**CPTR 327**

**Final Project - Tournament Spreadsheet**

**Members**:

Ashton Fisher

Cole Yeager

Aksana Buster

**Project Leade**r: Ashton

**Roles of other members:**

Cole - Technical work as well helping to design final formatting.

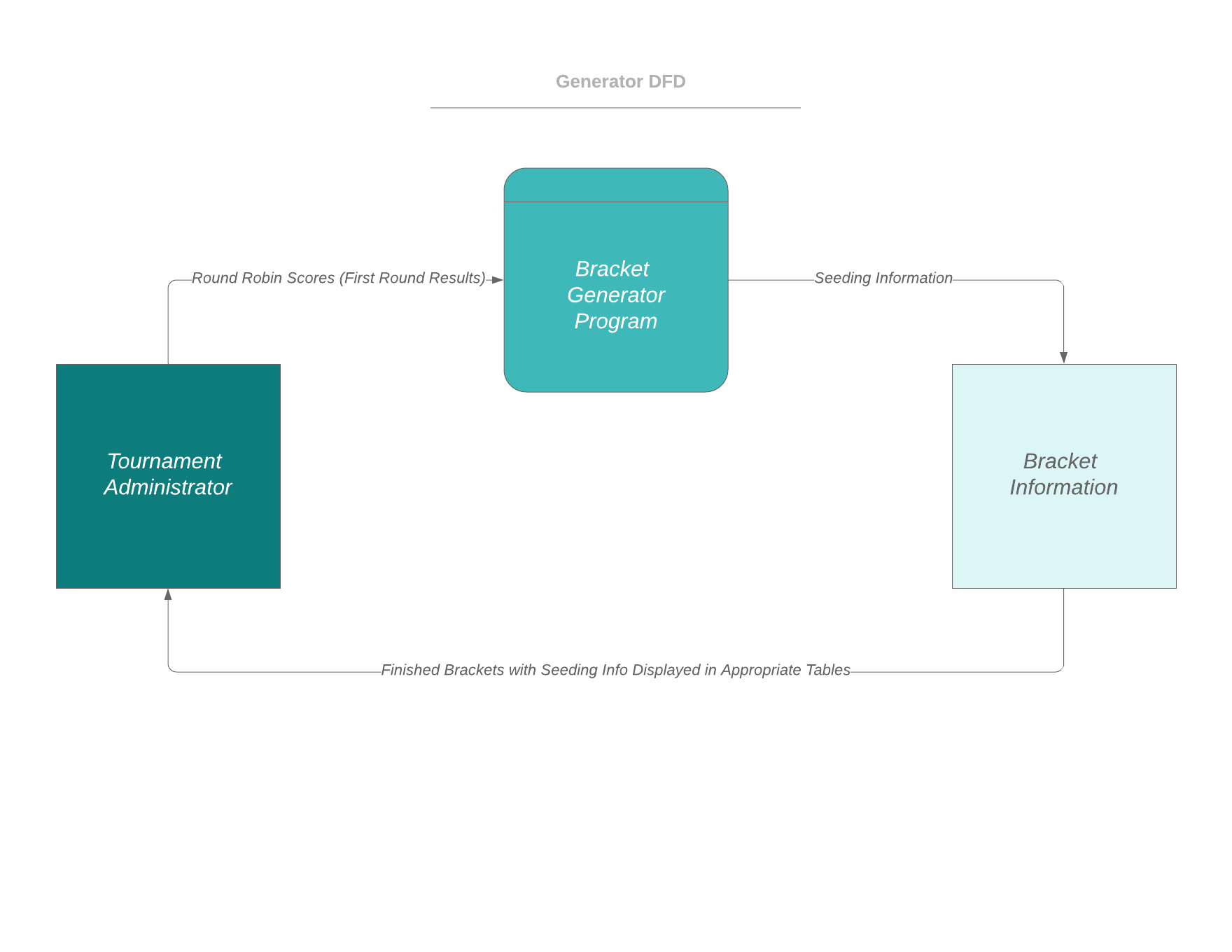
Aksana - Technical work as well helping to design final formatting.

