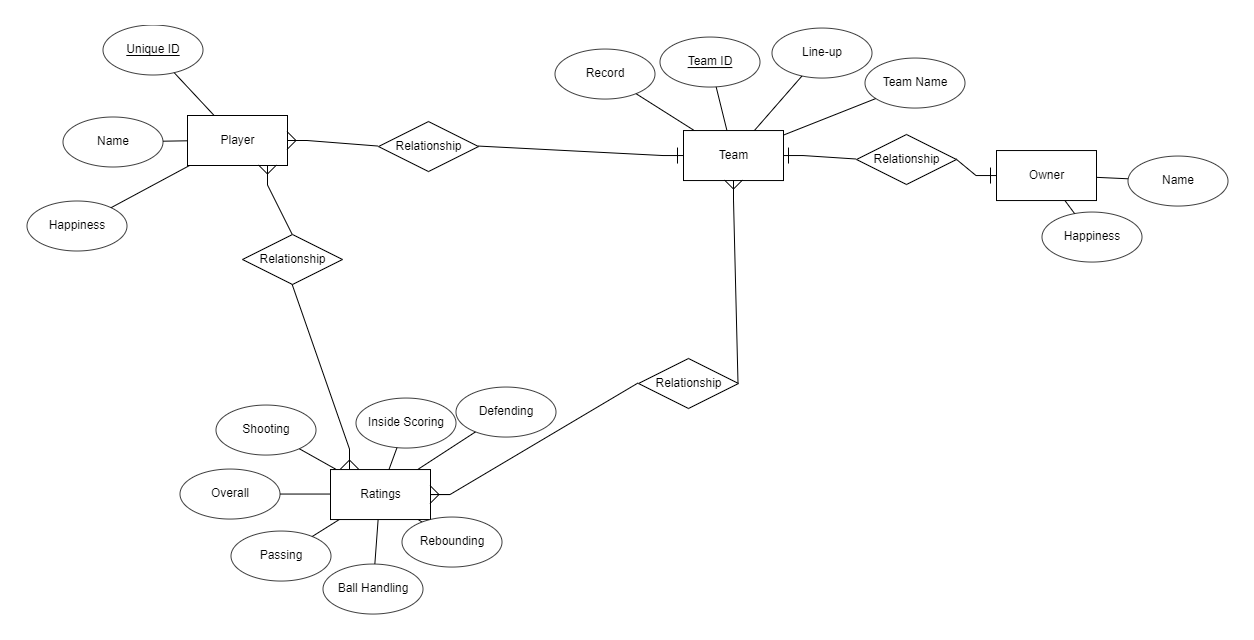
Post-match screen design



When the user has chosen to simulate the match, they will be taken to this screen. Here some details about the current match will be displayed. This includes the score, and the star performer for each screen. The button in the middle of the screen will now take the user back to the home screen, ready for the next match.

## Entity Relationship Diagram

An ERD displays the relationships of all entities available in a system. An entity is an object formed by a collection of data or properties. The purpose of this diagram is to show the logical structure of the system I am building.



(erdplus.com)

The following diagram shows the entities which will be present in the game. Player and team both have an ID as a unique identifier so that the system can easily find a specific player or team. This is very important for the trade system, as it is possible there are players with the same name, so the unique ID can be used when manipulating the database. At the moment, I only plan for an owner on the user’s team, as there isn’t much use for creating owners for the AI teams, therefore there is no unique identifier. The relationships work as follows, each team has one owner, multiple players, and multiple ratings, which are made up by the player ratings, each player belongs to one team but has multiple ratings, and an owner belongs to a single team. Each entity also has multiple attributes such as a player having a happiness value, or a team having its record.

## System Architecture

As stated before, the game is heavily User Interface based, meaning there are not many moving objects throughout the game. This also means that the quality of the gameplay will be more reliant on things going on behind the scenes rather than on the user’s screen. By this I mean that the scripts I implement will dictate the quality of gameplay. A script is a piece of C# code that is used in Unity Game Engine by being attached to a game object. In a 3D game this would be something like a character controller, however in my game, the scripts will be performing calculations.

