2019

Creating an Esports Coaching App using Machine Learning Techniques

C3486241 – COMPUTING BSC (HONS)

ANDREW HEATH

Abstract

The global video game culture has developed significantly since the early 2000's, with the sub-genre of Esports in 2018 having an estimated worldwide audience of over 299 million people. (Molina, 2018)

Esports as defined by the Telegraph newspaper (2017) is "competitive gaming at a professional level". At the professional level there is a highly developed structure for players to improve their skill sets, but for the amateur player there are fewer opportunities to be coached or have their performances analysed.

The purpose of this project is to create an application that any player can use to monitor their performance, providing data and guidance that can be used to improve their overall skill set. The Esport game chosen to develop this programme is Overwatch, a multiplayer first-person shooter game, developed by Blizzard Entertainment and released in 2016.

Contents

Introduction

In the past 5 years Esports have become a significant global entertainment media and attracted the attention of businesses and corporations worldwide. In Britain, video gaming, which includes Esports, in 2018 had sales of £3.86bn and is larger than home music and video sales combined (Ukie, 2018). Esports (electronic sports) for the unaware are Video games played at a professional and competitive level. The industry already has events and tournaments with bigger prize pools then some traditional sports tournaments. There are many different games that Esports is played professionally in such as League of Legends, Counter Strike: Global Offensive and Overwatch. Millions of people from around the world watch all of these different games when they are being played at a highly competitive level.

The game that this project focuses on is Overwatch, a team based first person shooter application where two teams of 6 players face off to complete different objectives on a map. Overwatch has a 40 million+ player base (Carpenter, N. (2018), n.pag.) with dedicated fans around the world. In the game, two teams of six characters, all with unique abilities and roles face off to complete certain objectives on different maps. As with many other games there is a casual side and a competitive side. The competitive side is usually referred to as "ranked" as there is a ranking system in place similar to chess' Elo system, called SR (skill rating). Players play and will get ranked somewhere between 0-5000 with clear divisions being, 1-1499 (bronze), 1500-1999(silver), 2000-2499(gold), 2500-2999 (platinum), 3000-3499(diamond), 3500-3999(master), 4000+ grandmaster. The top 500 player hold their own ranks as well. The average player usually lands around the gold and platinum rank (Vincenzo Milella, (2019), n.pag.)). Other games hold similar systems in their ranked play, and one of the objectives of this project is that the tools developed for Overwatch can be applied to other game platforms to help individuals develop.

Now that an introduction to overwatch has been done, some more detail on this project. Unlike the professional Overwatch league, many players don't have access to resources such as coaches or analysts. A player is able to look up their own statistics using the player lookup tool, a website provided by Blizzard, the creators of Overwatch. However, the player lookup tool only provides the raw statistics though and doesn't provide any graphs or manipulates the data in anyway. The goal of this project is to help players who don't have access to a coach or analyst. To help them get better and climb the ranked ladder. This is accomplished by using a machine learning method. The machine learning method employed in this project is a decision tree with other systems being evaluated.

The main purpose and ultimate goal of this project is to use a machine learning method to help guide and better players of Overwatch. Other goals of the project are to review relevant literature, design a product based on the research done, use a relevant design methodology, and evaluate the performance of the product by giving it to players of Overwatch to provide critical feedback. Some in depth requirements for the product are an area for statistics and area for performance over time and an area for the coaching aspect.

The structure of this report is as follows, a literature review, looking at relevant literature to the machine learning field. This is then followed by a technology review, which looks at the technologies that are and may be used in the project; a discussion about the design methodology used and then the implementation and testing done in the project. The evaluation of both the project as a whole and the product itself follows after. After the conclusion is the bibliography, and following the bibliography is the appendices.

Literature Review

This section discusses what previous approaches have been used in similar scenarios, what method of machine learning is used and what other choices or methods could have been used based on previous studies. When first starting to research for this project, a review of academic papers and journals that had already done something similar to basis of the project was completed, namely using machine learning algorithms that help coach players and improve their performance. The initial action was finding a definition for a machine learning algorithm. e.g. "A machine learning algorithm is expected to produce a model capable of predicting the correct class of a given input." (Goodfellow et al., 2018, pg. 59). Applying this to mean, in the relation to this project, the class we would like to predict being the Win, lose or draw outcome of a match. Signifying, a win as an increase in skill and a loss being a decrease in skill. The inputs given would be the statistics that are collected from the player profile. As the more statistics are collected, better prediction of results, leading to better coaching advice given to the player.

There are two main paths for machine learning, supervised and unsupervised learning. Unsupervised being a method to get results using only input data, for example clustering. Clustering has been used to identify well performing teams and underperforming teams in previous studies. This helped the authors, R. P. Bunker and F. Thabtah, identify which teams, and therefore which statistics to train their models with in their research.

Supervised being a method to predict based on both input and output data. Classification is an example of supervised learning and has historically been the method for predicting sports results. Classification involves building a data set and then using the data set to predict results. Many of the papers reviewed such as, a machine learning framework for sport result prediction by R. P. Bunker and F. Thabtah, used this method when applied to gambling prediction. This approach has the same output class as this project, predicting if a team or player would win, lose or draw. Classification has many different approaches such as decision trees and Artificial Neural Networks (ANN). Through the document review and other research, it was concluded that the preferred option was to use a decision tree, but the discussion will cover the benefits and disadvantages of using an artificial neural network as well.

Artificial neural networks are one of the most common approaches for Machine Learning (ML) mechanisms in relation to the sport prediction problem. "An ANN usually contains interconnected components(neurons) that transform a set of inputs into a desired output. The power of ANN comes from the non-linearity of the hidden neurons in adjusting weights that contribute to the final decision." (R.P. Bunker, F. Thabtah (2017) pg. 2). The output from the network often depends on the input features and other components such as the weights. The ANN is constructed after processing the training data set that contains the features required to build a classification model. An ANN is constructed of 3 layers, an input layer, output layer and a hidden layer. The weights in the ANN are continually adjusting to provide a high degree of predictive accuracy.

Purucker (1996) uses an artificial neural network to predict the results of the national football league (NFL). Data from the first eight rounds of the competition along with five features were used. These being yards gained, rushing yards gained, turnover margin, time of possession and betting line odds. An artificial neural network with backwards propagation was then used to give a predictive accuracy of 61%. The limited number of features and amount of training data was noted to give a low accuracy. Another study improved on Purucker by using more data and different data features, allowing the ANN to perform better than experts predicting results on the same matches (M.C. Purucker (1996)).

An artificial neural network would be the ultimate goal to implement into this project as they give a high degree of predictive accuracy compared to some other methods. When applied to this project it is necessary to identify which data points would be needed to use to give the highest degree of predictive accuracy. The player lookup tool gives a wealth of statistics related to a specific player and even for the specific characters. Aiming to make the system robust, focusing on the non-character specific statistics would be best. These statistics being objective kills, solo kills, deaths, all damage done, healing done, and time spent on fire. These stats will prove a good gauge on how well a person is performing. The statistic to be predicted in this case is the SR gain/loss. To implement an ANN into the project it would be necessary to build up a data set from the player, because a previous match history is not available from the player lookup tool. Users would have to spend a while not getting any coaching feedback to build up a data set before predictions could begin. (see appendices for all the data statistics available).

Once a test data set is built i.e. 10 entries, prediction could start happening. Prediction would happen by suggesting a player focus on one or more of that statistic and trying to improve it and seeing if it has any bearing on the SR gain of the player. After every time the player uses the Application a new data entry will be added and compared with the last predicted to see if chosen statistics were the right ones to focus on, in a simple form of backwards propagation. Also depending on what stats are chosen, the coaching area of the app may use more general language for example: Play more aggressive, support your healers and play more defensive.

Another method of classification are decision trees, which is the method chosen for this project. The main advantage of using a decision tree over an artificial neural network is that it has high readability and fast classification capabilities. This makes it more applicable because decision trees allow for feature selection. Meaning the learning process can be fine tuned by leaving and including different attributes.

A decision tree model is represented as a tree with the nodes in the tree representing characteristics of a data set. It can be considered as a set of if-then rules or a conditional probability distribution defined in the eigenspace and class space. Implementing a decision tree consists of these steps: feature selection, decision tree generation, and decision tree pruning.

The decision tree algorithm was first developed in the 1960s; in the 1970s J Ross Quinlan proposed a similar algorithm, ID3, that aimed to reduce depth of trees but ignore the number of leaves (Tang, X. et al, 2018, p. 239). Quinlan in the 1990's developed the C4.5 algorithm that is an extension of the ID3 algorithm. It made great improvements on the missing value processing, pruning and derivative rules of the predictive variables. These algorithms use the Gini index to create a simple two fork trees structure to classify new cases so that the missing data can be processed and more effectively classified and predicted.

A decision tree contains a root node, several internal nodes and several leaf nodes. Leaf nodes are nodes that denote decision results, and each node relates to a property test. Internal nodes are where a decision takes place. The path to a leaf from the root is related to a decision test sequence. This method of machine learning is designed to produce a strong generalization ability, in the sense that if a sample is presented that hasn't been encountered before it can determine its outcome class with high accuracy.

Decision trees are a common machine learning method, and also a natural decision-making process that humans do regularly. For example, when faced with buying an apple, you would check for certain markers. “Is it rotten?” “does it have blemishes?” “is it soft?” and so on but eventually you would get a decision about the apple. This is shown in a tree diagram below (see fig 1)

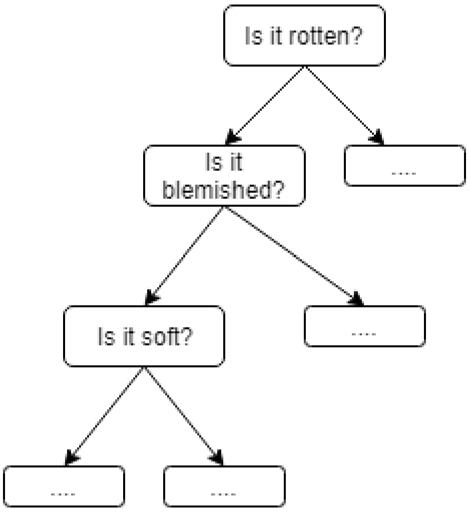


Figure 1.

This is a very simple example but following the same principles and applying it to an individual player will generate a tree that divides by the different stats. “Previous studies of sporting events have shown that accuracy varies from sports to sports, with an upper limit of 70 to 80 percent.” (Tang, X. et al, (2018), p. 240). The most critical factor in getting these high accuracy predictions is the quality of the statistics chosen to assess a player on.

The way a decision tree would be implemented into this project is that a player would play the game then use the application. Once the player has used the application a record of his statistics would be held in some form such as a .csv file, .txt file or on a SQL server. This would be repeated multiple times until a suitable data set has been constructed, 10+ entries into the data set would be the start of a suitable data set. Once a data set is constructed, a decision tree would be built from the data set. Example queries would be built from an average of player from all different skill brackets. Each query would relate to a different playstyle. For example, aggressive, defensive and passive. The results of the queries would tell a player if that is the play style needed to win their current games based on their most recent performance, the data set that has been built up. The data set will continue to build up over time, as the player uses the application more and more, leading to better accuracy and higher quality coaching.

Technology Review

This section discusses what and why the product was chosen as a mobile application and whether a web application would have been a better solution and why Xamarin was selected as a development platform compared to other development platforms. This section will also be used to look at current market solutions or similar market solutions that have provided foundational principles for this project.

