Anchorage Women’s Golf Association

Team-Play Tournament Tracker

Final Report

**Prepared for:**

Dr. Butler, Software Engineering A401

University of Alaska Anchorage

**Project Sponsor:**

Anchorage Women’s Golf Association

**Project Manager:**

Dr. Shawn Butler

**Development Team:**

Tricia Reilley

Brooks Woods

Dustin Fast

December 12, 2017

# Project Description and Objectives

The Anchorage Women’s Golf Association requested the development of an application to automate many processes of their yearly tournament management. Previously, tournament management had been done with pen and paper, which is laborious and prone to error. Our objective was to implement the following features into the application:

* Construct balanced teams based on even distribution of player handicap.
* Creation of unique player matchups across all rounds.
* Track player points and putts for each round.
* Allow denoting of substitutes in scores.
* Determine final tournament winners.
* Maintain records of previous tournament data.

# Project Challenges

The first challenge encountered during the project was the unfamiliarity of technologies to be used during development. Although the team had some experience with graphical user interfaces, Windows forms proved to be initially challenging to work with due to limited experience. Many hours had to be devoted to learning how to program with Windows Forms before the team had enough background to implement something worth showing the client. Additionally, the team had no experience using C#, the language used to code the Windows forms. Again, many hours were spent learning before the team could move forward effectively. The team decided to use Microsoft Access, which uses SQL, to manage the AWGA database and experience with SQL was limited for most of the team. Finally, experience using Git was also limited and the team had issues with merge conflicts until a better understanding was obtained.

The next challenge lied in collaborative development. Readability of other team member’s code was an issue at times, especially when one member was working on a portion of the project that was very different from another’s. Clear comments and communication became key to overcoming this obstacle. Similarly, the team often had to develop in parallel, one building the database and another developing a portion that would use the database, for example. At times, it was challenging to test code effectively until both team members had finished their task.

Coordinating with the client went well in the beginning, but later on travel and work responsibilities made it difficult to communicate effectively. We were generally able to continue working until a response was received.

# Description and Assessment of the Software Engineering Process

During the software engineering process, the team used the Agile method. The process was organized into 2 week sprints, which include coding and testing, with the goal of being able to show the client something (stories) at the end of each sprint.

Epics were taken on in a logical manner and were implemented as follows:

* Build the database
* Manage members
* Assign teams
* Assign matchups and schedule
* Input scoring
* Determine results

Bugs found at the end of the current iteration were generally fixed in the following sprint.

The assessment of the Agile process is positive. The team generally delivered what was expected with a few exceptions. The process took a bit of getting used to in terms of time management. Since many of the technologies used were new to the team, we initially underestimated the time required to implement some of the features. As the project progressed, the team was generally able to assess the time required more accurately, and distribute points more appropriately. As mentioned before, each sprint had its own design consideration, implementation, and testing in contrast to the waterfall method which would consider each as a separate task or sprint. Sticking to the Agile method helped the team to make steady progress thought-out the project, manage bugs, and not fall too far behind.

# Accomplishments vs. Original Objectives

Our team is proud to say that all our objectives were accomplished. The following is a list of all the client’s requirements:

* + Construct balanced teams by player handicap.
  + Creation of unique player matchups across all rounds.
  + Track player points and putts for each round.
  + Allow denoting of substitutes in scores.
  + Determine final tournament winners.
  + Maintain records of previous tournament data.
  + Happy client.

# Testing Plan and Results

Our team tested throughout the project shortly after features were implemented. This included negative testing such as invalid inputs, incomplete required fields, and out of order operations (such as making teams before declaring players eligible). The results of this testing are that all the user interaction is handled gracefully.

Positive testing was also performed along the way. Once the main functionality was completed, the team used prior-year tournament data (i.e. players, team/round structure, scores, and final results) and constructed a test tournament to ensure that the output matched the prior-year’s results. Our application produced results as expected.

# Lessons Learned

Our biggest lesson learned is that collaboration can be challenging. Group dynamics can lead do different opinions on how to approach problems. With no formal hierarchy, it is difficult to determine who has the final say, especially when there is no consensus. Moreover, it can sometimes be challenging to appreciate the work of others when you might have accomplished the same task in a different way. Merging work with others can also be difficult especially when one team member has made drastic changes to the code.

Finally, we have learned that new tools can be difficult and time consuming to adopt, especially when you must find all the learning resources on your own. As mentioned before, most of the technology used for this project, Git, C#, and SQL, were new to the team. This required many hours with no real deliverables immediately resulting from the time spent.