**COMPUTER INFORMATION SYSTEMS**

**Union College**

**2018-2019 Academic Year (S2)**

**Final Project**

***Tournament Bracket Generator***

**May 3, 2019**

**Instructor: Dr. Seth McNeill**

**Ashton Fisher**

**Cole Yeager**

The existing system is quite lacking in both usability and functionality. It is a very manual process, and there is quite a bit of room for human error to happen throughout the process. Currently an Excel spreadsheet is being used. The first problem that exists from this is that the original is only accessible on a single computer. Each time that the document is changed on that computer, it has to be re-emailed out and replaced on any other machine that was viewing that document.

The next problem is that there are many manual steps and with each one there is a chance for human error. The first is with score input; the integrity of scores that are input must be upheld for the entire system to function. These are input into the different pools that exist within the tournament, Figure 3. Next, once scores are input, there are some calculations that the spreadsheet performs. This includes data being transferred to a single sheet, Figure 4. Once this is done, someone has to manually look at all the data and perform all the tiebreaker situations by hand to determine seeding. This is where the largest chance for human error comes into play. It is very easy to miss just one scenario and therefore the entire seeding would be off.

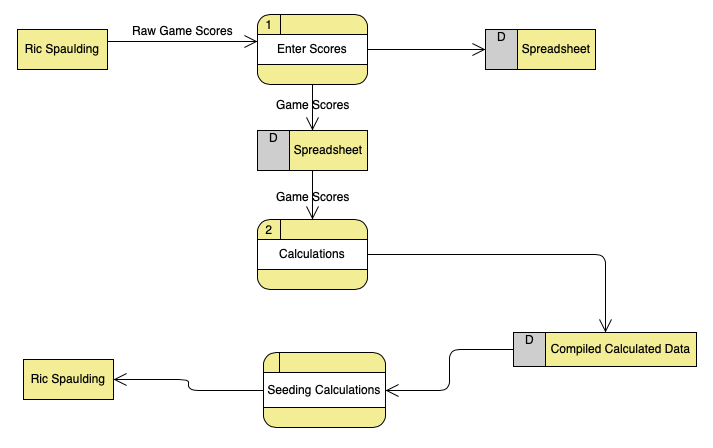


Figure 1

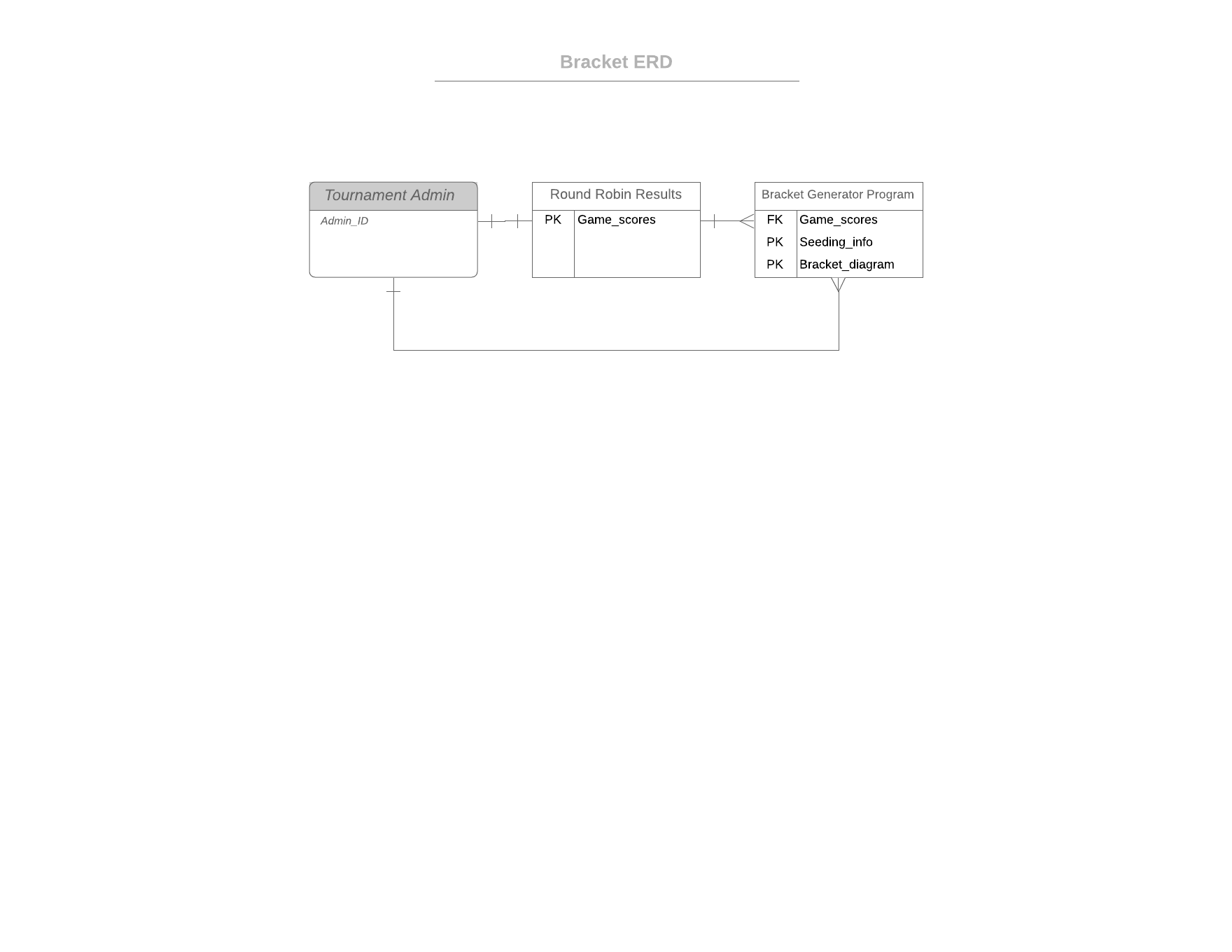


Figure 2

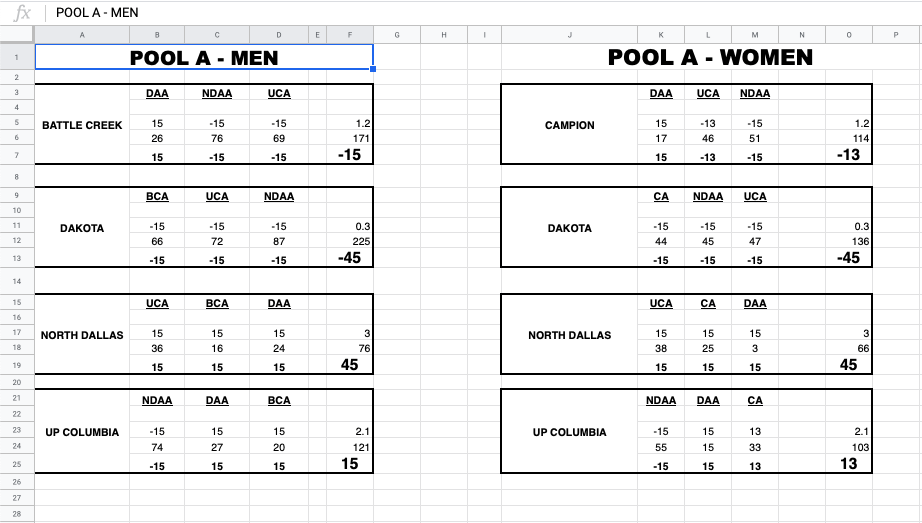


Figure 3

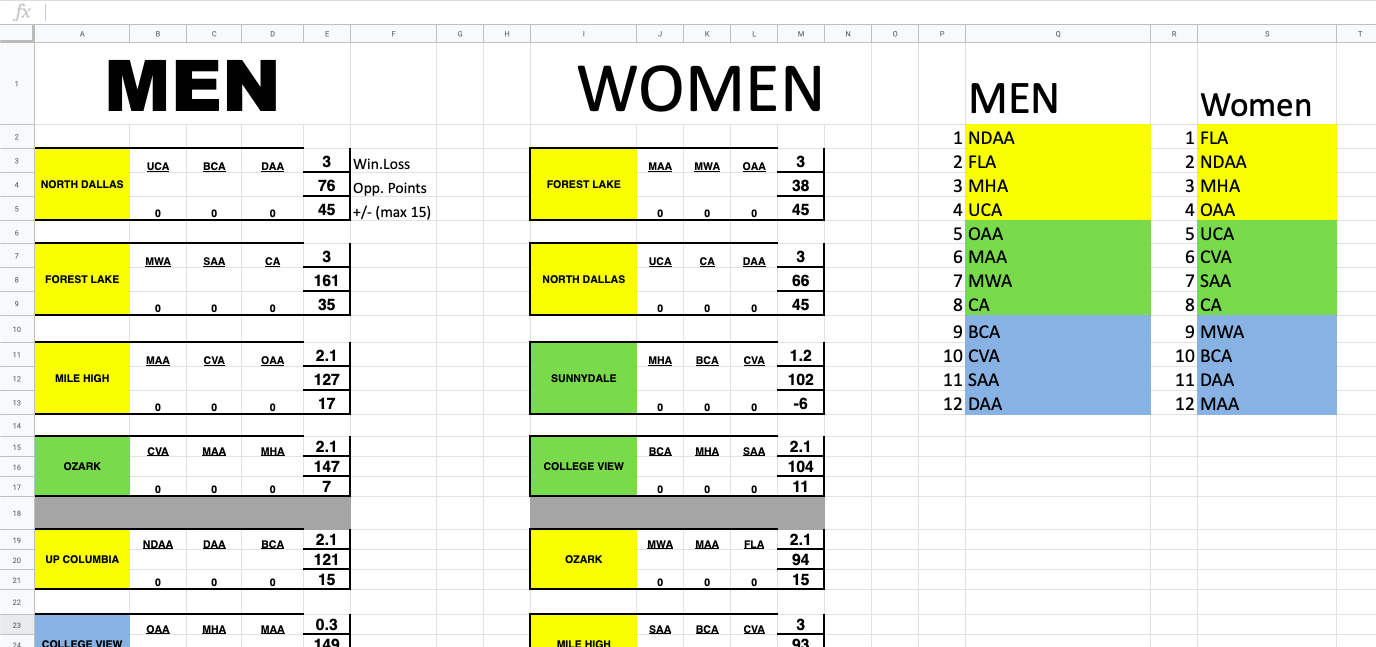


Figure 4

The existing system works but leaves far too much room for human error. Also, it only works well as long as Ric is around to make it work. He is the only one that truly understands the system, so if someone else were to take over they would have to start from square one. The thing that needed the most improvement was taking manual calculations away from the user and coding the computer to do them. This way there will be fewer mistakes. This is something that would be considered as a critically needed change. This is really the only thing that is a must have change from the old system.

There were some desired changes as well, though. This was more design related. Things were not formatted extremely well and so it could sometimes be confusing about what data went with what other data. A design that was very clear about where data belonged was something that we also spent some time doing.