The first significant selection is in regard to the platform needed to develop the product on, either as a mobile or as a web application. There are already a number of statistics gathering websites such as Winston’s lab or op.gg. Both of these just gather statistics though and do not offer any coaching advise. All of these websites further cemented the decision to develop the product as a mobile application, as it was felt that another web-based statistic site would be overshadowed by all the other applications on the market.

The next step was to choose how to develop the mobile application, specifically an application built for Android. Android application architecture is based on Java and it is the norm to develop apps for android in Java. After conducting some research an alternative with better advantages presented itself. Xamarin is a development solution that uses the .NET eco-system to deliver and build apps that can be deployed on multiple systems, android included. The .NET environment also has the advantage of using visual studio allowing for NuGet packages to be added to project easily and quickly. Also, c# is thought to be a faster language than Java overall. All of these advantages allow for faster and quicker production and workflow allowing for development turnarounds to be very quick. Other development solutions were considered such as, Android development studio or building the app in eclipse (an IDE) using Java.

The disadvantages of choosing Xamarin over another android development suite are that currently there is less support compared to other solutions although with the compatibility between android development and .net development, although the issues are not insurmountable. Many of the android widgets and functions have an equivalent in Xamarin, but it just takes some effort to find the solutions. One of the key advantages of using Xamarin is that Xamarin has access to NuGet packages, which allows easy additions of extra code libraries. This means solutions for certain development problems can be found easily and quickly.

The next main piece of software selected was the means for accessing the user's statistics. Unlike some other companies, Blizzard, the creators of Overwatch do not provide an application programming interface (API) although they do provide a player lookup tool, which when given a player Battle.net ID, is able to provide statistics in the form of a web page. Other game companies provide similar services or have their own API to provide statistics. Web scraping is a technique where providing a URL, a Hypertext Mark-up language HTML then can be parsed for the data needed. Various solutions are available, with some being code library's that can be purchased or open source versions that have similar functionality.

Iron Webscraper is a professional C# library, and is available commercially. It is marketed as a simple and flexible solution that can handle errors by itself and only requires a small amount of code to be written. Iron Webscraper also offers several advanced features such as creating dummy accounts, throttling your web requests and advanced caching. These features are very useful for other products but were not needed for initial development on this product, so another Web scraping library was found. The HTML agility pack (HAP) is an open source web scraping library by ZZZ projects. HAP is a much more basic web scraping library but is very easy to implement compared to Iron Webscraper and is free. HAP is designed to be a straight forward implementation using .NET's HTML doc function and using the nodes of the HTML document. It requires more lines of code and it's not as nuanced in its error handling as Iron Webscraper but fits the purpose needed.

Current market solutions for Overwatch are available but any that add a coaching aspect are usually a paid subscription service that uses real life coaches to coach player based on videos of a player's gameplay, such as omni coach. Other solutions include machine learning but require you to download a piece of software to run while playing, which can sometimes be a bannable offence according to a company's terms of service. There is some uncertainty at the time this product is being developed as some previous technologies have been outright banned by Blizzard, but some have not. Other current products that offer enhanced statics representation are Overwatch tracker and Overbuff. Both of these websites offer different ways of representing your stats. Overbuff focuses on laying out all the stats for each hero, stating with your most played hero at the top and least played at the bottom. Overbuff uses progress bars to compare your hero stats against other players and determine what percentile of player you are for specific statistics. Overbuff also has its own ranking system that displays where you are compared to other people who used the site. Overwatch tracker is more focused on using a clear layout and representing data in easy to read formats such as graphs. Most of the graphs are simple line/bar graphs that show the trends in players performance in statistics such as kd (kill/death ratio), win percentage, damage done, and healing done. League of Legends (LOL) is another competitive game with a thriving Esports scene. LOL has its own stat page in the client launcher, allowing players to see and compare their statistics with other players. The thing that stands out is the use of radar graphs. Radar graphs are used in sports a lot of the time to represent a player's statistics and can offer an easy to understand graphical representation.

Design Methodology and Product Design

This section is dedicated to reviewing the design methodology and design steps chosen during development of the project. The methodology selected was the Agile development framework. Other methodologies were considered or incorporated, such as RAD (rapid application development) and the waterfall method. This section will offer an insight to how the Agile method is applied to the product development. A look into the design techniques will also be reviewed in this section as well.

"Agile software development refers to a set of computer programming methodologies that emphasize flexibility, collaboration, efficiency, simplicity, and most of all, delivering working product to end users within short timeframes." (Codington-Lacerte ,2018). The agile method was created as a counter to the standard business practises that were seen to impede innovation and undervalue customers and employees. The agile method revolves around "sprints" which are a set amount of time that you work through a set of tasks. The package then reports and reflects on the performance at the end of the sprint, improving how effectively your team works.

The project needed to transfer previous experience with using Agile software in a team environment to one where it is applied to an individual. Transitioning the method to singular person was challenging. A strict plan was needed and a suitable scheduled was implemented. A Gantt chart is a useful tool for long term planning and was produced at the start of the project to give a rough timeline to produce the product. (see appendices). There are many tools online to help support an Agile methodology development, taiga.io is a website that had been used previously alongside the agile methodology. Taiga worked well as for a group using the agile method, as it allowed tasks to be assigned to specific people and also for a value to be assigned to the tasks. So, at the end of a sprint, the performance of the group as a whole could be assessed. Instead, the project opted to use Trello, which is a much more basic system., a sort of digital representation of post-it notes, which made it more robust and user friendly for the individual to use. Trello, along with the agile methodology, allows you to break down your development into specific tasks, i.e. add Login system and test. This system allows development to be very fluid. Testing of systems is also very fluid as it's up to the developer to test at any stage of development. This means you can tackle certain problems on a case by case basis rather than be forced to complete entire systems then go back and figure out what went wrong. This sort of development does not have key stages, rather it has Key sections. For example, the "login" system was the first keys section of this project followed by the Web scraper, and etc. testing and evaluation of these systems happens during the development process. This differs from other methodologies such as waterfall method. The waterfall method would have required completion of entire parts of the design progress before moving on. This can lead to a lot of iterations and can be very time consuming. Rapid application development (RAD) is another Design methodology considered for implementing at the design stage. Rapid application development involves rapidly building prototypes and getting feedback from the end user. The feedback continually affects the development of the product development and this process keeps going until the product is finished. An informal version of this was implemented by always asking end users if they enjoyed the product as it was being was developing and making sure to update the development if the feedback necessitated it.

The combination of rapid application development and Agile led to a very fluid product development, as there is no testing stage or development stage, rather they are meshed into one continuous cycle. This fluid product development allowed processing large portions of coding quickly. Other than development and testing the other key stages were designing and evaluating. Evaluation was done at the end of sprints, but an overall product evaluation was conducted as well. The evaluation at the end of sprints were conducted by looking back at the tasks completed during the sprint. Comparing this to other sprints allowed for a rough idea of how well development was proceeding and allowed for adjustment in pace of work.

Designing the product involved a combination of informal and formal design techniques. The product is expected to be used by many different players with varying levels of aptitude. To reflect this, the product needed to have a wide range of user input from the start and so interviews were conducted at the beginning of the project. (transcripts located in appendices) This intended to gather an idea of what end users would expect from the project. From these interviews a specification of what users would like to see was constructed that would form the basis of the product.

These product requirements were

- an easy way to see player statistics,
- an easy to read graph for performance,
- an area for the coaching aspect.

These were the three requirements that most people came up with. In addition to those identified above the use of a radar graph in the statistics area was added as this tool featured strongly in the earlier research   Wire frames were also designed to help with providing an image of what the app would look like. These wireframes were then sent out to some users to get feedback. The feedback was considered, and a second draft of wireframes was developed. Another aspect of the design that was developed was the backend data structure, as the product needed to store all of the data as it was generated. This went through a couple of revisions.   Again, a key aspect of the development platform used, Xamarin, allows for rapid deployment of prototype applications.

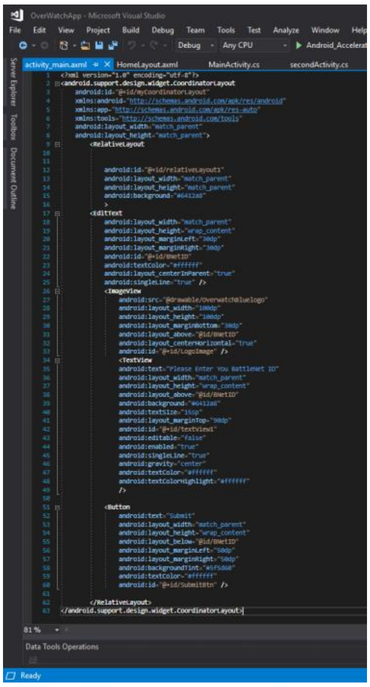
Rapid application development and Agile methods helped stimulate an effective development environment even though a stricter development cycle such as the waterfall method, may have helped with timelines. The Rapid application / Agile combination allowed freedom to work on other options which could be quickly drafted to evaluate their opportunity. Although this allowed for a large degree of freedom of design, it did mean there was increased room to experiment, which could be considered wasted time. A more formal design method would have helped keep to stricter timelines and possibly a better product.

Implementation and Testing

This section discusses the development of the product. The final product developed is a small application that collects a user's data from a specific webpage then uses that data to help improve the user's performance in Overwatch (a video game). The application is built for android devices but built using a technique that makes it easily portable to other operating systems. There are three main parts to this application:

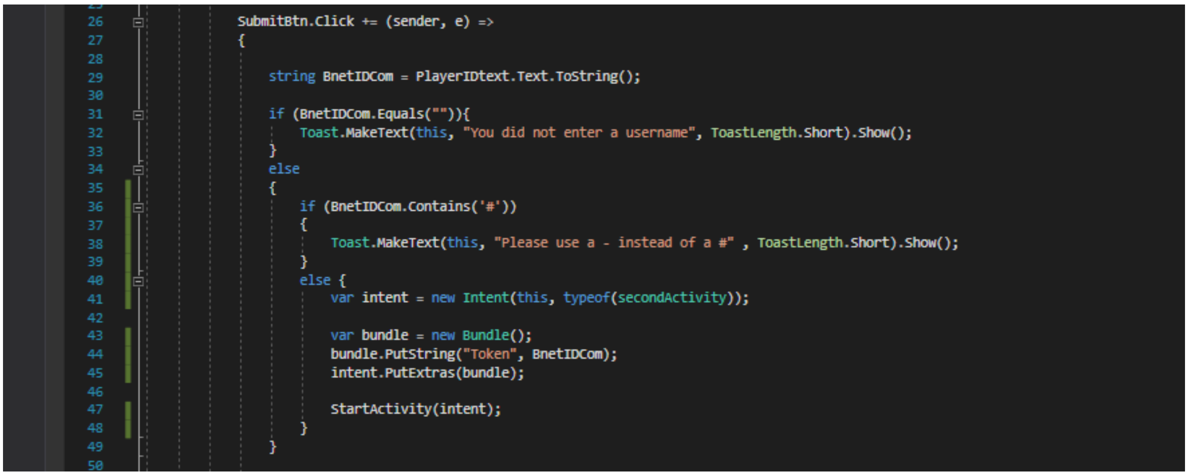
- the application itself (user interface and android code),
- the web scraper (to gather data), and
- the coaching aspect (using machine learning).

