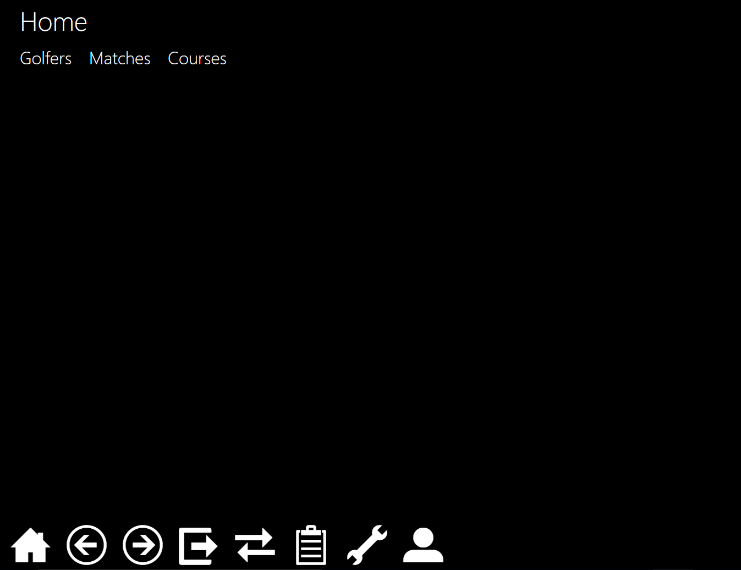
Iterations

Iteration 1: Imagine you are a Golf club owner and you were setting up a tournament. Normally this would be done on paper. This is risky because the paper is so easily damaged and lost. The aim of my first iteration is to be able to load up a tournament template and fill in the players first and last name. Then you enter the score for each hole which is then used to calculate the finally score which then can be ranked. Initially the template is an excel file which is loaded. Eventually I will make the score sheet part of the program and not a separate file. This means that everyone one that is part of the tournament will be able to access the score sheet which will be constantly updated. So far there is two templates one for a 9-hole game and one for a 18-hole game. I would also like it to automatically load the players in the tournament into the score sheet instead of manually having to enter them. The templates are stored on the Github Repository so are easily accessed by a user and come as part of the application.

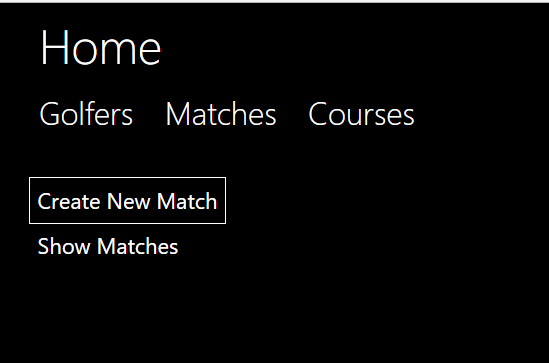
Things to sort out: You can’t create a tournament and add a photo but because you can’t add a photo you can’t create the Tournament meaning it exits with a “'Newtonsoft.Json.Linq.JObject' to type 'Newtonsoft.Json.Linq.JValue'”. To solve this problem I need to work out how to add a file in the initialization of that object . Another problem is that you can’t add players either. Considering these are both action required events that gives a hint at the problem.

Write up:

Step by step guide of how it works:

This is the main menu

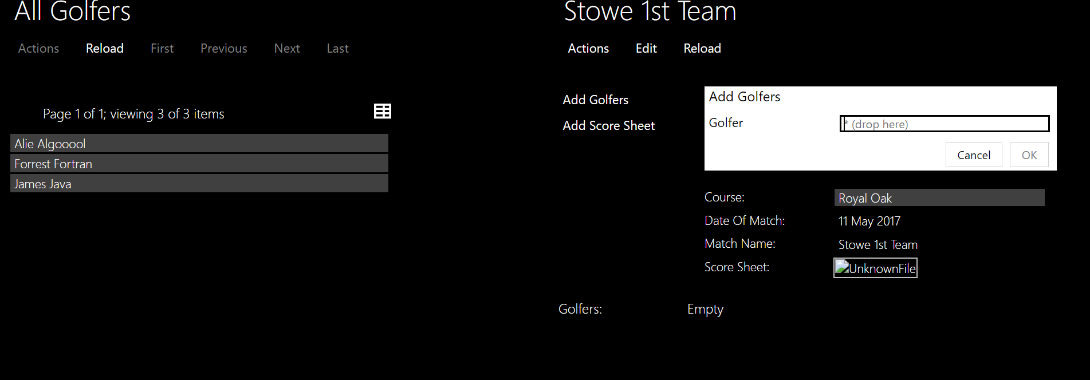
It consists of 3 options. This is different to what was first purposed but when it comes to the application it can be changed. The main aim is ease of use. To create a match you just click on matches. This leads you to the next photo. So far there is two options on matches to complete this iteration. You can either create a new match/ tournament or you can see the existing one.

