Handicap Calculator Project

Executive Summary.

Avid golfers are always wanting a way to track their progress and see how they are doing. My goal with this project was to create a way for golfers to track their rounds of golf and calculate their handicap. The system that I built provides a place where golfers can easily enter scores, and have the system go and grab certain critical information that is needed to calculate a handicap, and simply return the updated handicap to the player. I believe that avid golfers would be willing to pay a price in order to have all of this information so accessible and easy to use.

Implementation Documentation.

1. Updating The Course Directory

Golf course databases do not change very often, and it would be extremely time consuming to take the player through a series of searches in order to find the course that the player wants to identify. Due to this, I decided to simply maintain a course directory within the workbook, and when desired, the user could update the directory whenever they wanted. When the user goes to the Golf tab in the ribbon, they will see the “Update Course Directory” button, which when pressed, will update the course directory on the sheet “Course Listing.”

	A	B	C	D	E
1	**Course Name**	**City**	**Type**	**Phone Number**	**HTML Name**
2	Alpine CC	Highland	Private	801-322-3971	alpine-country-club
3	Bear Lake GC	Garden City	Resort	435-946-8742	bear-lake-gc-1
4	Bear Lake West GC	Fish Haven	Public	208-945-2744	bear-lake-west-gc
5	Ben Lomond GC	Ogden	Municipal	801-782-7754	ben-lomond-gc
6	Birch Creek GC	Smithfield	Municipal	435-563-6825	birch-creek-gc
7	Bloomington CC	Bloomington	Private	435-673-2029	bloomington-cc
8	Bonneville GC	Salt Lake City	Municipal	801-583-9513	bonneville-gc
9	Bountiful Ridge GC	Bountiful	Municipal	801-298-6040	bountiful-ridge-gc
10	Camperworld (Belmont)	Plymouth	Public	435-458-3200	camperworld-belmont

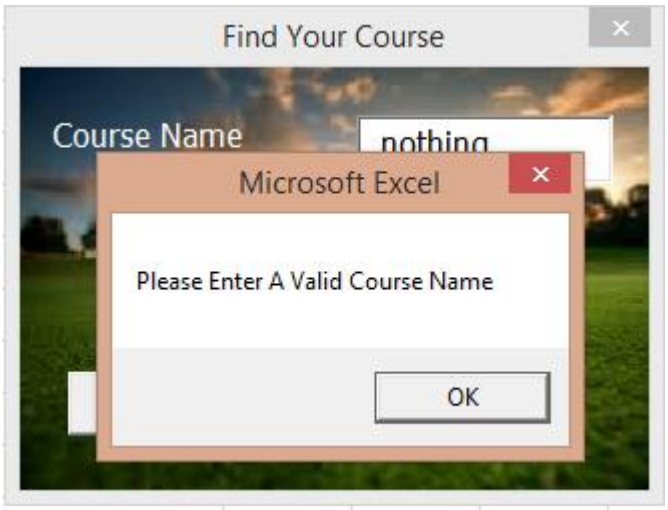
This macro goes online and pulls a comprehensive list of every golf course in Utah. The data that is pulled in includes Course Name, City, Type, Phone Number, and HTML Name.

2. Finding the Course

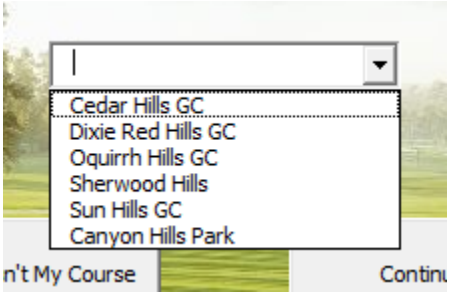
Once the Course Directory is up-to-date, the user can then begin recording scores from rounds that they have played. Using the button seen above, which is located under the “Golf” tab, the user can begin entering their score. When the button is pushed, the first thing that shows is a UserForm which will prompt the user to identify the course that they played.