The application was built using the Visual studio IDE and the Xamarin development platform. Xamarin allows you to code applications for platforms that wouldn't normally use c#. The first part of the application is the user interface using Android. Android UI is built using XML and AXML files. These xml files can either be normal XML files or can contain android widgets that allow for better looking and more fluid design. The three main "pages" or activities as android calls them are the "login" page (In quotes because no login procedure is filled out, just collecting the players battle net id), the stats page and the coaching page. The "login" page is a simple text field with a submit button, represented in AXML and c# code here:



```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
    android:id="@+id/myCoordinatorLayout"
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
        <RelativeLayout


            android:id="@+id/relativeLayout1"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:background="#6412a8"
            >
        <EditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginLeft="30dp"
            android:layout_marginRight="30dp"
            android:id="@+id/BNetID"
            android:textColor="#ffffff"
            android:layout_centerInParent="true"
            android:singleLine="true" />
            <ImageView
                android:src="@drawable/OverwatchBluelogo"
                android:layout_width="100dp"
                android:layout_height="100dp"
                android:layout_marginBottom="30dp"
                android:layout_above="@id/BNetID"
                android:layout_centerHorizontal="true"
                android:id="@+id/LogoImage" />
                <TextView
                android:text="Please Enter You BattleNet ID"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_above="@id/BNetID"
                android:background="#6412a8"
                android:textSize="15sp"
                android:layout_marginTop="90dp"
                android:id="@+id/textView1"
                android:editable="false"
                android:enabled="true"
                android:singleLine="true"
                android:gravity="center"
                android:textColor="#ffffff"
                android:textColorHighlight="#ffffff"
                />

            <Button
                android:text="Submit"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_below="@id/BNetID"
                android:layout_marginLeft="50dp"
                android:layout_marginRight="50dp"
                android:backgroundTint="#5f5d60"
                android:textColor="#ffffff"
                android:id="@+id/SubmitBtn" />

        </RelativeLayout>
</android.support.design.widget.CoordinatorLayout>
```

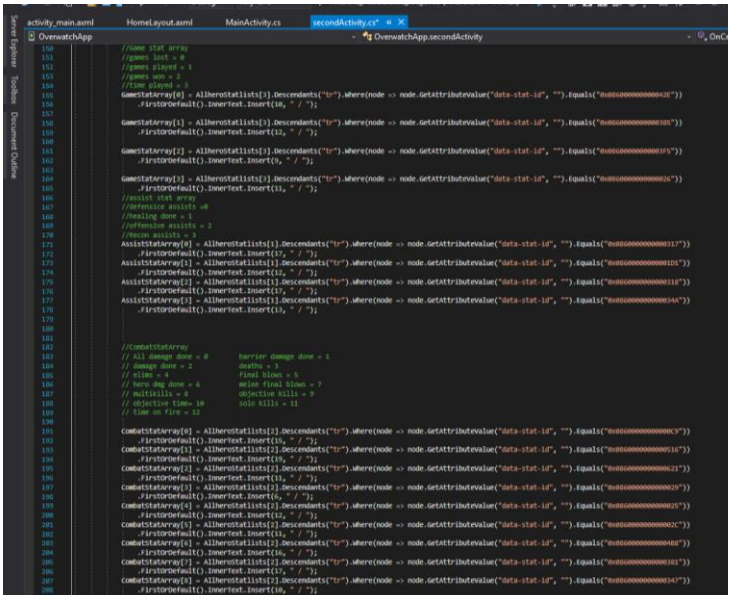
There are no forms in android programming as there are in other languages like HTML. In order to stop a false submission, for example if an empty field or a submission is not in the right format, an error message is displayed (unique to the error). Forms are submitted by reading a text field once an activity listener has been activated.



```
SubmitBtn.Click += (sender, e) =>
{

    string BnetIDCom = PlayerIDtext.Text.ToString();

    if (BnetIDCom.Equals("")){
        Toast.MakeText(this, "You did not enter a username", ToastLength.Short).Show();
    }
    else
    {
        if (BnetIDCom.Contains('#'))
        {
            Toast.MakeText(this, "Please use a - instead of a #" , ToastLength.Short).Show();
        }
        else {
            var intent = new Intent(this, typeof(secondActivity));

            var bundle = new Bundle();
            bundle.PutString("Token", BnetIDCom);
            intent.PutExtras(bundle);

            StartActivity(intent);
        }
    }
```

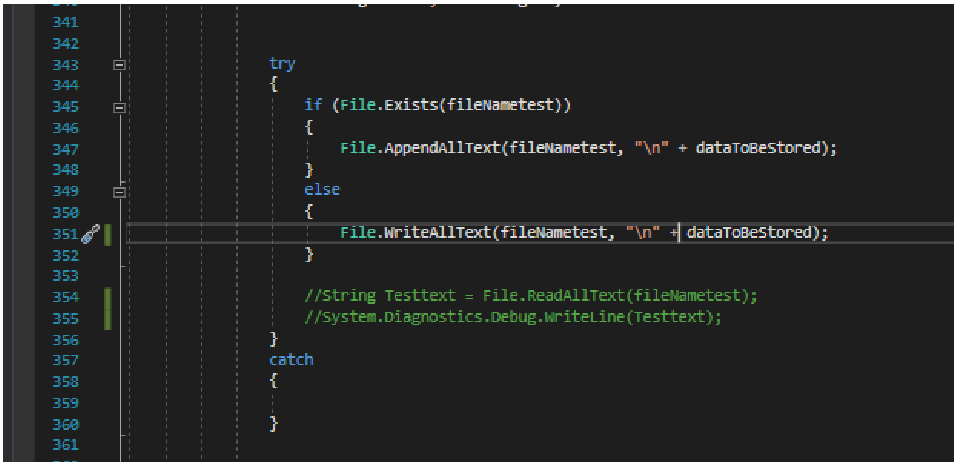
There have been many different approaches to error message reporting in android. The method chosen for this product is the "toast" message which is shown in the screenshots above. Other methods include a message box that requires user input to continue. Using the toast messages is the preferred method for short messages for the latest version of android. The Username that is entered is then passed through to the second activity, the statistic page using a "Intent". Intents are used to start different activities and to pass information between different activities in android.

The statistic activity is where the web scraper gathers the statistics to be used in the rest of the app. The HTML agility pack is the main driving force behind the web scraper. The html agility pack uses the html document object in c# and builds an object it can interact with. From the object, html nodes can be selected, such as divs and forms. More accurate selection can be carried out by using the id or class of the div or HTML container being selected, giving the ability to select specific instances on the html document. Implementing this to gather statistics from the player lookup tool, was difficult. On other websites data is stored in a table and you can use the html agility pack to gather data easily. The player lookup tool website stores its data in unique divs with their own 16-digit code. The statistic activity also contains the radar graph which is implemented using a library built for Xamarin by Alois Deniel called micro charts.



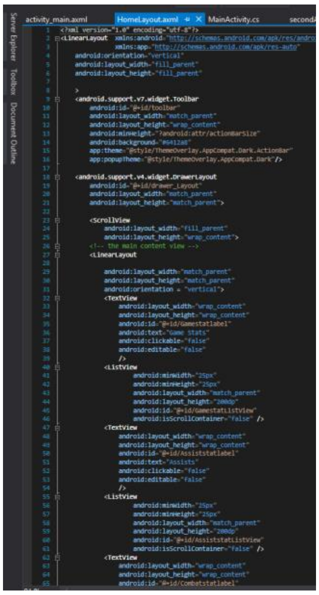
```
//Game stat array
//games lost = 0
//games played = 1
//games won = 2
//time played = 3
GameStatArray[0] = AllheroStatlists[3].Descendants("tr").Where(node => node.GetAttributeValue("data-stat-id", "").Equals("0x086000000000042E"))
    .FirstOrDefault().InnerText.Insert(10, " / ");

GameStatArray[1] = AllheroStatlists[3].Descendants("tr").Where(node => node.GetAttributeValue("data-stat-id", "").Equals("0x0860000000000385"))
    .FirstOrDefault().InnerText.Insert(12, " / ");

GameStatArray[2] = AllheroStatlists[3].Descendants("tr").Where(node => node.GetAttributeValue("data-stat-id", "").Equals("0x08600000000003F5"))
    .FirstOrDefault().InnerText.Insert(9, " / ");

GameStatArray[3] = AllheroStatlists[3].Descendants("tr").Where(node => node.GetAttributeValue("data-stat-id", "").Equals("0x0860000000000026"))
    .FirstOrDefault().InnerText.Insert(11, " / ");
//assist stat array
//defensice assists =0
//healing done = 1
//offensive assists = 2
//Recon assists = 3
AssistStatArray[0] = AllheroStatlists[1].Descendants("tr").Where(node => node.GetAttributeValue("data-stat-id", "").Equals("0x0860000000000317"))
    .FirstOrDefault().InnerText.Insert(17, " / ");
AssistStatArray[1] = AllheroStatlists[1].Descendants("tr").Where(node => node.GetAttributeValue("data-stat-id", "").Equals("0x08600000000001D1"))
    .FirstOrDefault().InnerText.Insert(12, " / ");
AssistStatArray[2] = AllheroStatlists[1].Descendants("tr").Where(node => node.GetAttributeValue("data-stat-id", "").Equals("0x0860000000000318"))
    .FirstOrDefault().InnerText.Insert(17, " / ");
AssistStatArray[3] = AllheroStatlists[1].Descendants("tr").Where(node => node.GetAttributeValue("data-stat-id", "").Equals("0x086000000000034A"))
    .FirstOrDefault().InnerText.Insert(13, " / ");



//CombatStatArray
// All damage done = 0       barrier damage done = 1
// damage done = 2           deaths = 3
// elims = 4                 final blows = 5
// hero dmg done = 6         melee final blows = 7
// multikills = 8            objective kills = 9
// objective time= 10        solo kills = 11
// time on fire = 12

CombatStatArray[0] = AllheroStatlists[2].Descendants("tr").Where(node => node.GetAttributeValue("data-stat-id", "").Equals("0x08600000000000C9"))
    .FirstOrDefault().InnerText.Insert(15, " / ");
CombatStatArray[1] = AllheroStatlists[2].Descendants("tr").Where(node => node.GetAttributeValue("data-stat-id", "").Equals("0x0860000000000516"))
    .FirstOrDefault().InnerText.Insert(19, " / ");
CombatStatArray[2] = AllheroStatlists[2].Descendants("tr").Where(node => node.GetAttributeValue("data-stat-id", "").Equals("0x0860000000000621"))
    .FirstOrDefault().InnerText.Insert(11, " / ");
CombatStatArray[3] = AllheroStatlists[2].Descendants("tr").Where(node => node.GetAttributeValue("data-stat-id", "").Equals("0x0860000000000029"))
    .FirstOrDefault().InnerText.Insert(6, " / ");
CombatStatArray[4] = AllheroStatlists[2].Descendants("tr").Where(node => node.GetAttributeValue("data-stat-id", "").Equals("0x0860000000000025"))
    .FirstOrDefault().InnerText.Insert(12, " / ");
CombatStatArray[5] = AllheroStatlists[2].Descendants("tr").Where(node => node.GetAttributeValue("data-stat-id", "").Equals("0x086000000000002C"))
    .FirstOrDefault().InnerText.Insert(11, " / ");
CombatStatArray[6] = AllheroStatlists[2].Descendants("tr").Where(node => node.GetAttributeValue("data-stat-id", "").Equals("0x0860000000000488"))
    .FirstOrDefault().InnerText.Insert(16, " / ");
CombatStatArray[7] = AllheroStatlists[2].Descendants("tr").Where(node => node.GetAttributeValue("data-stat-id", "").Equals("0x0860000000000381"))
    .FirstOrDefault().InnerText.Insert(17, " / ");
CombatStatArray[8] = AllheroStatlists[2].Descendants("tr").Where(node => node.GetAttributeValue("data-stat-id", "").Equals("0x0860000000000347"))
    .FirstOrDefault().InnerText.Insert(10, " / ");
```

This means a line of code needs to be written for every piece of data you would want to grab. An added delimiter is added before the gathered data is stored in an array to allow for easier manipulation later on. Each section of data is stored in its own array as a string. The delimiter allows for easy splitting of the data so the numerical value can be used by the coaching area of the app. The entire Web scraper method uses c#'s async value. This allows for the app to be usable if the app is not able to access the player look up tool, if it were to be unavailable for example. An instance of the gathered data, from the averages section of the players statistics, is stored in text file. This is the file that the coaching section of the application read from. The file is generated from the players name, allowing for multiple people to use the app on the same device.