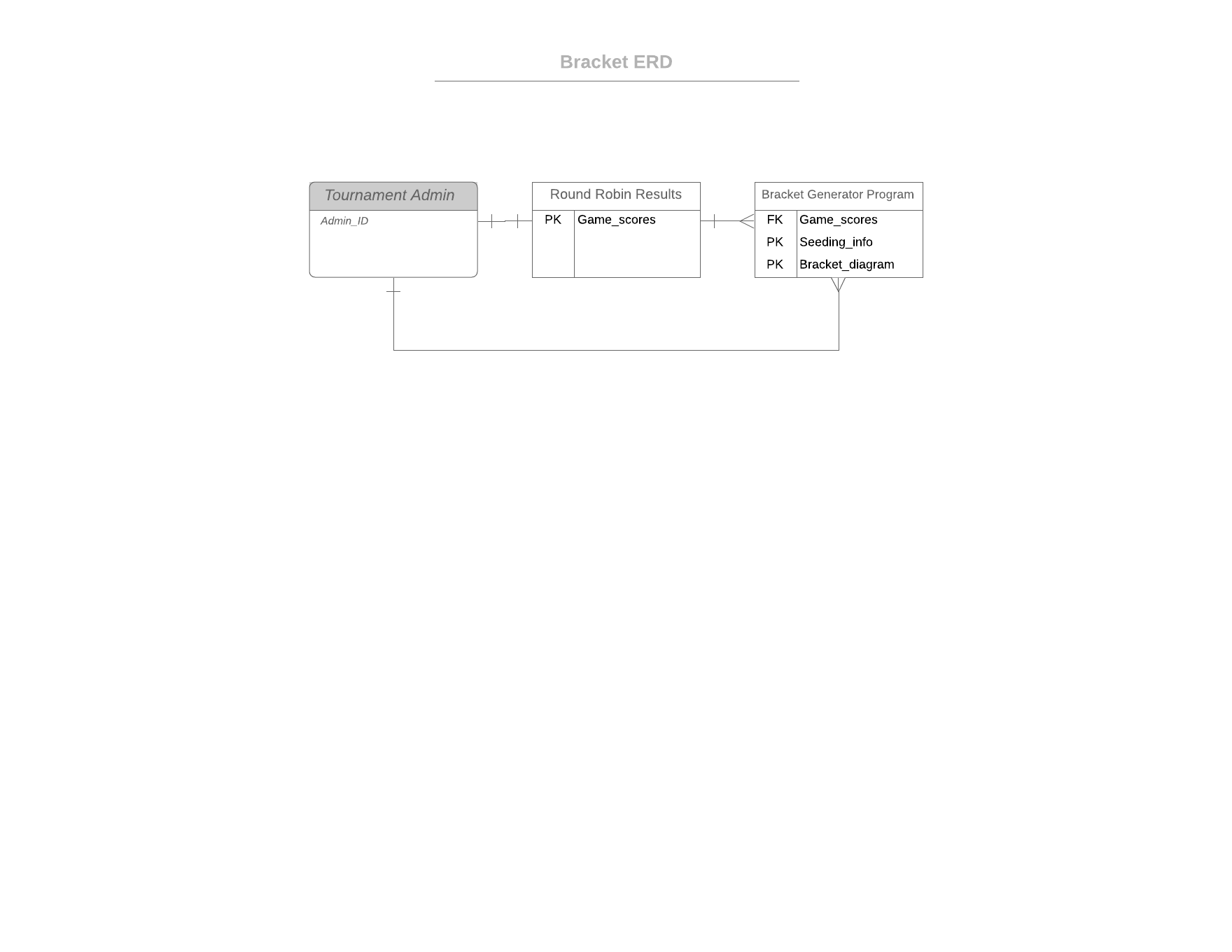
**Description**:

Every year Union holds several tournaments, hosting different academies from across the country. One of the most difficult and time consuming things is figuring out team seeding for the weekend bracket. But, if a spreadsheet could be created that would calculate all records and tie breakers, and then place teams in the correct seed automatically, it would save a lot of time.