These really were the only two areas of change. Data calculations and data formatting needed improvement, so that is what we spent our time doing. With the new design, all the user must do is enter team names and then team scores. Once that is done, all other calculations are performed by the computer, and formatted in an easy-to-read design.

Using the information we gathered during the interview process. We learned that our target users wanted the simplest version of our solution possible. Both Ric and Ashton were used to the old method of using an Excel sheet to calculate the seeding and spread of every team in the tournament. If we moved to a different program or developed a solution that involved more work, the end users would be less likely to want to implement our solution. We also learned that the simpler our final implementation could be, the easier it would be for Ric or anyone else to use in the future. In other words, we needed to follow the KISS acronym: Keep It Simple Stupid.

The first alternative solution we came up with is the simplest: Using a Google Sheets document with custom scripts running in the background to perform the movement of data between tables and the seeding of the teams. This solution, to us, was the simplest because Google Sheets is very similar to the previous method of using Microsoft Excel, and would therefore be the easiest to implement. The second alternative we came up with was to write an executable program in either C++ or Python with a custom GUI interface. An extra feature of this solution would have created a printable PDF file with the physical bracket information drawn up. This way you wouldn’t have to draw brackets by hand afterwards and place the teams in order, you could just print the PDF file and hand it out to anyone who needs that information. This solution requires the most amount of work. The final alternative solution we came up with is by far the most time consuming: Doing all the work by hand. Users would have to record all the data down on a piece of paper and perform all calculations by hand. Using this method would be the least accurate, because human error would make it easy to make a mistakes in adding the different numbers and making comparisons to determine seeds. This is by far the least productive alternative solution.

