Game Contest Server

Phil Brocker

Department of Computer Science & Engineering, Taylor University, Upland, Indiana 46989

Introduction:

Game Contest Server Project is a Computer Science Department project that was started in the fall of 2013. The main scope of the project is to allow automated players to be uploaded so that they may individually challenge each other as well as participate in tournaments. The Game Contest server is written in Ruby on Rails and has the flexibility to accept players written in multiple languages.

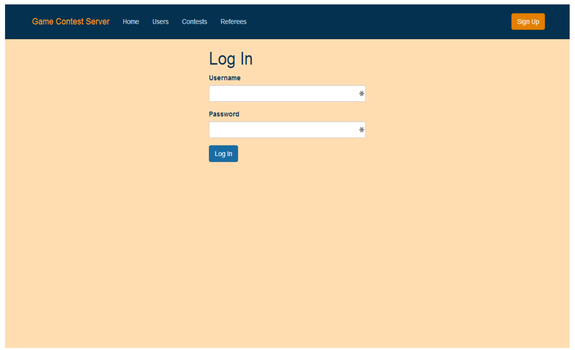

Figure 1: Screenshot of sign in page of Game Contest Server

Game Contest Server:

Game Contest Server front-end was primarily developed throughout the fall semester of 2013 by Dr. Geisler and students. It was then handed off to a group of primarily seniors, including myself, to continue to work on during J-Term J-Term. We were given the task of completing the front-end development as well as setting up back-end development that would actually run matches and tournaments. Some of the goals that we set for our team were the following:

J-Term Goals:

1. User Interface Redesign
2. Admin Interface Functionality
3. Continuous Test Integration
4. Back-end Development
5. Pagination and Search Functionality

By successfully accomplishing these goals, my team was able to implement full functionality of the project from front-end to back-end. As user, one can now upload their automated players and be entered into a contest by a professor. The tournaments and matches are then run via the back-end and the results are saved in a database which a user can view from the front-end website about each of their players and tournaments that they participated in.

Database:

In order to complete these goals, a server was setup to host the project and allow future development and access available to anyone on campus. Throughout the development process some of the front-end functionality, test, and ERD (Entity Relationship Diagram) had to be refactored in order to allow for improvement and flexibility in future development.

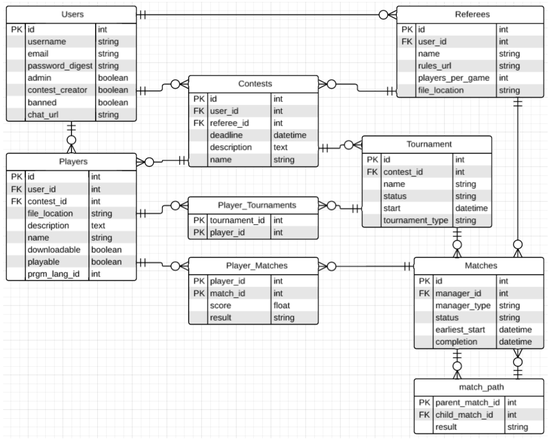

Figure 2: Screenshot of current ERD of Game Contest Server database.

Some of the database changes that had to take place were the addition of a tournaments table so that professors could specify which students would be included in which tournament. As well, a table associating players with tournaments was created. The reason why we added these tables was because without them there was no functionality to allow a contest creator to specify what players were associated with a specific game. Now a player may be associated with a contest such as checkers, but there is flexibility for the contest creator to run multiple tournaments such as single elimination and round robin without having to make a complete new checkers contest. This functionality allows a contest creator to remove players from a tournament if the player is broken. A match path table was added to allow the creation of games that rely on individual matches to be run. An example of this is single elimination tournament where the winner of a game moves on, but the loser is removed. The match path table is able to associate future games with past games allowing two winners to play each other. Without this table there would be no way to keep track of previous matches and matches would not be able to be created during game play.

Back-end:

The back-end functionality corresponds to the ERD very closely. It consists of a daemon that checks the database for tournaments that need started. The tournament runner script then creates matches and associates players with tournaments. Another daemon that is running listens for matches that are ready to be started. It then initializes the referee with a port to talk with the match runner on and initializes the players with a port which they will communicate with the referee on.

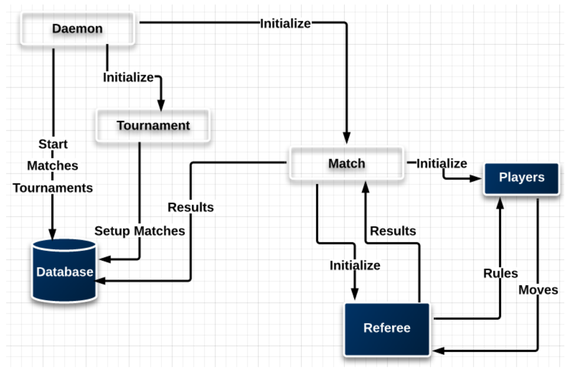

Figure 3: Screenshot of current Game Contest Server back-end.

The referee then handles the game control and reports the results of the game which is then written to the database. One of the biggest issues we had when creating the back-end was a huge road block when initializing a tournament. The problem that we ran into was when creating a tournament, we would temporarily store all players who were associated with the contest into player tournaments tables, but we only wanted to write to the database, the players that were actually included in the tournament.

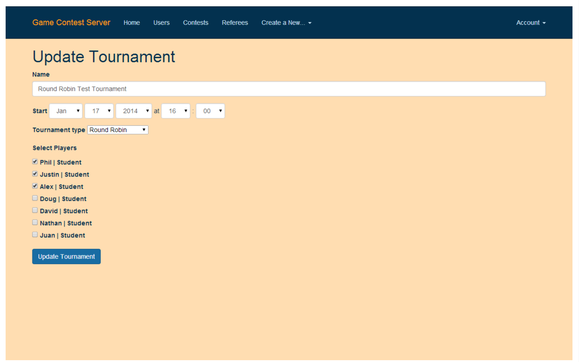

Figure 4: Screenshot of front-end updating a tournament.

The way that we overcame this problem is by adding functionality to the tournament model that allowed us to store the player ids that were checked off to be included in the tournament. It then queried the database only associating players to tournaments that had been checked off so all the other temporary player tournament tables that we stored in memory when creating the tournaments were destroyed. The same idea was used when editing. Since database objects already existed for players that were checked off in a tournament, when you unchecked them and submitted we would delete the records of those who were removed from the tournament by using the player ids that we stored in memory from the front-end check boxes.

Conclusion:

In conclusion I was surprised at how much more functionality that we completed than what we set out to complete. At this current stage, the Game Contest Server can successfully run both round robin and single elimination tournaments from front-end to back-end. As I was the team manager for the project, I believe that I can say as a team all the members meet my expectations of what they said they would accomplish. As then is still functionality still needs to be added. I would say that I truly enjoyed working on the project and would enjoy to continue developing it. I believe that it will be a great resource, not only for the department, but also for others as well.

Future Work:

- Individual challenge functionality both front-end and back-end
- Visually display games
- Record/Playback games
- Allow for more than two player games

Acknowledgements:

- Dr. Geisler for helping us to debug, brainstorm, and guide us along the way
- Dr. Brandle for help with network and port communication issues, as well as for his Battleship players
- Dr. White for his checkers players
- Nate White and Nathan Lickey for helping setting up the server and also help with performance issues
- Lara Horsley and Izzy for proof reading this poster for me.

Future Information:

- Please contact Dr. Geisler at jgeisler@cse.taylor.edu