**SDLC Stages**

* **Systems Planning and Selection**
  + **Priorities for the Project:**   
    The priority is to create a spreadsheet that can generate tournament brackets from information input from the user.
  + **Specification of System Scope:**   
    Tournament brackets for any academy tournaments Union College hosts.
  + **System Justification:**This spreadsheet would make creating brackets for UC tournaments much easier, and faster, which speeds things up and leaves more time for the Admins to handle anything else that may come up during a tournament.
* **Systems Analysis**
  + **Description of current system:**  
    The current system is a rough Excel document in which many different data points have to be entered. The score of a given game has to be entered, as well as the spread of the game and the opponents point score total. Once all of this is entered it is totalled for each team so that record, total spread and total points allowed can be seen. Then seeding has to be performed by manually comparing the different data between all of the teams.
  + **General recommendations:**  
    The first recommendation is going to be simplification of input. Our dream is that the only data that has to be entered is the game score between two teams. Then all of the other data can be calculated. Once the user has entered the game score, it should automatically adjust the total seedings upon comparison of different data points.
  + **List of alternatives:**  
    One alternative is to perform all of this analysis by hand. It would be extremely time consuming to record all the data down on a piece of paper. Then it would also be very easy to make a mistake in adding the different numbers and making comparisons to determine seeds. It is not a good alternative. The other alternative is on the other end of the spectrum: Coding an entire application with a custom GUI in either Python or C++. This will also be extremely time consuming, so much so that it is doubtful that we would be able to get a stable version implemented by the time of the deadline.
  + **Justification for choice of particular alternative:**  
    The reason we have chosen to pursue the particular alternative is that it relieves time and stress from the tournament director. During tournaments they are being stretched a million different ways to try and cater to the many different needs of the participants. This means that they do not have time to sit down and pour over the data to make sure the seeding is correct. It would save Ric hours of time if all he had to do was enter in game scores and have seeding calculated automatically, which is the alternative we have chosen.
  + **Acquisition plan for needed parts of chosen alternative:**  
    We will only need two things: Google Sheets and Google Scripts. These resources are open source and free to anyone with a Google account, which is also free.
* **Systems Design - Detailed Specifications**
  + We would be inputting game results from the first round (“Round-Robin”) of the tournament into the spreadsheet. The scores against opposing teams should be all we need, and the spreadsheet should be able to seed the teams automatically and generate the brackets for the rest of the tournament. An extra step would be to have the spreadsheet physically draw out the brackets so that administrators could print them out and post them instead of having to read them off.





|  |  |  |
| --- | --- | --- |
| **Conditions/Courses of Action** | **Rules** | **Rules** |
| Program Type | Google Sheets/Scripts | Python or C++ Program |
| Easily Accessible by Anyone | **X** |  |
| Easy to Use | **X** |  |
| Custom GUI |  | **X** |
| Maximum Compatibility | **X** |  |
| Draws Bracket Table for User |  | **X** |
| Supports Tie/Tiebreakers | **X** | **X** |
| Upgradability | **X** | **X** |

**UPDATE 4/1/19 - We have decided to move forward with Google Sheets/Scripts**

* **Systems Implementation and Operation** - Only minimally required for this class but the following are a good minimum:
  + **Documentation - all that you have so far:**  
    We have the excel spreadsheet that has been used to calculate tournament seeding in the past.
  + **A preliminary training plan for the new system:**  
    There should be minimal training required to use our system. Once it is complete it should be very simple to use. Our goal is to be able to show Ric how to use the new system in a matter of minutes. It should be as simple as opening a Google Sheet document, and entering in game scores in the required areas.

**Interviews**

This will require us to interview the athletic director, Ric Spaulding, as well as group member Ashton Fisher. Both have spent a lot of time configuring brackets for Union tournaments in the past and so will know answers to all the questions we may have. Ric Spaulding will know exactly what outcome is going to be the most desireable and productive and so his opinion will be very valuable.

**Gantt Chart and Network Diagram**

1. Create basic Google Spreadsheet
2. Rough format of document
   1. Create areas for the different team “pools”
   2. Create area to compile team data
   3. Create area for final seeding
3. Figure out how to display teams so entering in game scores are the simplest
4. Calculate and display team record based on scores entered
5. Calculate and display team spread (difference between opponents score and team score. Can be positive or negative number) based on scores entered
6. Calculate and display opponents points scored (sum of all allowed points) based on scores entered
7. Figure out how to automatically seed all teams based on a system of tiebreakers
   1. First determinant is record
   2. Second determinant is spread
   3. Third determinant is opponents points scored
8. Final spreadsheet should allow the user to simply enter in game scores for all games. Then it will display the three categories of data so that data integrity can be manually checked if desired. Third it will automatically seed all teams based on the data criteria.