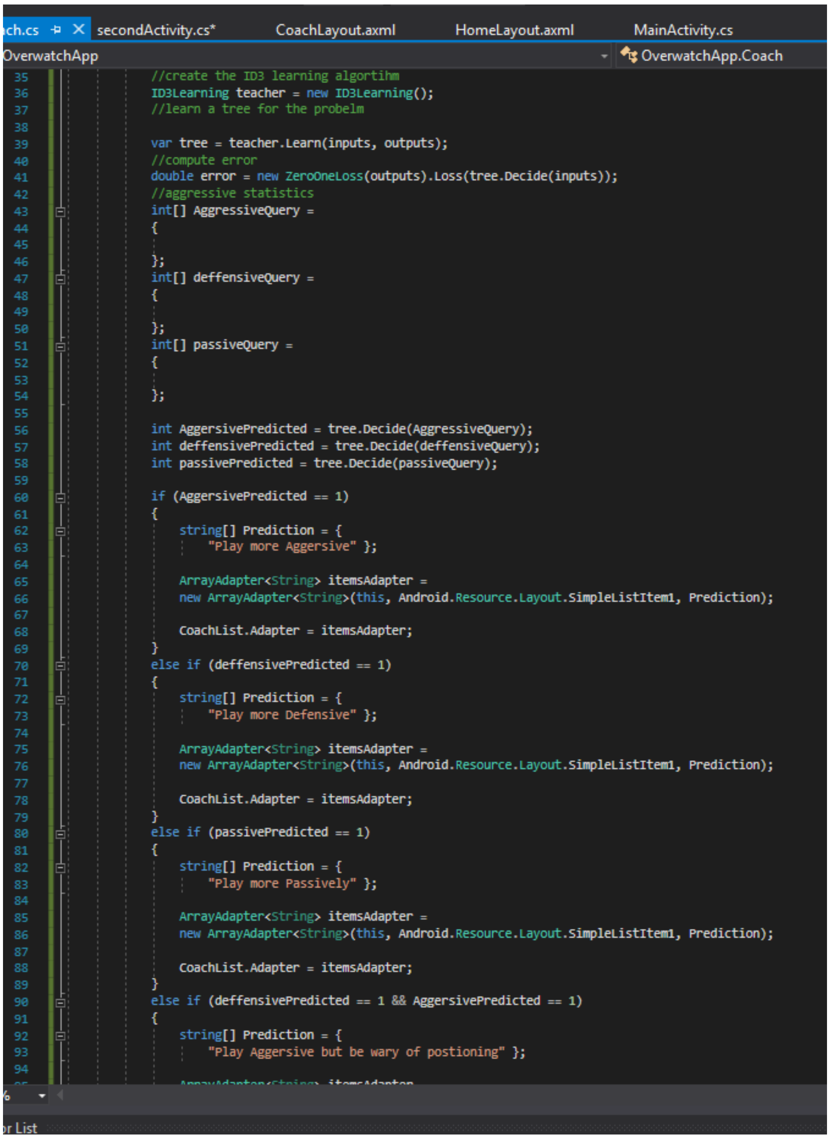
```
try
{
    if (File.Exists(fileNametest))
    {
        File.AppendAllText(fileNametest, "\n" + dataToBeStored);
    }
    else
    {
        File.WriteAllText(fileNametest, "\n" + dataToBeStored);
    }

    //String Testtext = File.ReadAllText(fileNametest);
    //System.Diagnostics.Debug.WriteLine(Testtext);
}
catch
{

}
```

The coaching section of the app is accessed through the navigation drawer. The navigation drawer is available by swiping from the right, revealing the drawer. An about page and a button to get back to the statistics page is also included in the navigation drawer. The drawer is implemented with an android widget.



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout  xmlns:android="http://schemas.android.com/apk/res/andro
               xmlns:app="http://schemas.android.com/apk/res-auto"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"

    >
    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:minHeight="?android:attr/actionBarSize"
        android:background="#641288"
        app:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
        app:popupTheme="@style/ThemeOverlay.AppCompat.Dark"/>

    <android.support.v4.widget.DrawerLayout
        android:id="@+id/drawer_Layout"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <ScrollView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content">
        <!-- the main content view -->
        <LinearLayout

            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation = "vertical">
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:id="@+id/Gamestatlabel"
                android:text="Game Stats"
                android:clickable="false"
                android:editable="false"
                />
            <ListView
                    android:minWidth="25px"
                    android:minHeight="25px"
                    android:layout_width="match_parent"
                    android:layout_height="200dp"
                    android:id="@+id/GamestatListView"
                    android:isScrollContainer="false" />
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:id="@+id/Assiststatlabel"
                android:text="Assists"
                android:clickable="false"
                android:editable="false"
                />
            <ListView
                    android:minWidth="25px"
                    android:minHeight="25px"
                    android:layout_width="match_parent"
                    android:layout_height="200dp"
                    android:id="@+id/AssiststatListView"
                    android:isScrollContainer="false" />
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:id="@+id/Combatstatlabel"
```

The code for the coaching section is very simple as it uses the accord.net framework to implement an ID3 algorithm decision tree. The inputs for the section are built into a data table with the different attributes of the table being the different statists from the averages section (All Damage Done, Barrier Damage Done, Deaths, Eliminations, Final Blows, Healing Done, Hero Damage Done, Objective Kills, Objective Time, Solo Kills, Time Spent on Fire). A SR gain is calculated from the last SR stored and that is used as the label or the Class (the attribute to be predicted). The data is gathered from the file mentioned earlier. The inputs and outputs are then fed into to a tree learning method so a tree can be taught. Once the tree has been taught, it can be queried. An example query would look similar to this {18.25,4.29,4.50, 14.50, 7.95, 1.26, 11.69, 1.77, .26.27.36} This example queries are modelled off of statistics of three different players all from different brackets of SR. This is intended to represent an average baseline, with other queries adjusting the statistics to be higher or lower for different playstyles. If any of the example queries come back positive, then that's the play style that is suggested to the user. If more than one query comes back as a positive more in-depth coaching can be done. Due to time constraints the tree code not be fully implemented in the project, impacting the overall success.



```
//create the ID3 learning algortihm
ID3Learning teacher = new ID3Learning();
//learn a tree for the probelm

var tree = teacher.Learn(inputs, outputs);
//compute error
double error = new ZeroOneLoss(outputs).Loss(tree.Decide(inputs));
//aggressive statistics
int[] AggressiveQuery =
{

};
int[] deffensiveQuery =
{

};
int[] passiveQuery =
{

};

int AggersivePredicted = tree.Decide(AggressiveQuery);
int deffensivePredicted = tree.Decide(deffensiveQuery);
int passivePredicted = tree.Decide(passiveQuery);

if (AggersivePredicted == 1)
{
    string[] Prediction = {
        "Play more Aggersive" };

    ArrayAdapter<String> itemsAdapter =
    new ArrayAdapter<String>(this, Android.Resource.Layout.SimpleListItem1, Prediction);

    CoachList.Adapter = itemsAdapter;
}
else if (deffensivePredicted == 1)
{
    string[] Prediction = {
        "Play more Defensive" };

    ArrayAdapter<String> itemsAdapter =
    new ArrayAdapter<String>(this, Android.Resource.Layout.SimpleListItem1, Prediction);

    CoachList.Adapter = itemsAdapter;
}
else if (passivePredicted == 1)
{
    string[] Prediction = {
        "Play more Passively" };

    ArrayAdapter<String> itemsAdapter =
    new ArrayAdapter<String>(this, Android.Resource.Layout.SimpleListItem1, Prediction);

    CoachList.Adapter = itemsAdapter;
}
else if (deffensivePredicted == 1 && AggersivePredicted == 1)
{
    string[] Prediction = {
        "Play Aggersive but be wary of postioning" };