The Userform lets the user enter the name of the course that they played. The user can choose to either “Find Course” or “Cancel.” If the user chooses cancel, then the UserForm closes and the program ends. If the user chooses to find the course, then the system will go to the sheet “Course Listing” and attempt to find the course name that the user entered. If no exact match is found, then the program will execute find in

order to find possible courses that resemble what the user entered. If find does not return any value, then the user is prompted with this.



When the user presses “OK” in the message box, then the message box will disappear and the user can enter another course name. If the program ends up finding a course, or multiple courses, that may be what the user is looking for, then the following UserForm, “Perhaps You Meant One of These…” will appear. If the system finds more than one match for the course name, then they will all appear in the drop down.



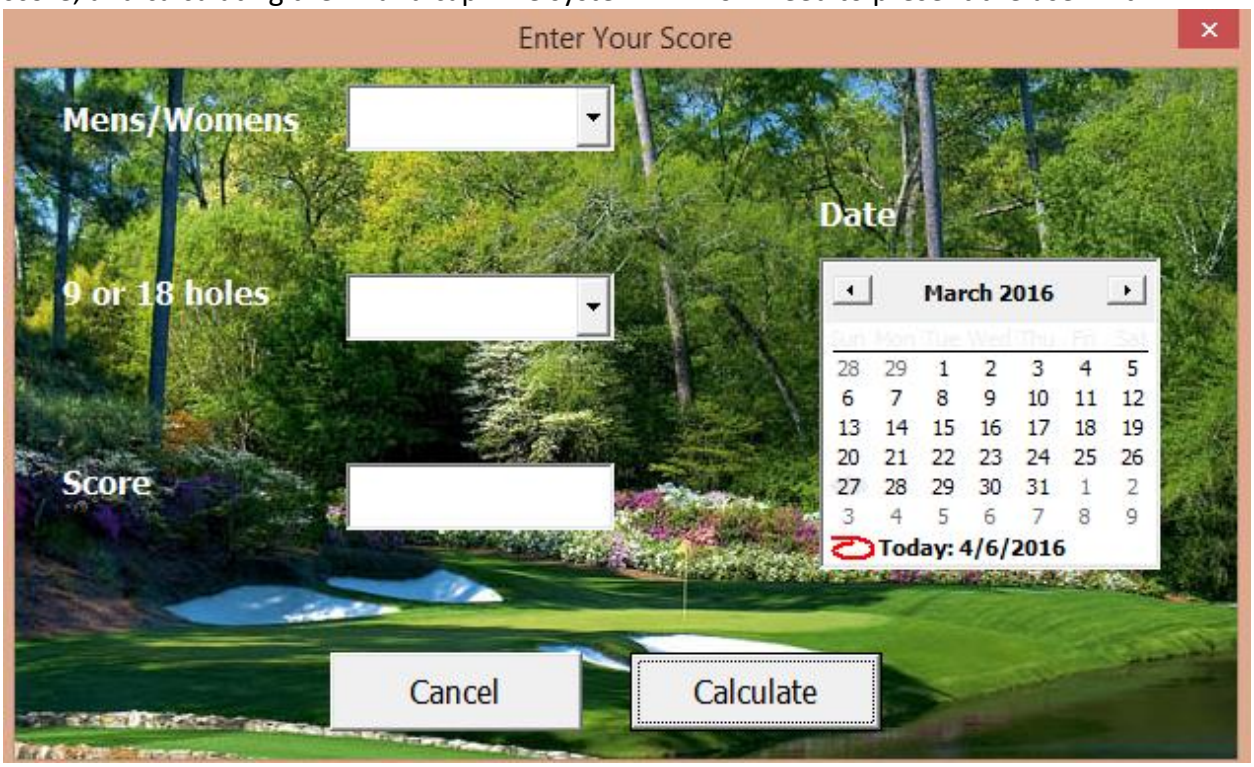
After this, then the user will either terminate the program by indicating that none of these options was the course they were looking for, or they will indicate one of the courses is the

correct option. Assuming the user picks one of these options, or they originally put in a correct name, the system will now have the correct course name to continue entering the score and calculating the handicap.