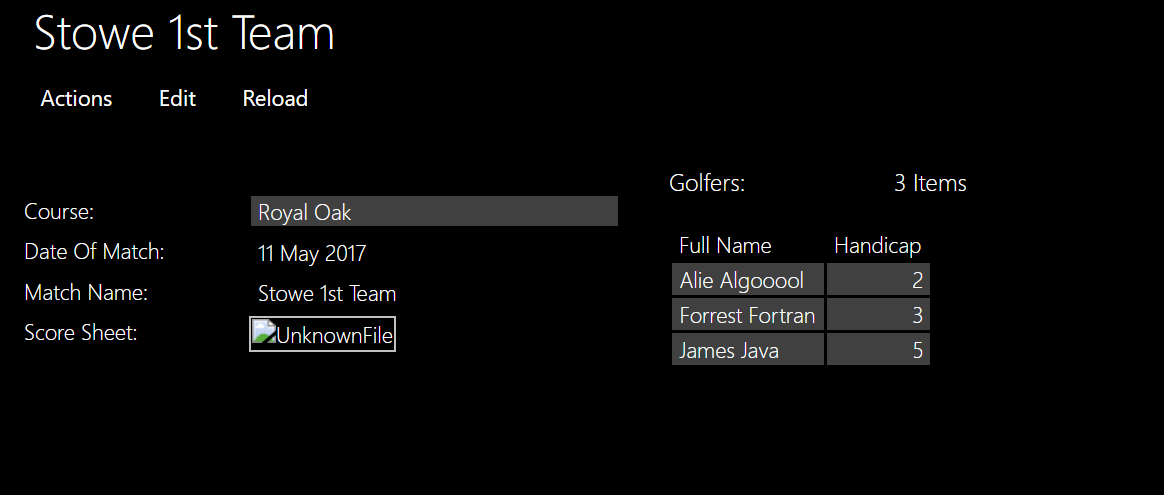


Once show matches is clicked there is a list of matches that have been created in this example there is on Stowe 1st team but many more can be created each match has further detail which has the course name, date of match and match title. Each match also has actions. They’re 2 actions one is add golfers. This is so that you can add your team as a manager to the match.



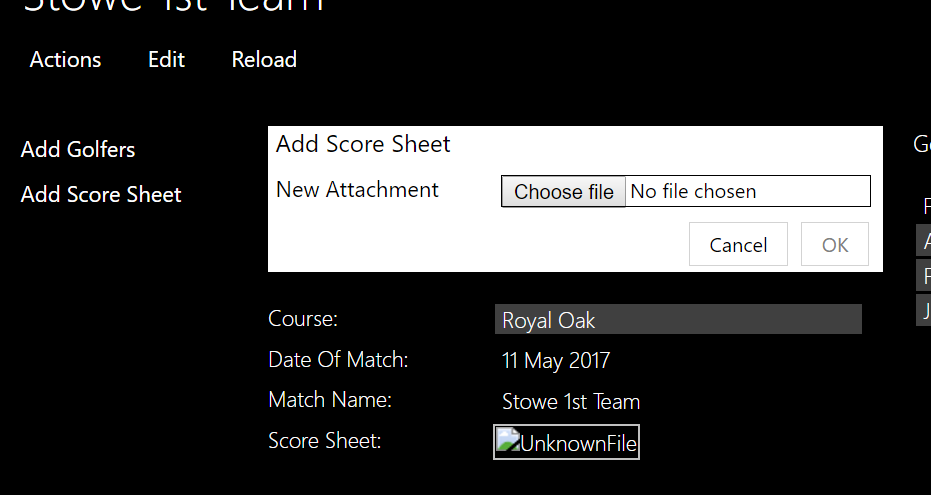
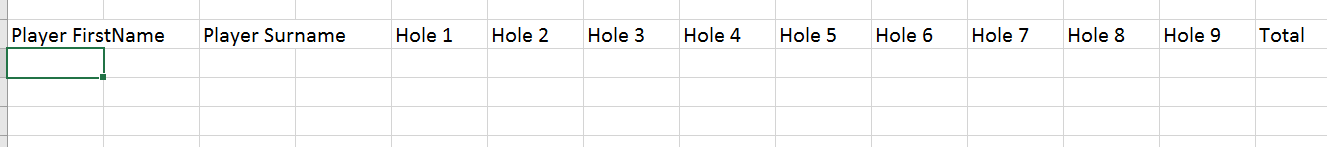
If you chose to add golfer you can split your screen giving half the screen at main options and the other half were u left off this means that you can drag and drop a golfer from the list of golfers that can be edited at any point. This is shown below.

So far you can add as many golfers as you want.

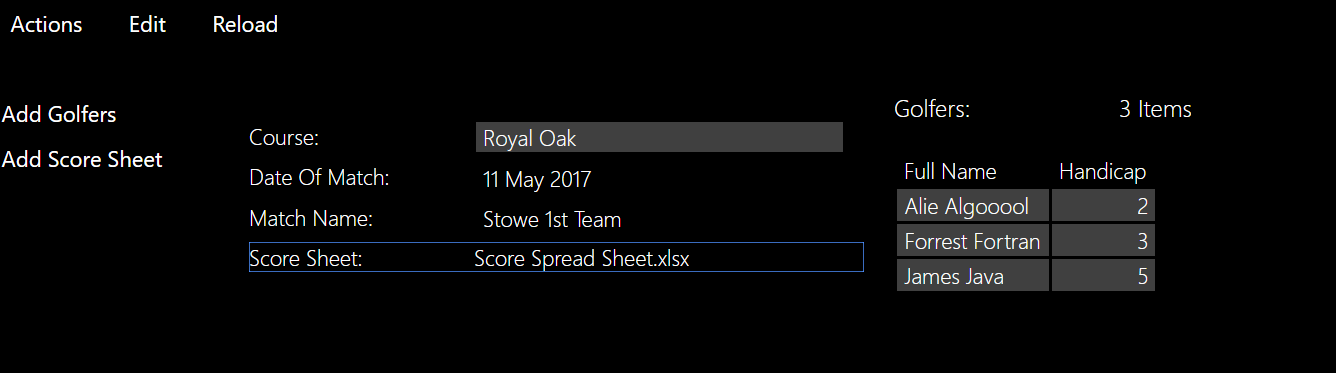


The other action was add score sheet this is done by using a file attachment.