    ArrayAdapter<String> itemsAdapter
```

Testing for these systems was done while the product was being developed, as per the agile methodology. For the login page, multiple battle net ID's were used to test if the system could access the different statistics page. Entries into the text field that would cause an error were tested as well. For example, a name without the corresponding id number, would return a format error. An empty field would return a null error. An entry using a hashtag symbol rather than a dash would return a format error. Any errors that were found were handled with android's toast message system that displays a message at the bottom of the screen. The statistics page was tested by making sure that there were no errors happening from changing the page or activity as there is no user input in this section. An error was found, when changing from the about page back to the statistic page because the user's battle net id would be reset. This was remedied by passing an instance of the battle net id between the activities whenever it happened.

The decision tree section was tested by using an account to build a test data set and use predetermined queries applied against it. A problem found in testing is that if the person does not have a substantial amount of play time the average section was incomplete, or the data pulled was not enough to make an accurate prediction. To remedy this some statistics were added to the data inputs for the decision tree learning and others removed that were not adding any value. The statistics that were removed were objective time per 10 min, solo kills per 10 min, time on fire per 10 min and objective kills per 10 min. The statistics added were an elimination to death ratio and damage done to death ratio. This was intended to help give the tree learning some higher quality attributes to use for prediction.

Product Evaluation

Here is where the product itself is evaluated. This was done by comparing it to the original vision of the product, comparing to the product specification, and evaluating the performance of the product as felt by the users.

The original vision for this project was to design an easy to use personal coaching and statistics app for individual Overwatch players using machine learning and web scraping technologies to gather data to evaluate on that data to help a player get better at the game. Evaluating the finished product against this initial vision, the product is successful. The web scraping technologies were successfully implemented using the HTML agility pack framework, allowing a user to gather their data and quickly view their statistics on their mobile device. The machine learning aspect was implemented as well. Using the accord.net framework, an ID3 algorithm, a decision tree can successfully be built from a user's data. The tree is built on a player by player case, offering a personalized solution and more tailored advice. The coaching aspect did not mature as expected. A more natural language-based interface was intended, to offer the illusion of having an actual coach. Due to the limitations of the data available from the player lookup tool, advice that was to be given to players was not as detailed as planned. A more in-depth pool of statistics to pull from would offer better advice given to players. For example, area of death, general positioning statistics and a better averaging system. Compared to the original version, the depth and amount of coaching was not attained.

Evaluating against the project specification, there are a few things that were not successful but a lot of positive outcomes as well. In the project specification it is originally stated that the mobile application, to be developed in android, would use java as a coding language. During the project development, Xamarin was found as an alternative development language. Xamarin was a better option as it allowed for development of the back-end (machine learning and web scraping) code in c# and the front-end (UI) in XML. C# was more familiar and also has a lot of Libraries to help with finding solutions to problems. XML is similar to HTML and so was very intuitive to develop with. C# is also said to increase performance of certain tasks in the application. Although with the scale of this application, it may have not been an issue had it been developed in Java. Java is android native programming language. This means that there was a lot of support for developing application for android but if the application intended to be ported to other systems not running android it would have to be built again from the ground up. Xamarin allows for easy code redistribution, making it a better fit for multi-platform development.

The project specification stated that a SQL database was going to be used for storing the gathered data. In retrospect this was not a good idea as the amount of data that was going to be sent back and forth may have become a problem and slowed down the performance of the application. Instead a local file system was used, storing the data in a local file allowed for faster access and didn't impact the performance.

It also stated that the player would be able to look up their statistics that are available and get personalised feedback. Players are able to look up their competitive statistics and get a personalised radar graph to help represent the data shown. So, in that aspect the product is successful, although the amount of data pulled from the site is not the complete available data set. The casual data set was not selected as it was not needed for the prediction of result. Unnecessary processing would have taken place, leading to a slower performing application. A complete data set of all statistics would be helpful for players that do not want to play competitively.

"As the player uses the app more the feedback gets more in depth as it learns how the player plays." the application was successful in this as the implemented machine learning method, a decision tree is dynamically generated when ever a user launches the app. The prediction in theory gets better with more training data to build a decision tree, which leads to a better prediction rate. The original vision was that the system would learn what statistics lead to certain play styles, which in turn lead to a win. A data clustering method, used in tandem with the decision tree method, may have been able to accomplish this. Due to the time constraints of the project, a more uninvolved solution was constructed. Due to the time constraints of the project the machine learning method could not be implemented properly. To allow users a view of the available coaching, the coaching methods were given to them so that they could give feedback on the advice.

Users of the application were asked to fill out a short questionnaire after using the application for any amount of time. This was intended to get a feel for how successful the players felt using the application, and to see if they had any input for improvements. The results of the survey are located in the appendices. Users that replied to the survey were 2 silver player, 4 gold players, 2 platinum players and 1 master player. This gave a good range of player skill levels to evaluate the performance on. Of the ten users who submitted a response, 80% said that they liked the statistics page as it is. The two users that submitted negative responses commented on what they would like to see, one saying they would like to see casual statistics as well as competitive. The other saying, they wanted more unique statistics. Better data manipulation could lead to more unique statistics and casual statistics could mean more people use the application. Lower ranked people tended to answer that they only enjoyed the statistics area of the application so adding in the casual statistics would benefit them. Of the ten players 60% of players answered that they enjoyed the coaching system. The biggest complaint was that the advice given by the machine learning algorithm was not helpful or in depth enough. The master's player (the highest ranked player that submitted results) said that the coaching felt too basic for his SR level. This really was unavoidable with the amount of statistics that Blizzard provides on the player lookup tool. More statistics could offer a better analysis of a players play style, leading to better advice. Only 50% of users felt the coaching experience Would help their personal play. This answer correlates with the time spent using the application where players who use the app for longer felt it did while player who hadn't felt it didn't. This is understandable as a larger training data set for the decision tree would lead to better results, but the less users used the app the less data entries were created. This could be remedied by adding a minimum amount of games to be played before coaching can begin, allowing for a suitable dataset to be created.

Project Evaluation

This section explores the success of the project as a whole. Evidence of the project management techniques will be elaborated on. Reference to the original aim and objectives of the project will be made and assessed on how well they have been addressed. Problems and issues will be given to give a balanced evaluation. Finally, the successes of the project overall and of the project management system will be evaluated.

The project management techniques included using Trello, an online website that allows for a list making along with the agile development methodology and Scrum. A sort of virtual post-it note holder. Trello allows a user, using the agile product design methodology, to efficiently keep track of what tasks need to be done, are in the process of being completed, and that are finished. Screenshots located in the appendices, show how they Trello was used on this project. Trello was used in both the product design and for managing the project as a whole. Product tasks were split into individual tasks. For example, "create the User interface login", and "code the login function." The project was also broken down into smaller and more manageable tasks. An example being, "write introduction" and "write literature review." This filled the Trello board up, and could sometimes get cluttered, so a colour coordinated systems was employed to separate all the tasks into their respective areas. Once a task had been completed it would be updated on the Trello board. If it hadn't been updated in real time, it would be updated at the end of a sprint of the agile methodology. The agile method works well in tandem with Trello but the strengths of agile cannot be fully utilised due to the limited functionality of Trello.