During this stage of development, this is a list of the scripts I am planning to use, along with their purpose:

* Main Menu – This script will run the entire main menu scene. It allows for the clicking of buttons to lead to the correct result.
* Home Screen – The script which displays the home screen of the game, loads the information on the next match as well as goals, issues, and team stats.
* Player Database Script – This is intended more for referencing than to be directly used by an object. This script will hold the details for every player on each team, including name, ratings, and happiness. These details are then combined to determine each team’s overall ratings. After any action, this script will run an update function to make sure data is always accurate.
* Trade Script – This will run when the trade button is clicked. It will load the player database and allow the user to trade one of their players for another team’s player. My current plan is for this to work on a system that considers the players ratings and happiness, then tries to match this with a player on another team to provide the user with several trade options.
* Match Script – When the continue to match button is pressed, this script will be executed. It will load the match screen, displaying information about the player’s next match. On the press of the play match button, this script will use the team rating system, comparing both teams’ offense against the others defence, to calculate a result for the match. It does this via calculating each teams’ score and then the winner has the higher score, rather than just picking a winner. This provides a fairer way of calculating results and will not just award the stronger team with the win every single time.
* Happiness Script – This script again uses the database to perform calculations regarding player and owner happiness, based on events in the game. For example, after a game, winning players will see an increase in happiness, and players who are left on the bench will receive a decrease. Losses will also decrease owner happiness. Once the happiness of a player/owner reaches a certain level, and issue will appear in the home screen, which could then lead to the user being fired and losing the game.
* Standings Script – As the user plays their matches, the other teams in the league must also being playing matches to provide an up-to-date league table. This script will make sure the matches are being simulated in time with the user’s games. At present, I intend for the results to be pre-set so that I can dictate the difficulty of the game, as the player must be in the top 8 teams to reach the playoff stage. However, there is room to expand this to be random to provide a different experience every time the game is played.
* Line-up Script – Will be called when the user navigates to the line-up screen. This script will allow the user to change which players they use in their teams starting line-up. This will then update the team ratings on the database. All players effect the team rating; however starters are weighted more heavily due to their increased playing time. The player should tinker with their line-up a lot as the match script uses both offense and defence ratings to calculate results, so a balance must be found.

# Test Plan

My testing plan relates directly to the agile development strategy I am implementing. This is where a feature of the game is planned, designed, developed and tested before it is added to the game and I begin to work on the next function. This is called unit testing, where each ‘unit’ of the game is tested individually.

Unit testing makes sure that a broken mechanic is not being implemented which could damage the overall game. Once the mechanic is implemented I will then perform integration testing, where I test whether the new mechanic has meshed with the rest of the game properly. Sometimes two working features could create conflicting results, such as two pop ups appearing at the same time over one another, integration testing would look to resolve these kind of issues.

Once the entire game is developed, I will perform beta testing. This is where I will test the game by playing it, acting as the user, to make sure that the game is playable. I will do this because while it is useful to set out a list of tests to perform to make sure things work as expected, it is important to just run the game and see what bugs I might find that I didn’t test for.

As I will be testing the game myself, all testing will be white box testing, where the tester is aware of the internal structure and coding of the game. Black box testing would be helpful as it gives more of a players point of view, however due to the strict deadline for my project, white box testing is more efficient as I can easily tie any bugs to the part of the code which could be causing them.

I will also be completing functional testing, where each functional requirement will be tested against the system. This type of testing ensures that the game meets the demands of the system. Most other testing strategies only test what is present in the game, while this method could find things that have been left out but are required.

Below are my set of test cases, these are the tests which I intend to perform on the game. For each test case I will detail what I am testing, the process performed to create the test, such as a specific input and the expected result. These tests can then be performed and the columns of actual result and whether the test passed or failed can be filled in. The test cases will focus on the functional and unit tests, as these can be planned and have an expected result, whereas integration and beta testing are more tailored to finding bugs rather than testing results.