3. Entering Additional Information

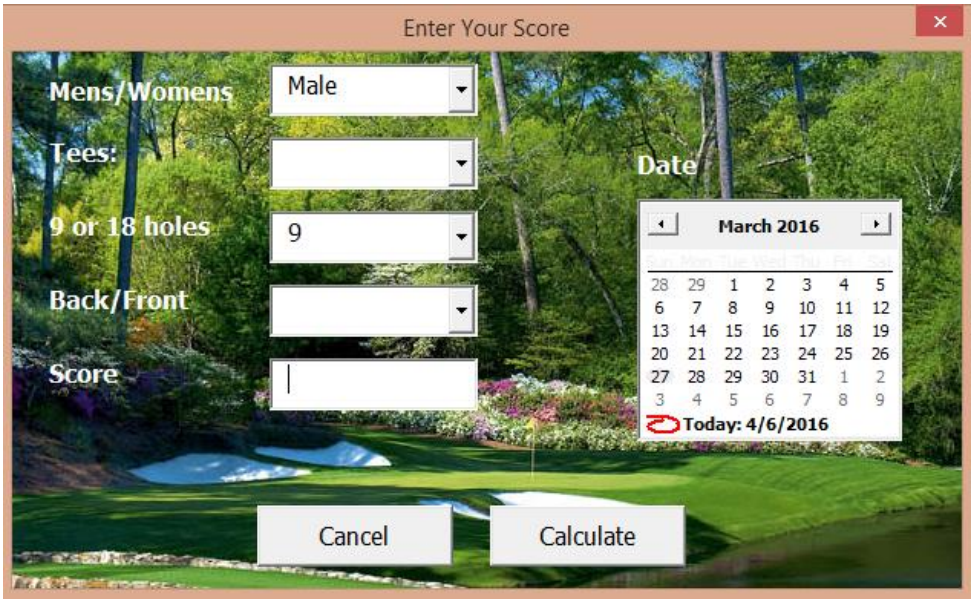
When the system has the correct course name, it will go back onto the uga.org website, and grab different tee color options from the correct course page, so that it may present the user with these options on the next UserForm. I used the agent class to run through the html code and find how many tee color options each gender needed, then I ReDimmed each array corresponding to each gender, and put in each color option for each variable in each array. The system now had all the options needed to present to the user.

At this point the system can now continue to guide the user through entering their score, and calculating their handicap. The system will now need to present the user with

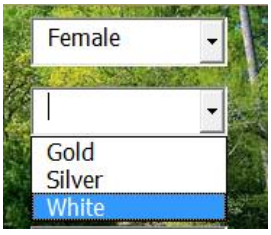


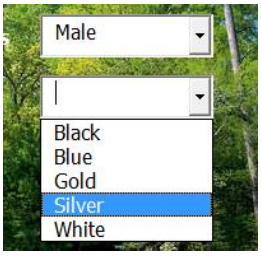
different options for their round. Even though the system now knows the course that the user played at, they will still need to enter different options for if they are “Male” or “Female” what tee color the user played, how many holes they played, the date, and their score.

On this form there are a couple of options that are missing, one is what tee color the user played from during their round, and the other is if the user played from the back or the front. The reason these are missing initially is because they require that the user first enter “Male” or “Female” in order to present the correct options for the tee color. Also, if the user selects that they only played “9” holes, then an option for either “Back” or “Front” will show.



As you can see above, depending on what the user picks, certain other labels and textboxes will show. Also, different tee colors will show depending on if the user choose “Male” or “Female.” (Note that the fact that certain colors are on both “Male” and “Female” options is not a mistake, the course has some of the same color options for both males and females.)