Trello is a very good tool for a single person development team to use and worked well for the development of this product and project. A more in-depth management system like taiga.io would help with the more technical aspects of the agile development methodology, like calculating sprint performance. Taiga.io has built in systems for calculating the performance but Trello is a much simpler and faster system to use, so was preferred for this project. Trello working in along with the Agile method worked very well for this project.

There were multiple methodologies used for both the project and the product development. The project as a whole was developed using with the agile method and Scrum method. While the product was developed with multiple methods hybridized into one, the agile method as a project management methodology worked well as it was used at the same time while developing the product. Allowing the project to be broken down into smaller task, that could be incrementally updated through out the project. At the end of a week the tasks were reflected on as per the scrum method, to get an idea of the speed of development.

Using the multiple methodologies got very cumbersome and overbearing. Focusing on one method may have improved the productivity of the project, allowing for more development and research time. The multiple methodologies did allow a lot of freedom in the development of the project as it allowed identification of the attributes of the methodologies that suited best.

A Gantt chart (located in appendices) was used at the beginning of the project to lay out an overall timeframe. The Gantt chart was very useful and helped keep the project on track overall. A more involved chart that was updated every week at the end of a sprint would have been beneficial to the overall development of the project and the product.

Looking back at the original aim of this project was, “To design and build a system to assist and guide players of Overwatch, using machine learning techniques and data analysis algorithms.” Most of that was achieved. A system was built to assist and guide players. It did use machine learning techniques but there was no data analysis in the final product. Data was manipulated before it was fed into the machine learning algorithm, but no explicit analysis was done. Also, the ability to “assist and guide players” is limited due to the fact that the data that can currently, at time of writing, be used are themselves limited. Contextual statistics and better/more statistics overall could offer better usage and results. The machine learning techniques employed in this project are very surface level and could be interchanged with others easily. A more involved system or a different technique may result in better or more accurate results but may also require more time for training and testing.

The original objectives were:

- To critically review relevant literature
- Design a product based on the research
- Use a relevant design methodology
- Test and evaluate my product internally
- Give the software to players to evaluate the success of the product

All of these objectives were successfully achieved. Relevant literature to the classification problem of predicting a result was conducted successfully. Multiple pieces of literature and academic journals were consulted offering different solutions to the classification problem, namely Decision trees and Artificial neural networks. A project was designed and successfully deployed based on the research conducted. A decision tree based on the research conducted was applied to the data gathered about players, allowing for a prediction to be made.

Multiple design methodologies where used in this project, in an informal implementation. Allowing for flexibility when developing this project. The product was successfully evaluated and tested internally before it was given to users, ensuring a good end user experience. The software was given to users at the end of development to test the successfulness and quality of the product, this was done by a quick survey. So overall, the project successfully achieved these goals set out at the beginning of the project.

Some issues encountered during the project included finding suitable technology to be utilised and choosing a development platform to use. A lot of time was expended finding technologies that could be utilised in this project. Issues were that a lot of the technologies used such as the web scraping API, were found after a long time spent researching options leading to time missed in the development stage. Many different technologies were found but a lot of them required a significant cost or a subscription fee to be used commercially. These larger API’s also came with a lot of extra features that were not necessary in the design scope in the final project and would take up disk space. The longer search led to more trimmed down API’s at the expense of time. Also, the wealth of development platforms led to a long time taken to choose a suitable one. So, most of the issues encountered on this project were based in the fact that they used a lot of time. something that could have been remedied with more research into the related technologies before the project had begun.

Summary and Conclusions

In conclusion, the overall project was a success. An application was built and deployed successfully and allowed player of overwatch to progress. The amount and to what degree they progressed was not the desired outcome but still an acceptable one. Due to the fact that the machine learning implementation was not as effective as hoped, A more in-depth system, as stated before would have satisfied they goals set out at the beginning of the project to a higher degree.

Summarizing the research done into relevant literature, there is a lot of evidence that sports prediction is possible and accurate to a high degree. From the research conducted the key points for developing a successful application are the type of data gathered (the amount and quality of the data) and the classification model used (i.e. Artificial neural networks or decision trees).

From this project we can conclude that the data gathered has huge impact on the outcome of the prediction and the quality of the prediction, as noisy attributes in a data set will stop the model from being built and trained well. Another aspect is choosing the right model to use in your application. An Id3 decision tree was used in this project due to technology constraints, but more applicable models may be a better fit for other applications. The conclusion that can be drawn from the literature review are that sports prediction is a well-established field with lots of resources available and past examples into traditional sports. Transitioning these resources to an Esports setting is more difficult than for example, translating a method used on a football game to a basketball game. An expert in the field is needed to make sure that any data used in the predictive modelling is relevant and necessary. Better quality data will produce better quality results, no matter the classification model used.

The technologies used in this project were chosen based on experience of the systems, applicability to the scope, cost and time constraints of the project. The development platform used, Xamarin was used due to its advantage of being able to redeploy to multiple operating systems in addition to the added advantage of being able to develop the application in c#. Other development suites are available for developing applications for android and more systems but Xamarin was the most applicable for this project due to the advantages offered. The two code packs or API's used in this project were the two most useful found at the time of development. Other similar packs were available but the HTML agility pack and accord.net offered lightweight solutions compared to some larger more cumbersome packs. A more robust and varied web scraping API would offer better assurances against changes or errors that might occur but may also come with a lot of extra and unnecessary code for the purpose of the project. The accord.net machine learning framework was chosen as it offered an id3 decision tree, with an easy and lightweight implementation. Accord.net also offers other machine learning methods and data analytic tools so could be useful in a number of different applications and projects.

The conclusion that can be drawn from the development stage of the product and the development of the project as a whole are that, as a singular developer the Rapid application design and agile informal hybrid method works very well. This is because this combination it gives a high degree of freedom to develop in your own time and does not adhere to any strict deadlines. In reflection, for a project like this, where the end goal is very clearly designed product. A traditional approach such as a waterfall method may have been more applicable. Comparing the project with a client every week may have been counterproductive to project as a whole. A clearer and more direct approach may have helped produce a working project quicker which in turn could have been improved and finer tuned in the given time frame.

From both the technology and research sections the recommendations for future projects would be:

- to make sure the data used is useful and not noisy (have a lot of outliers or empty attributes)

- to have an expert in the field your building the project for give pointers on which data points to use.
- to examine multiple different machine learning techniques and to find the one that offers the best results.

Depending on the game the application is being built for, the developing company may offer an API to collect and gather statistics, so if there is one, a web scraper is not necessary. If a Web scraper is to be used, using a lightweight API would be best or build a purpose built Webscraper for the application. Due to the fact that the website data is gathered from may go down or may not be available, a backup of useful data would help with the user experience of the application, by allowing offline access.