The person that I interviewed had a fair deal of experience with Google Sheets. He made mention of some pros and cons that he found in Google Sheets. For instance, the user made some associations of his experience with Excel. Excel is universally known as an excellent tool for organizing data. The user especially found Google Sheets’ query function to be useful. He recalled a project where the function allowed him to write a SQL query to represent the data within the sheet itself. Another benefit was the calibration. The user expressed, “The calibration was nice, you can have multiple people working at the same time.” I imagine this would be a great tool for group projects.

Some cons the user disclosed was, “the charts and tables are not as good as Excel’s, they are missing some functionalities that one would prefer in Excel.” He then added that in order to get some of that functionality, you have to first understand that Excel has Macros built-in. So, Google sheets can have a similar experience, but you would have to use a Google Apps script to get similar functionality.

After asking about what his experience was like, we got into logistics and statistics of it all. One project he worked on in particular was Union College’s Course evaluation form. The user narrowed the idea as a database where students would answer the given questions and all of the answers get logged in a database where the answers could be accessed to be weighed. He said this endeavor’s maintenance time was recorded as, “three hours of figuring out what went down.” The initial time it took him to configure everything, was about eighty hours; however, the overall time spent was an additional eighty hours, approximately one-hundred and sixty hours total. A potential solution would be stick the data in a query and spit out a spreadsheet, much like what we did in this project.

We decided to use the first alternative solution we came up with for our final solution. We chose to go this route because it would be the easiest to finish and implement before our deadline. We also went this direction because the Google Sheets interface is familiar to our primary user, and would therefore be the easiest for the end user to understand. The Google Sheets document is very simple to use, and only requires the user to do four things: Add team names to their designated pools, add their scores for each game, move the data to the rest of the tables, and seed the teams. The last two steps are the most complicated, so we made custom macro functions (a feature of Google Sheets that writes JavaScript functions based on your actions while the feature is activated) that are linked to two buttons in the Seeding page. When clicked, each button performs the macro functions and moves the data and seeds the teams. To make sure the spreadsheet couldn’t be broken by accident, we locked every cell that didn’t require user input.