4. Calculating the Handicap

After the user chooses all of the options, including the date, they then can then click "Continue" and have the system go on to record their score and calculate their handicap. For help in understanding what happens next, here is the formula to calculate handicap:

Handicap Differential = (Adjusted Gross Score - Course Rating) X 113 ÷ Slope Rating

*Handicap Index = Average(Lowest x Number of Recent Scores) * .96*

As you can see from the formulas above, a lot of information is needed to calculate the handicap index. Each tee color for each gender for each course has a different Course Rating as well as a different Slope rating. So once the system has all of this information, it then goes back onto the uga.org site to find the corresponding ratings that match the information that the user has entered. Once the system has this information, it adds a row onto the "Handicap" sheet, and records the new score in the new row.

Handicap Index	10.5

Score History					
Date	Course	9 or 18	Back	Score	Differential
3/27/2016	Alpine CC	9	Back	78	77.31
3/27/2016	Dixie Red Hills GC	18		85	17.13
3/27/2016	Cedar Hills GC	18		90	15.21
3/27/2016	Alpine CC	18		90	23.49
3/27/2016	Sherwood Hills	18		90	18.74
3/27/2016	Birch Creek GC	18		76	10.77
3/27/2016	Alpine CC	9	Back	45	18.12
3/27/2016	Alpine CC	18		76	7.11
3/27/2016	Alpine CC	9	Back	45	18.12

Now that the system has calculated the differential and recorded the appropriate information from your most recent round, it can then go on to updating your handicap index. In order to calculate the handicap index, there are certain rules of how many scores should be used in order to calculate.

Number of Handicap Differentials Available	Differentials Used
5 or 6	Lowest 1
7 or 8	Lowest 2
9 or 10	Lowest 3
11 or 12	Lowest 4
13 or 14	Lowest 5
15 or 16	Lowest 6
17	Lowest 7
18	Lowest 8
19	Lowest 9
20	Lowest 10

I used a case statement to determine how many scores should be used, then I had the system find the lowest x number of scores, and then calculated the Handicap Index, which is put into cell “B1”. If there are not enough scores to calculate a handicap, the system will inform the user with a message box, record their score, and end the program.

Discussion of Learning and Conceptual Difficulties Encountered.

UserForms

As you can see from my examples, I used UserForms extensively in this project. One of the difficulties I encountered was how to manipulate the UserForm based on what options the user chooses. I learned how to change things based on if a certain textbox is changed. I also learned how to dynamically add values based on information that comes in from the web. One difficulty that I encountered is that the lists are constantly changing. One course may have 5 tee color options, while another may have 10. I used arrays extensively to dynamically pull items from the web as well as add items to the UserForm. I also learned how to manipulate the look

and feel of the UserForm not only when it initializes but also when the user changes items in the UserForm.

Working with Different Modules

Due to the fact that there are several different steps involved in the execution of the system, I used several different modules and several different UserForms. I also found that I needed to use public variables extensively in this project. I found that by using different modules, I needed to be very careful as to the order that my code is executing to ensure that certain events take place in order. Working with different modules was a challenge but I learned a lot from having so many different modules all working together.

Working with Change

One of the issues I encountered with this case is that because of the fact that every course is different, I had to learn how to dynamically find how many variables I need in an array, and then ReDim the array based on the number of variables. I also found that the course html pages are all very different. I had to cycle through the html first to find how many tee options there are for each gender, ReDim the array, and then allocate a certain value to each variable in the array.

User Errors

When I began working on this project, I started realizing that there were a lot of places where the user could put in wrong information and mess up the system. To counter this, I tried to formulate the system to make sure that the user could not make any errors. Most times the

user won't know the exact name listed for the course, so I created a way to put in a similar name, and then identify the exact course that the user wanted. I also tried to eliminate the possibility for the user to enter unnecessary information. For example if the user plays 18 holes, then they shouldn't be required to indicate if they played the back or the front. I also added controls to ensure that the user cannot enter a text value for their score, it must be an integer. I had to think about a variety of different scenarios to ensure that the user would not be able to make any errors while entering the information, to make the system as easy-to-use as possible.

Assistance.

Although I used internet searches quite often while doing my project, just to find bits of information here and there, I did not consult with any other persons on how to write my system.