For future work, an ideal situation would be that Blizzard has developed their own API (application programming interface) to relay data and more accurate statistics. This would allow anyone building a similar application to have better quality data to feed the machine learning method of choice. For applying to other games, research into how data would be acquired would be needed. If no API is available, a web scraper will need to be employed, such as the Html agility pack. Again, research should be carried out to determine if the web scraper chosen has the right applications for the project.

Future applications that build on this project should attempt to make the coaching system more robust and effective. The problems encountered during this project were that the coaching system was very one dimensional, in the fact that pre-determined submissions were being made to the decision tree. These predetermined submissions were built from averages of multiple players from all different player brackets but were not indicative of what a player should play like at their level of play. So, to further the machine learning coaching aspect of this project, more data analysis could be carried out. Applying other machine learning methods, such as clustering could be used to find correlations between a player or players skill bracket and the statistics associated with them. Thus, building a better player profile to submit to a user's machine learning method.

Bibliography

Carpenter, N. (2018). *Overwatch: Number of Players Worldwide | Dot Esports*. [online] Dot Esports. Available at: https://dotesports.com/news/overwatch-player-count

Codington-Lacerte, C. (2018) ‘*Agile software development’*, *Salem Press Encyclopaedia*. Available at: https://go.openathens.net/redirector/leedsmet.ac.uk?url=http%3a%2f%2fsearch.ebscohost.com%2flogin.aspx%3fdirect%3dtrue%26db%3ders%26AN%3d90558239%26site%3deds-live%26scope%3dsite

Ian, G. Patrick, M. and Nicolas, P. (2018) *Making machine learning robust against adversarial inputs.*

Michael C. Purucker (1996) *Neural network Quarterbacking* IEEE Potentials, Potentials, IEEE, (3), p. 9. doi: 10.1109/45.535226.

Molina, B. (2018) *Why watch other people play video games? What you need to know about esports [online]* Available *at:* https://eu.usatoday.com/story/tech/news/2018/01/12/more-people-watch-esports-than-x-dont-get-here-basics/1017054001/

R.P. Bunker, F. Thabtah (2017) *A machine learning framework for sport result prediction*.

Tang, X. Liu, Z. Li, T. Wu, W. Zhenhua W. (2018) *The Application of Decision Tree in the Prediction of Winning Team*

The Telegraph (2017) *What are esports? | A beginner's guide* [Online]

Ukie (2018) *The games industry in numbers [online]* Available at: http://ukie.org.uk/research

Vincenzo Milella (2019). *Overwatch Competitive Rank Distribution: PC and Console - March 2019*. [online] Available at: https://www.esportstales.com/overwatch/competitive-rank-distribution-pc-and-console

Appendices

Appendix A: design documents

Fig 2. Wire frames created before user interviews

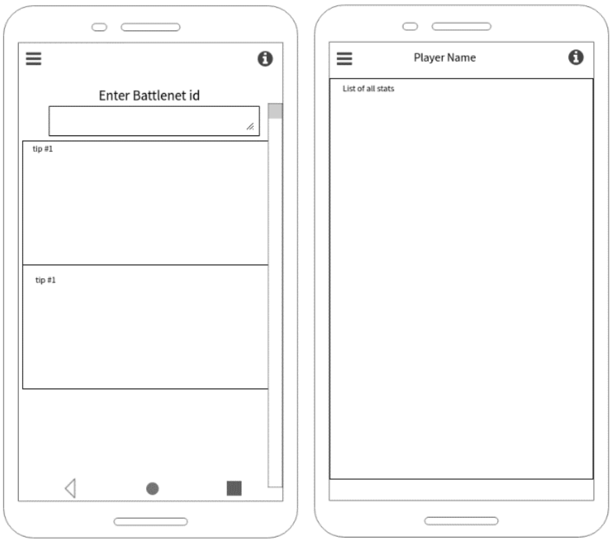
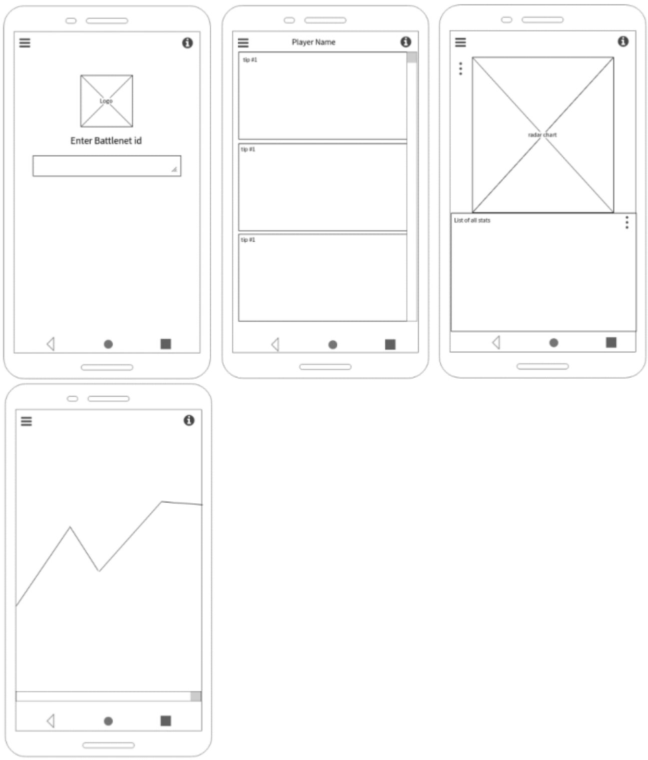


Fig. 3 : Wire frames created after user interviews:



The main differences are the new login screen, a dedicated screen for the coaching area and the inclusion of the radar chart. The addition of the performance page is also different to the initial designs.

Appendix B: interview transcripts and product Testing

Two transcripts of interviews conducted with players of overwatch, on what they would like to see in the application at beginning of project:

Conducted 15-dec-1018.

Interviewer: Hello, are you okay with me recording this interview.

Interview subject: yes of course.

Interviewer: so, if you were to have a coaching app for overwatch what key features would you like?

Interview subject: The ability to see my past performance along with my current stats.

Interviewer: Anything else?

Interview subject: I would want to compare myself with other players, to see how good I'm doing.

Interviewer: okay cool, any other specifications or requirements?

Interview subject: No, that's it.

Interviewer: Alright, thanks for your time.

Conducted 17-dec-1018.

Interviewer: Hello, are you okay with me recording this interview.

Interview subject: yep, no problem.

Interviewer: So, if you were to have a coaching app for overwatch what key features would you like to see?

Interview subject: I would like to be able to see my current stats.

Interviewer: Anything else?

Interview subject: If the coaching aspect is in depth, an area to get coaching information and pointers by itself.

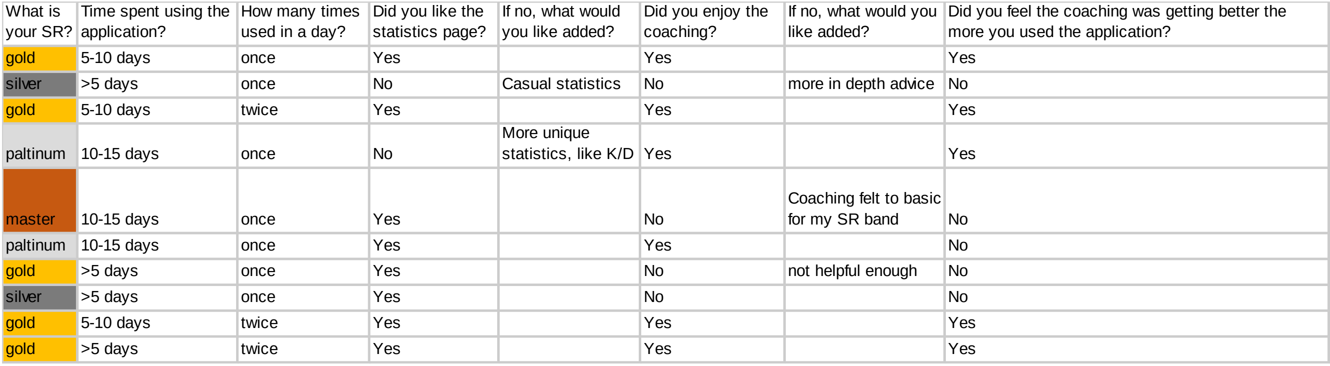
Interviewer: by itself, why?

Interview subject: so that the coaching I'm getting doesn't get hidden by other aspects on screen.

Interviewer: Alright, thanks for your time.

Interview subject: no problem.

Fig 4. Results of product evaluation survey:



What is your SR?	Time spent using the application?	How many times used in a day?	Did you like the statistics page?	If no, what would you like added?	Did you enjoy the coaching?	If no, what would you like added?	Did you feel the coaching was getting better the more you used the application?
gold	5-10 days	once	Yes		Yes		Yes
silver	>5 days	once	No	Casual statistics	No	more in depth advice	No
gold	5-10 days	twice	Yes		Yes		Yes
paltinum	10-15 days	once	No	More unique statistics, like K/D	Yes		Yes
master	10-15 days	once	Yes		No	Coaching felt to basic for my SR band	No
paltinum	10-15 days	once	Yes		Yes		No
gold	>5 days	once	Yes		No	not helpful enough	No
silver	>5 days	once	Yes		No		No
gold	5-10 days	twice	Yes		Yes		Yes
gold	>5 days	twice	Yes		Yes		Yes

Appendix C: Project management

School of Computing, Creative Technologies and Engineering

Level 6 Production Project

MEETING RECORD SHEET:	**Meeting Number:1**
Student: Andrew Heath	**Student I.D.: c3486241**
Date of Meeting:04/10/2018	**Supervisor: Duncan Mullier**

Actions agreed at previous meeting (completed or comment):

6 □

Comments of student (if any):

…This was our first meeting and so only was able to give my brief plan. Also, I was able to give my thought on how the project should proceed. After doing some research into machine learning I found that it was probably a better idea to take aspects and techniques used in machine learning to help inform my project rather than using a machine learning software.

***ABOVE here** – student to complete before Meeting with supervisor. **BELOW here** – complete at the Meeting.*

Next meeting (date/time)**:**

Agreed Actions to complete before next meeting:

1 **Conduct more thorough research into ML and ANN**

2	**Complete wireframes of project**
Comments of supervisor (if any): Good first initial meeting, some more research into how machine learning is going to be incorporated into the project is needed.	

School of Computing, Creative Technologies and Engineering **Level 6 Production Project**	
MEETING RECORD SHEET:	**Meeting Number:2**
Student: Andrew Heath	**Student I.D.: c3486241**
Date of Meeting:30/1/2019	**Supervisor: Duncan Mullier**
Actions agreed at previous meeting (completed or comment):	
1	***Conduct more thorough research into ML and ANN***
2	**Complete wireframes of project** □
Comments of student (if any): After conducting some more thorough research into, I've solidified my path in the machine learning aspect of the project, some basic wireframes have been completed	
ABOVE here *– student to complete before Meeting with supervisor.* ***BELOW here*** *– complete at the Meeting.*	
Next meeting (date/time)**:**	

Agreed Actions to complete before next meeting:	
1	**Choose an IDE**
2	**Progress on report**

Comments of supervisor (if any):

First draft of the report came across as rambling and Didn’t get straight to the point. Progress into the.

School of Computing, Creative Technologies and Engineering

Level 6 Production Project

MEETING RECORD SHEET:	**Meeting Number: 3**
Student: Andrew Heath	**Student I.D.: c3486241**
Date of Meeting: 27/03	**Supervisor: Duncan Mullier**

Actions agreed at previous meeting (completed or comment):	
1	**Choose an IDE**
2	**Progress on report**

Comments of student (if any):

I have chosen an IDE to develop in along with the language changing to C# using Xamarin.

ABOVE here *– student to complete before Meeting with supervisor.* ***BELOW here*** *– complete at the Meeting.*

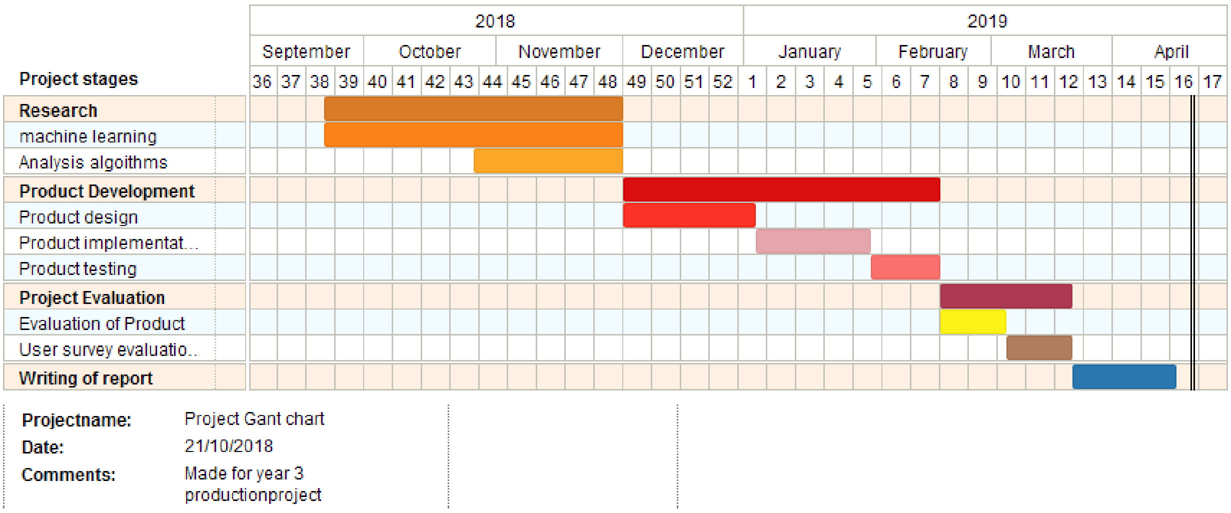
Next meeting (date/time)**:**

Agreed Actions to complete before next meeting:

Comments of supervisor (if any):

Report contents still to unstructured need to be more to the point.

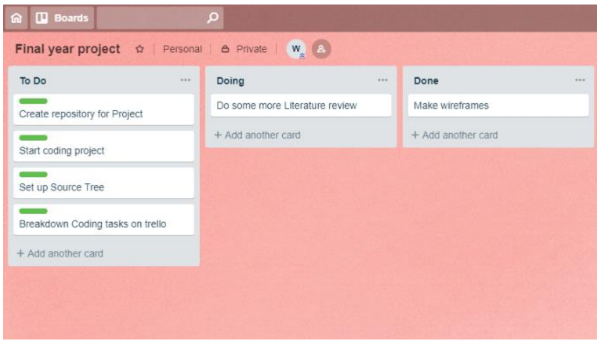
Fig 5. Gantt chart made at beginning of the project:



Projectname: Project Gant chart
Date: 21/10/2018
Comments: Made for year 3 productionproject

Appendix D: Working in a professional manner

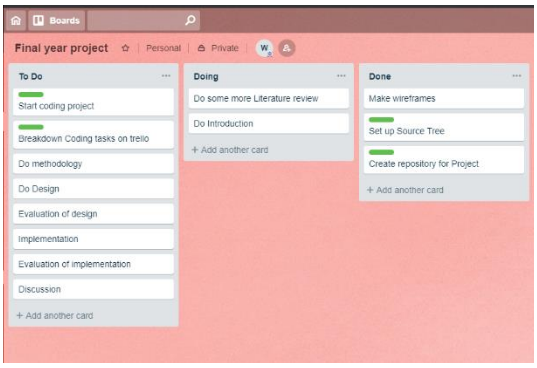
A collection of screenshots, where Trello has been used to organize tasks.



To Do
- Create repository for Project
- Start coding project
- Set up Source Tree
- Breakdown Coding tasks on trello
- + Add another card

Doing
- Do some more Literature review
- + Add another card

Done
- Make wireframes
- + Add another card



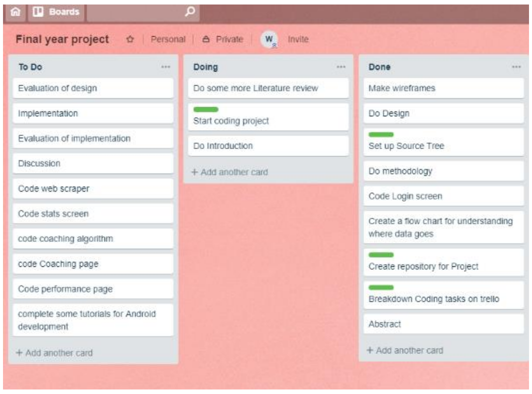
Boards

Final year project ☆ Personal | Private | W

To Do
- Start coding project
- Breakdown Coding tasks on trello
- Do methodology
- Do Design
- Evaluation of design
- Implementation
- Evaluation of implementation
- Discussion
- + Add another card

Doing
- Do some more Literature review
- Do Introduction
- + Add another card

Done
- Make wireframes
- Set up Source Tree
- Create repository for Project
- + Add another card



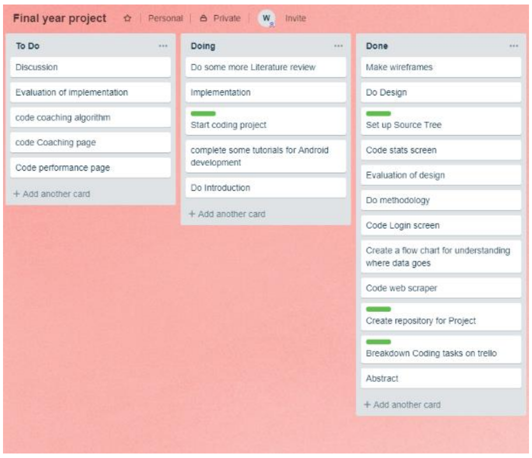
Boards

Final year project ☆ Personal | Private | W Invite

To Do
- Evaluation of design
- Implementation
- Evaluation of implementation
- Discussion
- Code web scraper
- Code stats screen
- code coaching algorithm
- code Coaching page
- Code performance page
- complete some tutorials for Android development
- + Add another card

Doing
- Do some more Literature review
- Start coding project
- Do Introduction
- + Add another card

Done
- Make wireframes
- Do Design
- Set up Source Tree
- Do methodology
- Code Login screen
- Create a flow chart for understanding where data goes
- Create repository for Project
- Breakdown Coding tasks on trello
- Abstract
- + Add another card

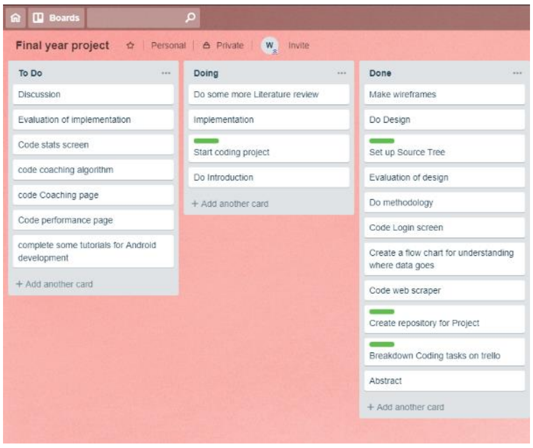


Final year project ☆ Personal | Private | W Invite

To Do
- Discussion
- Evaluation of implementation
- code coaching algorithm
- code Coaching page
- Code performance page
- + Add another card

Doing
- Do some more Literature review
- Implementation
- Start coding project
- complete some tutorials for Android development
- Do Introduction
- + Add another card

Done
- Make wireframes
- Do Design
- Set up Source Tree