## Test Cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test** | **Process** | **Expected Outcome** | **Actual Outcome** | **Pass/Fail** |
| Clicking the button for the current page doesn’t reload the page. | Click the button corresponding to the page which is currently displayed. | Current page is not reloaded. |  |  |
| Quit Confirmation screen, to prevent accidental quitting. | Click the quit game button. | A menu pops up asking if user wants to quit. |  |  |
| Every button across the top bar leads to the correct page. | Click the button for each page across top bar. | Game navigates to the correct screen. |  |  |
| Records are updated after each match according to result. | Play a match and return to home screen. | Team record is updated with new result added. Records for all other teams are updated in standings. |  |  |
| Menu screens are updated to match the next opponent after each match. | Finish a match and return to home screen. | The next opponent section of screen is updated. |  |  |
| After a trade, each teams ratings are recalculated and roster is adjusted. | Make a trade. | On the roster screen, the players are switched and ratings are updated. |  |  |
| When a goal is met or fails, this is updated on the home screen. | Pass or fail a goal. | When a goal is passed, this is reflected and happiness for relating player or owner increases. If failed the happiness falls. |  |  |
| Once all reguar season games are finished, the game advances to the playoffs if the user team qualified. | Play all matches up to the playoffs. | If team qualifies, next match screen is updated with the playoff games. If not, game over. |  |  |
| If owner happiness falls too low, user is fired. | Make the owner happiness fall below threshold. | Pop up conversation with owner where player is fired and game over screen appears. |  |  |
| The result of winning the playoffs. | Win all playoff games. | Team wins the championship and winning screen appears. |  |  |
| Player happiness is updated after each game. | Play a game and then navigate to player happiness tab. | If the team won the game, player happiness rises, if they lose then it falls. Higher rated players who are not in the starting lineup also have happiness fall. |  |  |
| When a player’s happiness falls too low, they demand to be traded. | Continuously bench a star player until their happiness falls below threshold. | A message appears in player issues with player demanding a trade. |  |  |
| When trying to trade a player, the user is presented with a set of offers. | Go to the trading screen and select player to trade. | At least 3 offers appear for the player, user can then select which trade they want to perform. |  |  |
| When a game is played, results follow set boundaries. | Play 5 matches and observe results. | Teams scores must be between 70 and 140, star player can’t score more than 70, in most cases the stronger team wins, but there can be upsets. |  |  |
| Testing the full screen option. | Going to settings and pressing the full screen button. | Game goes into full screen mode. |  |  |
| Mute button can toggle sound. | Go to settings and press mute button. | If sound is on, it is toggled to off and vice versa. |  |  |

# Implementation Report

In line with my agile develpoment methodlogy, my prototype of the game focuses on executing a few well-made components, rather than basic implementation of every feature.

In this early version, I have aimed to develop the building blocks of the game, starting with the main menu. I have fully developed a scene which allows the user to change settings, quit game, and start a new game. This is in line with some of the basic functional requirements. My approach is to fulfill as many of the functional requirements as possible before moving onto additional features which are not necessary to play through the game.

I have also begun work on the game scene. This includes the startup home screen where the user can perform most actions, including navigating to other menus in game. The design of this menu follows that of my plans and is something that will be implemented at least loosely across all menu screens to create a consistency of theme across the game.

The next component of the game I will implement is the database as it is the next logical step to allow the biggest elements to function, like the trade system and match engine. I have some initial code for the database, but have not been able to integrate it across all menus where the data is required. Therefore at the time of writing, there is pre-set data filling in the columns that the database would automatically fill.

The current prototype has both front-end and back-end aspects to it, with menus being developed for the user to interact with on the front end, and scripts written to handle some of the inputs and data on the back-end. I thought it was important to work on both as this makes the game easier to test.

# References

<https://ieeexplore-ieee-org.proxy.library.dmu.ac.uk/stamp/stamp.jsp?tp=&arnumber=4042919> history of video games, first game

<https://books.google.co.uk/books?hl=en&lr=&id=PTrcTeAqeaEC&oi=fnd&pg=PT9&dq=history+of+video+games&ots=aoCm49-T76&sig=XTNzeVgN8GA6HS77lOMZ5mU5ncA&redir_esc=y#v=onepage&q=history%20of%20video%20games&f=false> History – pong

<https://www.businessinsider.com/sports-video-games-history2011?r=US&IR=T> First sports game

<https://imaginesports.com/news/sports-simulation-game> Definition of sports sim

<https://search-proquest-com.proxy.library.dmu.ac.uk/docview/235279181?pq-origsite=summon> Sports sim for students

<https://www.sportingnews.com/us/nba/news/nba-cavaliers-lebron-james-isaiah-thomas-injury-plays-nba-2k/e07jvdtoj391jbzsfj6qsz2e> Lebron plays 2K

<https://www.gamesradar.com/uk/football-manager-2021-review/> fm review

<https://www.samford.edu/sports-analytics/fans/2019/Does-Defense-Actually-Win-NBA-Championships> Defence wins championships stats