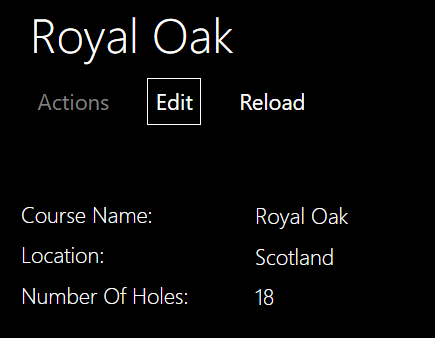
When you click on the file attachment a pop up of a spread sheet comes up which can be edited. In a future iteration the score sheet will be a part of the application rather than a file attachment this means the golfers will be automatically added and so would the number of holes. Also each hole will have its own characteristics meaning that it will have a hardness rating.



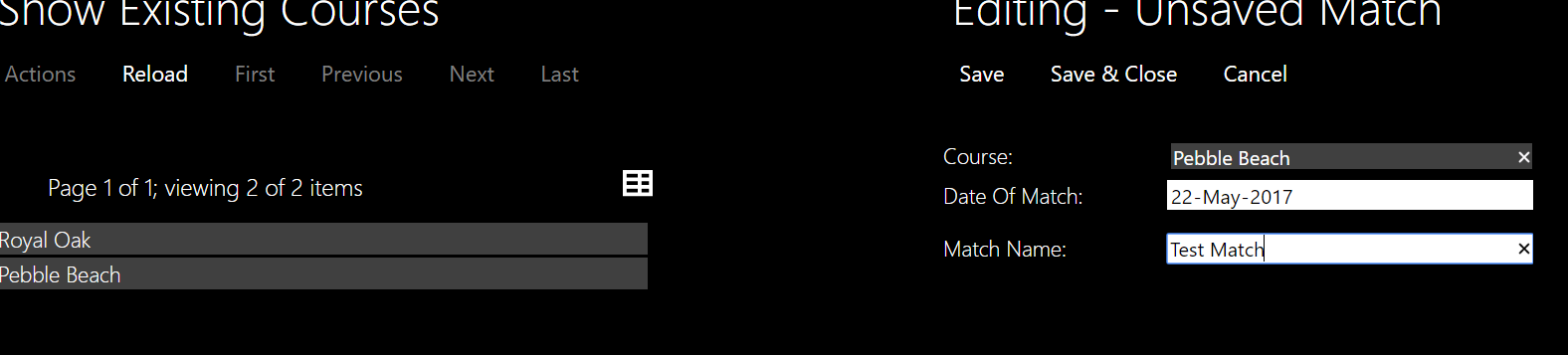
A finished match template looks like :



If you want to know more about the course you can just click on it and another page will pop up with all the detail about that course. This will be used with the scoresheet in further iterations.

This is great for families wanting to come watch cause they can find out where the club is. In future the location will be more advanced with maybe a map showing the location.

Secondly creating a new match is really simple and easy on choosing that option this page pops up:



Once again to add a new course you need to go split screen and drag and drop. The Date is really cool and you just click the day on a calendar that comes up. Once you press save that match will now be on the list for anyone to see and can be edited, golfers can be added and a score sheet can be added.

Iteration 2:

Aim: To create a collection of holes with each course meaning that each course has a certain amount of holes and each hole has a distance, hardness and par. This will become very useful for scoring and will make an important addition to the scoring.



This Screen shot shows that each course now has a number of holes each with a difficulty rating and distance. The difficulty rating will be useful for finding out the scores.

This is done using a collection.

This also means that the Course can be found at the match and therefore the holes can be further easing the next iteration.

Iteration 3:

Aim: To create a scoring system that is more user friendly than the original version and follows the same format as a score card. In order to fulfill this task, I will need to recognize that there is more than one type of match. To get started I will focus on creating a match play format. When I have managed that I will be able to create the other three main formats which is stroke play foursome and Stableford.

Stroke play: This is more frequently used on the pro circuit where the handicaps are usually 0. This is simple since it’s just the number of shots added up and at the end u subtract the handicap.

Match play: For this type of match you travel in a party usually consisting of 2,3 or 4 players. You go around and the lowest score on the hole wins regardless of the amount of shots. If the hole is drawn where everyone has taken the same amount of shots the hole is shared meaning everyone gets the point for that hole who drew In score. This means the handicap is subtracted at each hole rather than at the end. The person with the most holes wins.

Foursome: This is a technique usually used with events for families where you go around in a pair within the party and you take alternate shots. This then can be played in the same way as stroke play, match play or stableford.

Stableford : This has a very different way of scoring and follows the same handicap technique as match play but you score using the following technique.