- Code stats screen
- Evaluation of design
- Do methodology
- Code Login screen
- Create a flow chart for understanding where data goes
- Code web scraper
- Create repository for Project
- Breakdown Coding tasks on trello
- Abstract
- + Add another card



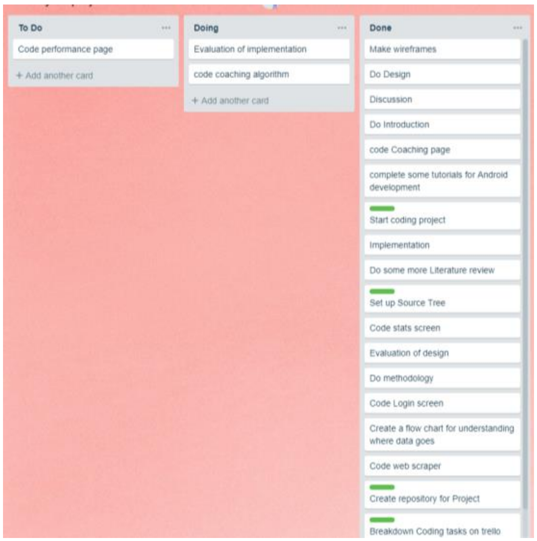
Boards

Final year project ☆ Personal | Private | W Invite

To Do
- Discussion
- Evaluation of implementation
- Code stats screen
- code coaching algorithm
- code Coaching page
- Code performance page
- complete some tutorials for Android development
- + Add another card

Doing
- Do some more Literature review
- Implementation
- Start coding project
- Do Introduction
- + Add another card

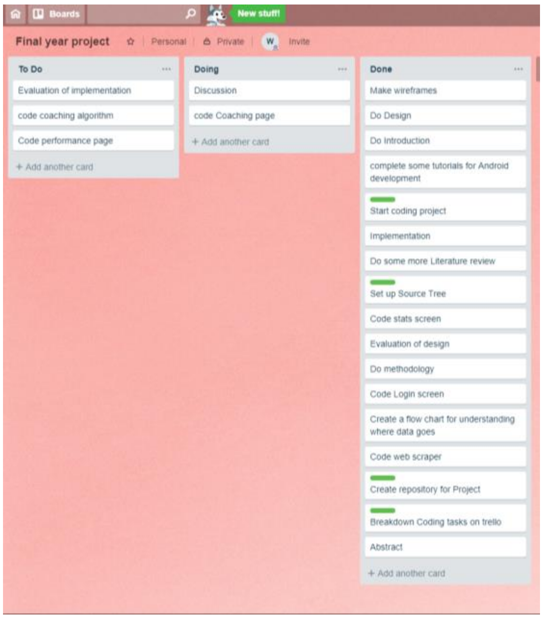
Done
- Make wireframes
- Do Design
- Set up Source Tree
- Evaluation of design
- Do methodology
- Code Login screen
- Create a flow chart for understanding where data goes
- Code web scraper
- Create repository for Project
- Breakdown Coding tasks on trello
- Abstract
- + Add another card



To Do
- Code performance page
- + Add another card

Doing
- Evaluation of implementation
- code coaching algorithm
- + Add another card

Done
- Make wireframes
- Do Design
- Discussion
- Do Introduction
- code Coaching page
- complete some tutorials for Android development
- Start coding project
- Implementation
- Do some more Literature review
- Set up Source Tree
- Code stats screen
- Evaluation of design
- Do methodology
- Code Login screen
- Create a flow chart for understanding where data goes
- Code web scraper
- Create repository for Project
- Breakdown Coding tasks on trello



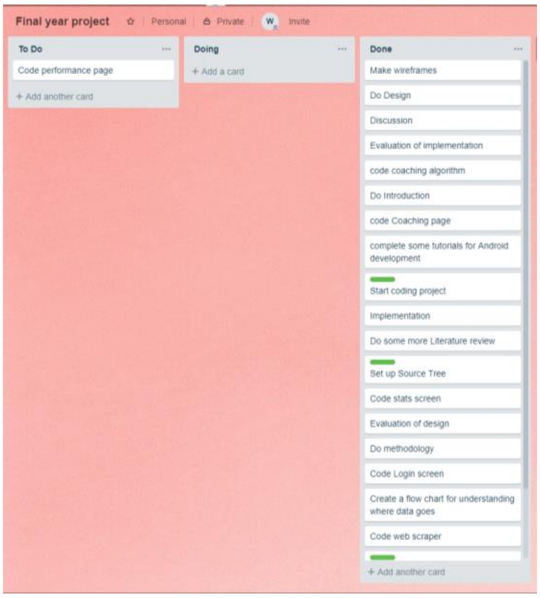
Boards | New stuff!

Final year project ☆ Personal | Private | W Invite

To Do
- Evaluation of implementation
- code coaching algorithm
- Code performance page
- + Add another card

Doing
- Discussion
- code Coaching page
- + Add another card

Done
- Make wireframes
- Do Design
- Do Introduction
- complete some tutorials for Android development
- Start coding project
- Implementation
- Do some more Literature review
- Set up Source Tree
- Code stats screen
- Evaluation of design
- Do methodology
- Code Login screen
- Create a flow chart for understanding where data goes
- Code web scraper
- Create repository for Project
- Breakdown Coding tasks on trello
- Abstract
- + Add another card



To Do

- Code performance page
- + Add another card

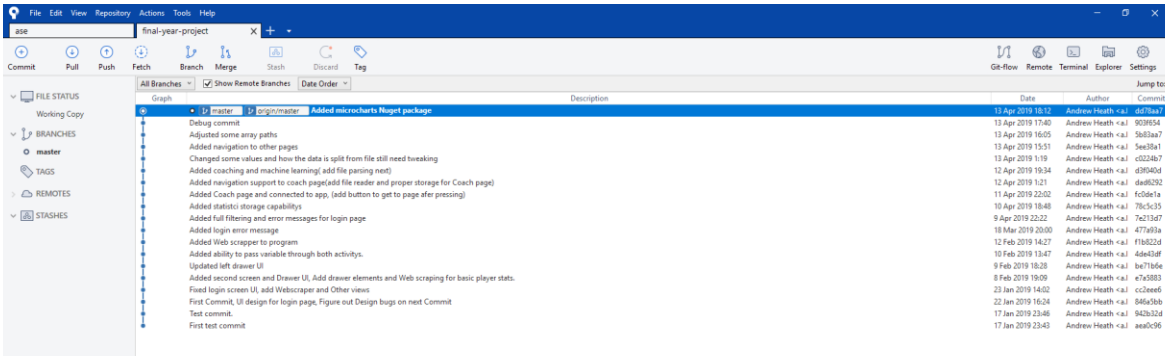
Doing

- + Add a card

Done

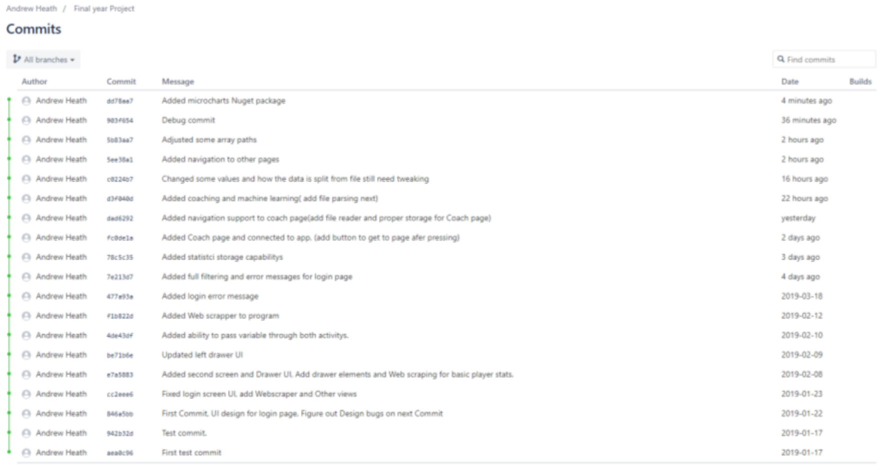
- Make wireframes
- Do Design
- Discussion
- Evaluation of implementation
- code coaching algorithm
- Do Introduction
- code Coaching page
- complete some tutorials for Android development
- Start coding project
- Implementation
- Do some more Literature review
- Set up Source Tree
- Code stats screen
- Evaluation of design
- Do methodology
- Code Login screen
- Create a flow chart for understanding where data goes
- Code web scraper
- + Add another card

Screenshot of commits from Source tree:



Description	Date	Author	Commit
Added microcharts Nuget package	13 Apr 2019 18:12	Andrew Heath <a.l	dd78aa7
Debug commit	13 Apr 2019 17:40	Andrew Heath <a.l	903f654
Adjusted some array paths	13 Apr 2019 16:05	Andrew Heath <a.l	5b83aa7
Added navigation to other pages	13 Apr 2019 15:51	Andrew Heath <a.l	5ee38a1
Changed some values and how the data is split from file still need tweaking	13 Apr 2019 1:19	Andrew Heath <a.l	c0224b7
Added coaching and machine learning(add file parsing next)	12 Apr 2019 19:34	Andrew Heath <a.l	d3f040d
Added navigation support to coach page(add file reader and proper storage for Coach page)	12 Apr 2019 1:21	Andrew Heath <a.l	dad6292
Added Coach page and connected to app, (add button to get to page afer pressing)	11 Apr 2019 22:02	Andrew Heath <a.l	fc0de1a
Added statistci storage capabilitys	10 Apr 2019 18:48	Andrew Heath <a.l	78c5c35
Added full filtering and error messages for login page	9 Apr 2019 22:22	Andrew Heath <a.l	7e213d7
Added login error message	18 Mar 2019 20:00	Andrew Heath <a.l	477a93a
Added Web scrapper to program	12 Feb 2019 14:27	Andrew Heath <a.l	f1b822d
Added ability to pass variable through both activitys.	10 Feb 2019 13:47	Andrew Heath <a.l	4de43df
Updated left drawer UI	9 Feb 2019 18:28	Andrew Heath <a.l	be71b6e
Added second screen and Drawer UI, Add drawer elements and Web scraping for basic player stats.	8 Feb 2019 19:09	Andrew Heath <a.l	e7a5883
Fixed login screen UI, add Webscraper and Other views	23 Jan 2019 14:02	Andrew Heath <a.l	cc2eee6
First Commit, UI design for login page, Figure out Design bugs on next Commit	22 Jan 2019 16:24	Andrew Heath <a.l	846a5bb
Test commit.	17 Jan 2019 23:46	Andrew Heath <a.l	942b32d
First test commit	17 Jan 2019 23:43	Andrew Heath <a.l	aea0c96

Screenshot of commits on bitbucket



Commits

Author	Commit	Message	Date	Builds
Andrew Heath	dd78aa7	Added microcharts Nuget package	4 minutes ago	
Andrew Heath	903f654	Debug commit	36 minutes ago	
Andrew Heath	5b83aa7	Adjusted some array paths	2 hours ago	
Andrew Heath	5ee38a1	Added navigation to other pages	2 hours ago	
Andrew Heath	c0224b7	Changed some values and how the data is split from file still need tweaking	16 hours ago	
Andrew Heath	d3f040d	Added coaching and machine learning(add file parsing next)	22 hours ago	
Andrew Heath	dad6292	Added navigation support to coach page(add file reader and proper storage for Coach page)	yesterday	
Andrew Heath	fc0de1a	Added Coach page and connected to app. (add button to get to page afer pressing)	2 days ago	
Andrew Heath	78c5c35	Added statistci storage capabilitys	3 days ago	
Andrew Heath	7e213d7	Added full filtering and error messages for login page	4 days ago	
Andrew Heath	477a93a	Added login error message	2019-03-18	
Andrew Heath	f1b822d	Added Web scrapper to program	2019-02-12	
Andrew Heath	4de43df	Added ability to pass variable through both activitys.	2019-02-10	
Andrew Heath	be71b6e	Updated left drawer UI	2019-02-09	
Andrew Heath	e7a5883	Added second screen and Drawer UI. Add drawer elements and Web scraping for basic player stats.	2019-02-08	
Andrew Heath	cc2eee6	Fixed login screen UI. add Webscraper and Other views	2019-01-23	
Andrew Heath	846a5bb	First Commit. UI design for login page. Figure out Design bugs on next Commit	2019-01-22	
Andrew Heath	942b32d	Test commit.	2019-01-17	
Andrew Heath	aea0c96	First test commit	2019-01-17	

Appendix E: Project plan

Project aim:

To design and build a system to assist and guide players of Overwatch, using machine learning techniques and data analysis algorithms.

Project Objectives:

The objectives of my project are to critically review relevant research literature. I will design my product informed by the research I have conducted. I will then research development methodologies and choose one to use. I will use this methodology to develop a piece of software. After developing my software, I will test it for bugs and problems and once completed will evaluate my development methodology. Once I think that the software is done I will give the software to players of Overwatch and ask them to fill out an evaluation form of some kind. This will be used to evaluate if a system like this is a benefit to the player.

Research:

The research I will conduct will be looking into the field of machine learning and the various techniques used already. Things such as Topic modelling, unsupervised clustering, and reinforcement learning. These are just some examples of techniques I've found already, and I will use these techniques to inform the development of my product. I will also research into data analysis already in place in relation to the field of sports and statistic prediction. Using a combination of these research topics I hope to produce a helpful piece of software.

Product:

The product to be produced is going to be a mobile application for the android platform. The application will allow players of the game Overwatch to look up their statistics that are available to the public and get personalised feedback on how to better improve their skills and gameplay. This will be done by using machine learning techniques and data analysis algorithms to inform the player on how to get better at the game. As the player uses the app more the feedback gets more in depth as it learns how the player plays. It will be developed in java, with the possibility of an SQL database being used as well.

Project Evaluation:

The evaluation of the product will start with my evaluation of the design methodology I used to develop my product. This will be done by comparing my actual work schedule to my proposed one in the Gantt chart below. After that I will evaluate the product itself to make sure it is usable and is up to the standard I want. After that I will give the product to players of overwatch and ask them to use it. I will develop a questionnaire to determine whether the product has helped the users. I will then evaluate the responses to the questionnaire.

Many of the changes made to the original project plan were due to the time constraints of the project.

Appendix F: user guide

Complete list of statistics available:

Current skill rating(SR) Games Lost, Games Played, Games Won, Time Played, Defensive Assists, Healing Done, Offensive Assists, Recon Assists, All Damage Done, Barrier Damage Done , Damage Done , Deaths ,Eliminations ,Final Blows ,Hero Damage Done ,Melee Final Blows ,Multifils ,Objective Kills ,Objective Time ,Solo Kills ,Time Spent on Fire ,All Damage Done ,Barrier Damage Done ,Deaths ,Eliminations ,Final Blows ,Healing Done ,Hero Damage Done ,Objective Kills ,Objective Time ,Solo Kills ,Time Spent on Fire ,Cards ,Medals ,Medals - gold, Medals - silver, Medals - bronze

User instructions:

1. Download the APK file and install to an android device
2. Once installed launch application and enter battle.net id in the format name-number (e.g. warden-21323)
3. Browse statistics and swipe from the left to open navigation drawer
4. Select the coach tab by clicking on the coach button
5. The coach will now build a new data entry so continue to do this step and the coach will improve over time.

Appendix G: listing for code used

César Souza (2018) **Accord framework** [source code] by available from: <accord-framework.net>

Microsoft corporation (2018) **.NET Core in visual studio** [software] available from <code.visualstudio.com/docs/other/dotnet>

ZZZ projects (2018) **HTML agility pack** [source code] available from: <html-agility-pack.net>

Alois Deniel (2018) **Micro charts** [source code] available from**:** **<github.com/aloisdeniel/Microcharts>**