>1 over fixed score = 0

1 over fixed score = 1

Fixed score = 2

1 under fixed score = 3

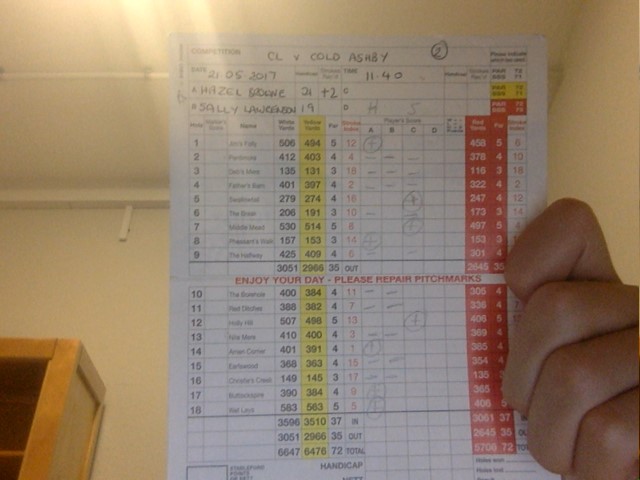
2 under fixed score = 4

3 under fixed score = 5

4 under fixed score = 6

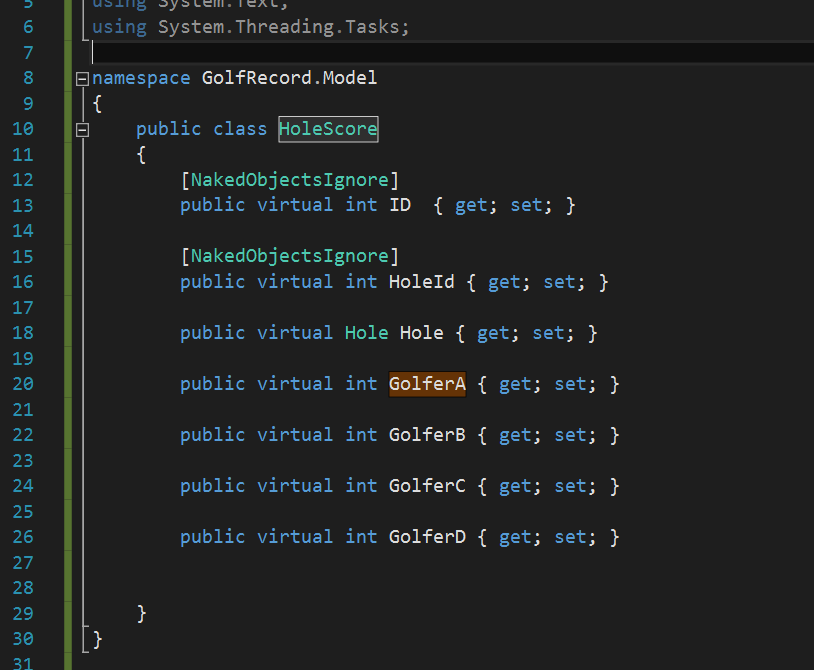
Then the person with the largest score wins.

To make sure that each match is different and has a different scoring section each math will have a type. I have spoked to my client and he has given me a score card of Stowe’s course. This has shown me that a lot more data, including a difference in red yards’ white yards and yellow yards where the stroke index might be different and also the par. Below is two screen shots one showing a blank score sheet and a filled out one.

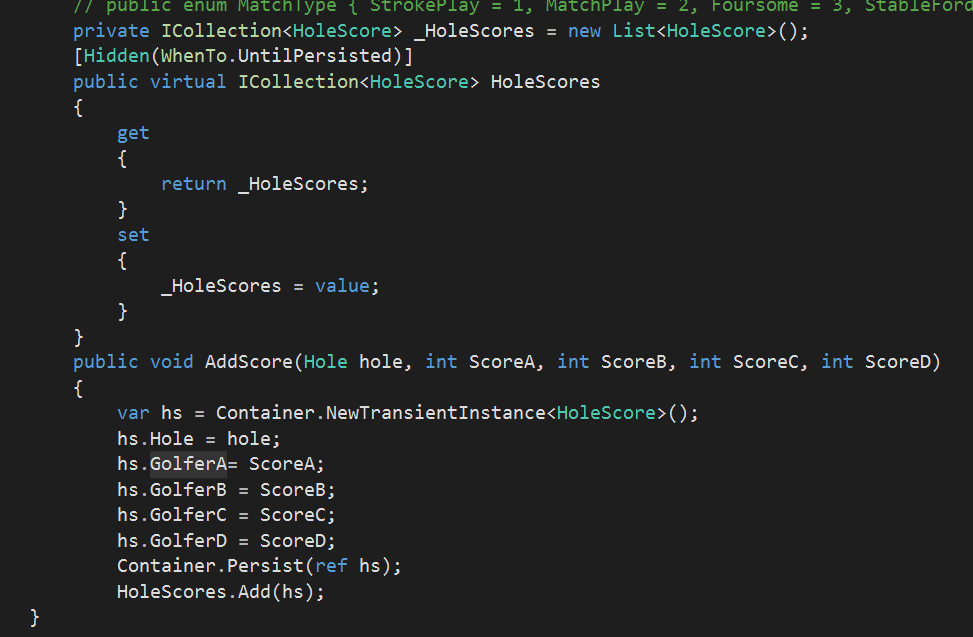
To prevent the golfer from having to calculate who wins the hole they will enter the number of shots that they took on the hole and my program will calculate the winner of that hole and place that on the online card. This will calculate the handicap system. This will require the collection of holes that was created in the iteration above.



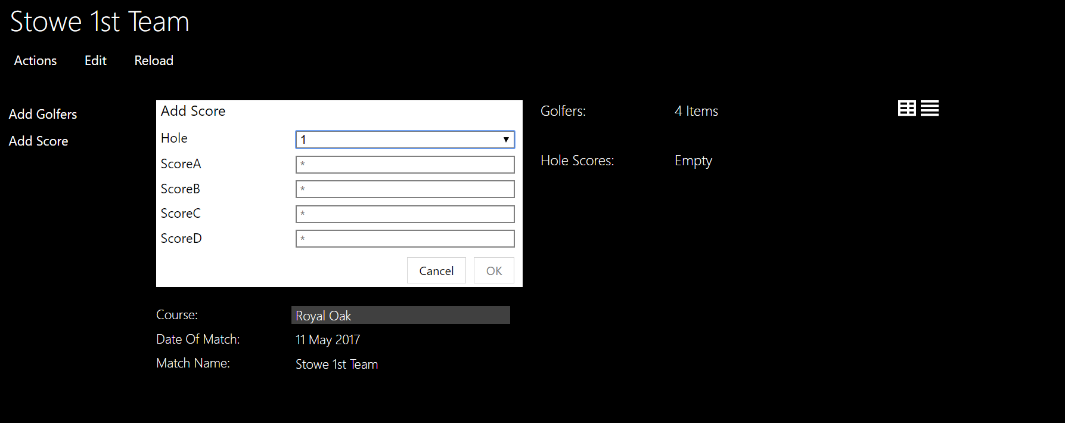
I Started by creating enums which meant that you could only chose certain games modes with a drop down box. I commented them out because they are for future use.



The next move to complete this iteration was to create a HoleScore class which contains 4 golfers and a hole. This means that each hole can have a score and is the basis for each of the match plays. In future, I will make it so that GolferA, B, C, D will be each golfers name so that the user doesn’t get confused. After this I created a ICollection of holes for a match and made it possible for each player to have a score. The reason why GolferA to D is an int is because it is the result that you input.



To make the program more user friendly I created a default score meaning that it will automatically go to the next possible hole although you can change it when you want to skip a hole or something. Below is what the code does.





Iteration 4: To create the same idea used before for Stroke play matches and make it possible for match play matches. This will mean working out an algorithm for each hole.

To start this, I had a meeting with one of my clients involving how the type of match worked.

Facts learnt:

Can’t have 4 or more members in a match.

Males can’t have a handicap above.

This is very valuable information and will have an impact on how to code this problem.

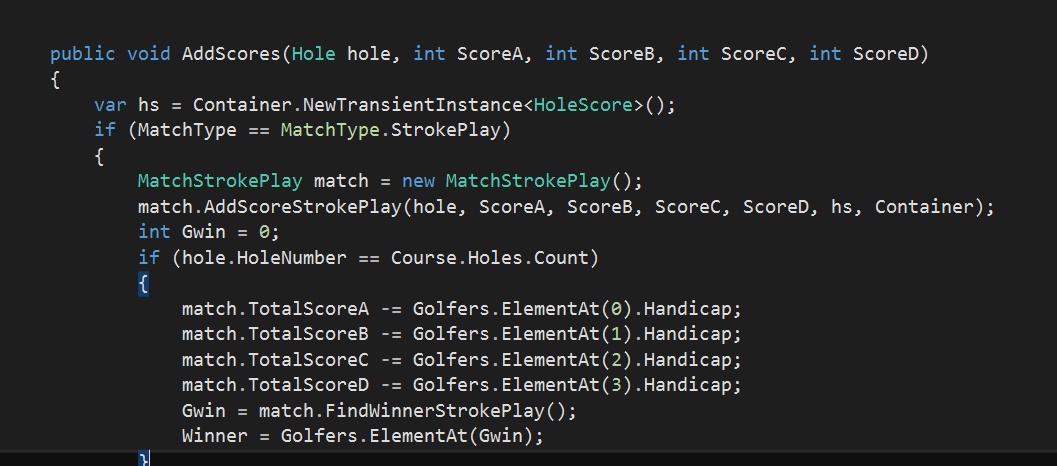
To make my code easier to read I have decided to split up where the code for Stroke Play and the code for the Match play is. This means that it is easier to navigate. I will also be able to show off my understanding of inheritance. I will also use the enums previously created to decide what type of match is being created.

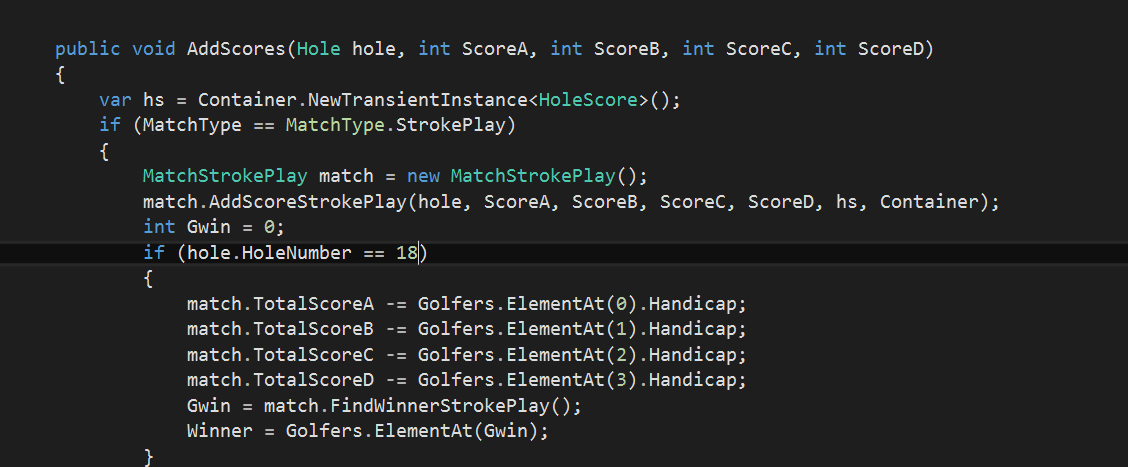
Iteration 4

Aim: To make my entire system more user friendly with various changes. To start I will make the system automatically calculate how many holes need to be played before the end. Have the matches automatically add the winner. Have the girls automatically play off the Ladies Handicap rather the mens since they are different. Add validation for golfers being added to a match since MatchPlay can only be played by 2 players and my current program allows as many people as wanted to join. Finally, I shall aim to add some Email validation as well.

For automatic hole calculation its really simple:

The 18 has just been changed for Course.Holes.Count.





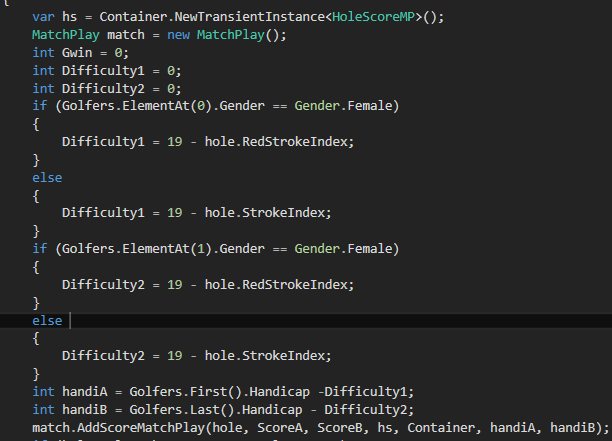
This really small changed allowed me to find the winner after 9 holes as shown below





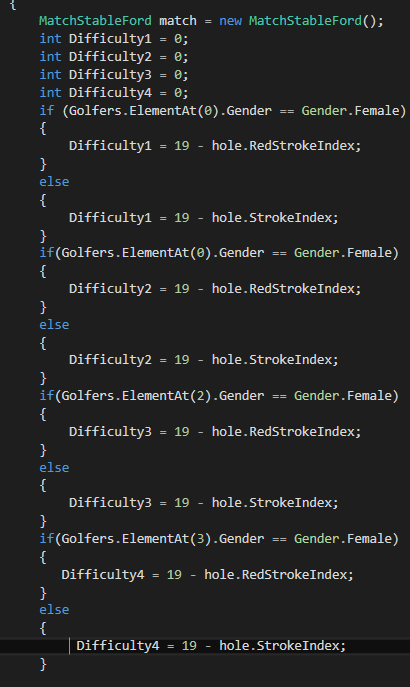
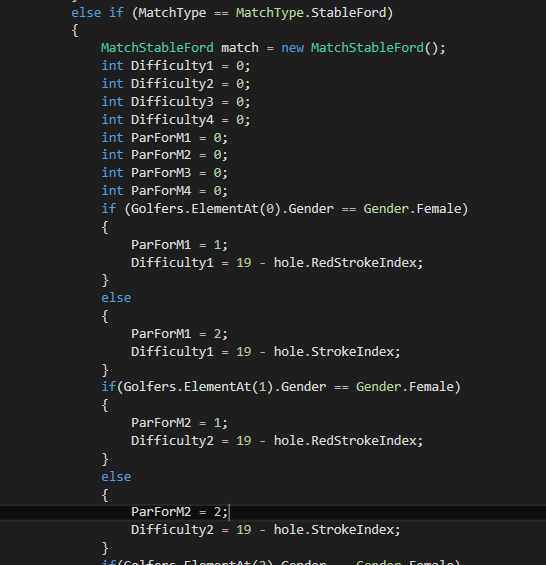
The second little change that was made is that girls have different StrokeIndex to boys this is because they play off different colors. This is also a really simple change.

StrokePlay required no changes since it doesn’t use StrokeIndex but for MAtchPlay I wrote a long winded way in which shall be reviewed at a later stage to see if it can be more efficient and compressed to better code.

As you can see it just goes through each golfer works out whether they are male or female. This works because gender has been made into enum in which a golfer can either be male or female not trying to genderist. The Difficulty follows the same algorithm as before.

Stable Ford is a bit more complicated.

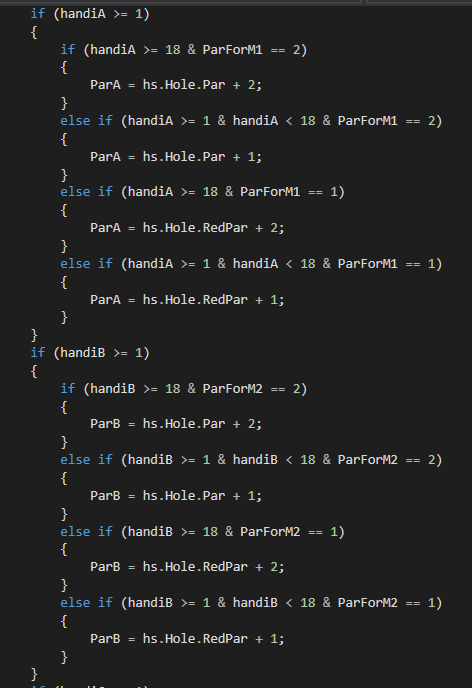
I have added the same long-winded way as above for the handicaps.



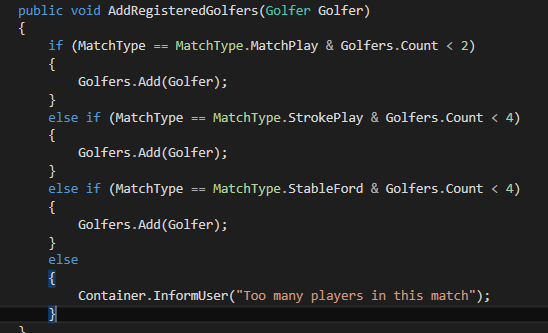
On reading through my note with my client I then realized that In stableford it’s not only the stroke index that will be affected for female it’s also the Par since females playing off a different par rating.

This meant that I had to make a few more changes to the code.

ParForM is then passed into the method that calculates and adds the scores to hole score.

Inside the Stable Ford add score method I then added:

Finally For Golfer validation I created another if then else statement on the add new golfer method which is really simple.

Simply if the user tries to add too many players it just won’t add them.

For email validation there is ceratian protocols that must be followed this can be shown in the finite state machine below:\

A-Z, 0 - 9

A – Z, 0 – 9, . ,

@

.

A – Z, 0 - 9

A – Z, ., 0 - 9

A – Z

A – Z

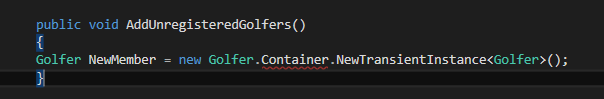
Drawing out the Finite State machine is very useful for seeing the checks that must happen to ensure the email is a valid one.

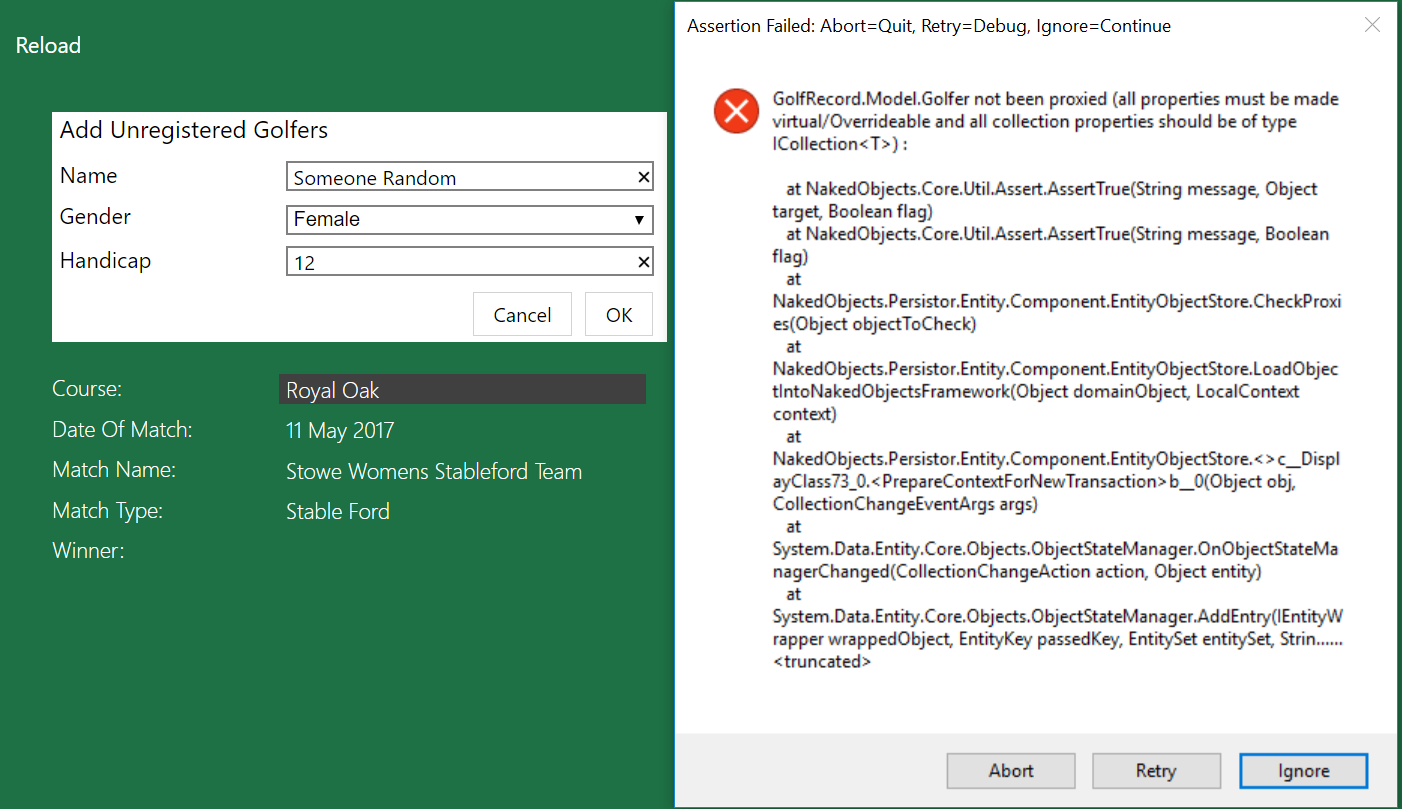
// Code showing this validation on email.

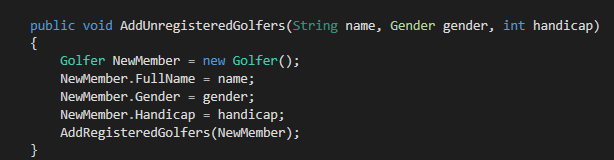
Iteration 5:

At the moment, you can only play with preexisting members of the golf app and this is a big problem. To sort this problem out I need to create a new type of golfer which only requires handicap, gender and a name. AT first thought this means that the collection of golfers is going to be a problem as I can’t have a collection of Golfers and Unregistered of Golfers.

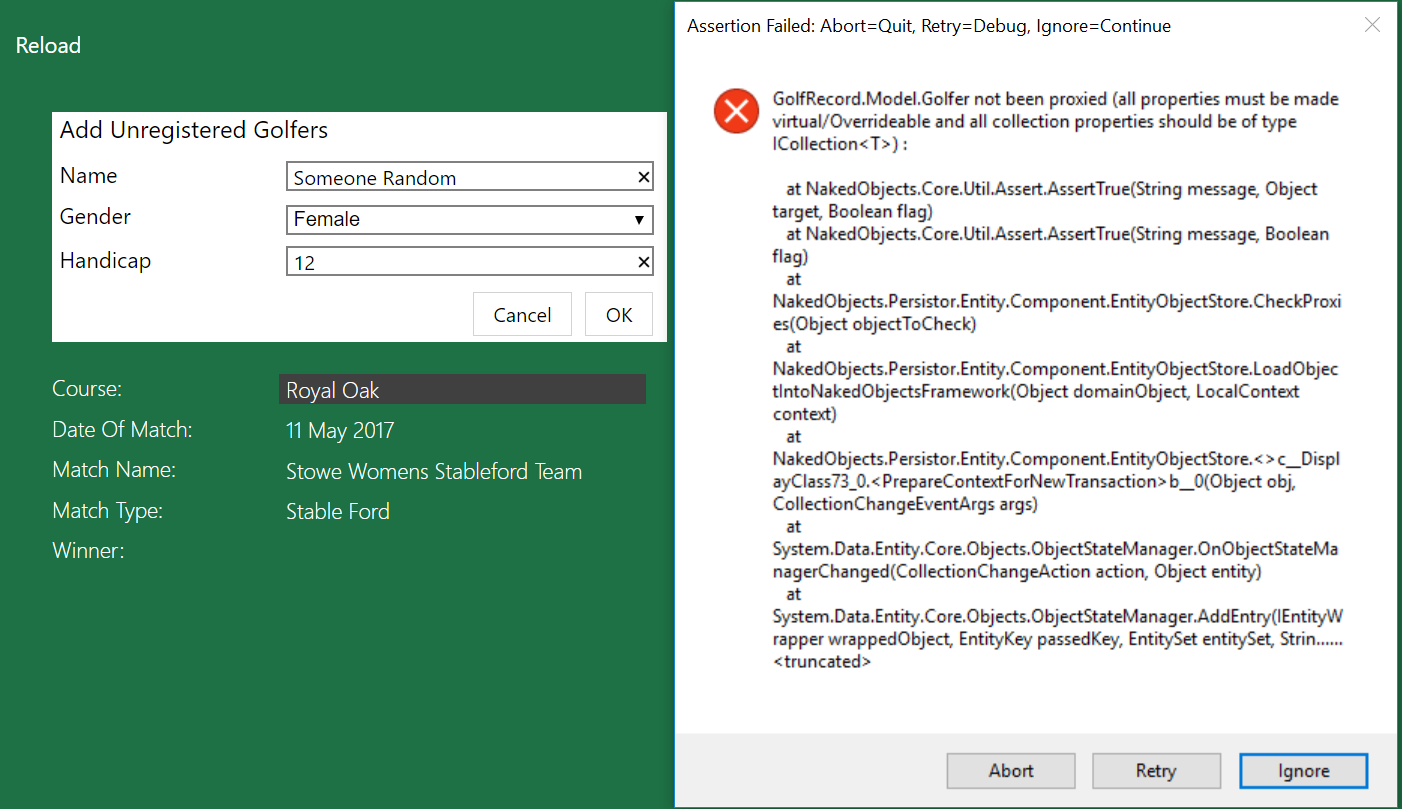
For my first attempt, I deiced that Unregistered Golfer could be a type Golfer but you only input the data necessary.

As you can see there is a problem with Container because Container isn’t a type in Golfer and is an injected service which is in all my classes almost.

As a second attempt, I wrote: At first glance I realized that there is nothing for the Id of Golfer which is the primary key. On testing this attempt the program runs and while trying to use the function I was given a correct style member:



Upon Pressing Ok a new error message that I haven’t seen before came up.

Since Entity is mentioned a lot with the error message I assume that this is the problem with GolferId that I mentioned before. Therefore, I try allowing you to enter Id as well to see if that makes a difference. This didn’t send an error but still didn’t work it just froze preventing me to click ok.

Upon re reading the error message it mentions that all the properties should be virtual.

Iteration 6:

Aim: to create a system to allow golfers to have friends meaning it is easier for them to access their favourite golfers quickly and see what they are upto. I will also invoke a autocomplete to this method to improve usability.