TOUCHE: Coding Competition Software

Matt Goldsberry
Department of Computer Science and Engineering, Taylor University, Upland, Indiana 46989

Introduction

Touche is a coders system. We coded to allow for more coding, an interesting concept. Touche is a system designed to handle all of the work needed to hold a coding competition. Interest in coding competitions is growing as fast as the computer field itself. It is only natural that programs are written to enhance them. Our system has been in development for many years. It was built on the back of overstressed profs. and teams just like ours. Naturally because of the sporadic develop there were places where the code was not in the best shape. There were lots of inconsistencies, superfluous commented out code, and little to no explanation of what the code was doing.

The old system is pictured above. It reeked of the 90's.

Coding competitions

Coding competitions are battles of the mind. Players enter as teams and try to get as many questions right as possible. The questions are compiled and graded by our system. The grade is not returned to the teams straight away though. The compiler responses are sent to a human judge who then has the final say. The teams then get confirmation of whether what they submitted is correct or not. They do not get very much information though. There are only seven possible errors that they could receive. Finding where they went wrong is up to them (intentionally so). Any submission that comes back incorrect is treated as a penalty for that team. The way that teams are graded involves the time taken to get a correct submission and the number of incorrect submissions. The winner is the one who gets the most questions right in the shortest amount of time and fewest number of submissions.

Configurable time penalty

One of the things that I personally contributed to the system is the configurable time penalty feature. What this does is allow the admin to set how much the teams are penalized for an incorrect submission. The way that this works is that a number of minutes is added to the team's time for each incorrect submission. The way this is calculated is this:

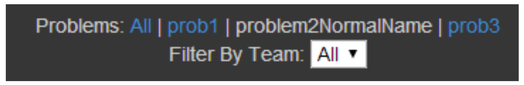
$$T+(S-1)*P$$

(Where T is the time taken to get a correct submission, S is the number of submissions, and P is the penalty that has been set.) The reason that it is "(S-1)" is because not all of the submissions are incorrect submissions. Presumably the last submission is a correct one that the team should not be penalized for.

The way that this was implemented was mainly using the database. When the admin sets the field in the form it sets a field that I added to the database. The standings page queries the database and then plugs the value it receives into the formula above.

Shortened problem name

Another piece of functionality that I implemented was the idea of a shortened problem name. The problem name field allows for a max of thirty characters. Anywhere where all of those names need to be displayed is going to take up a lot of space. One place where this became a problem was on some of the judging pages. The clarifications page and submissions pages allow for filtering by problem. That means that all of the problem names are not only displayed, but are displayed as links. This really complicated things from a UI standpoint. The user should not be forced to visually wade through so much text in order to find the problem that they want to look at.
Under the hood all I did was add a field to the database. When the admin sets the field the database updated. The other pages query the database to get the value.



Above is one example of where all of the problems are displayed. The short name for all of the problems was set to "probX" by me. The long name for problem 2 is displayed because it is selected. This allows the clarity of the full names to be accessable without all of the clutter.

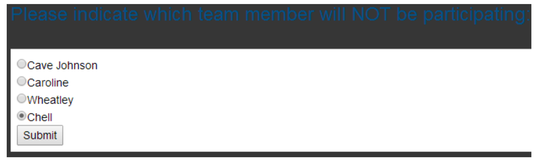
Non-participant page

The last piece of functionality that I got done was a page that gets displayed for the contestants. The issue that was fixed with this fuctionality is because each team is allowed to list an alternate participant. In competitions that are pre-requisite to other competitions they need to know who actually participates. Only the three participants that participate in the first contest are allowed to be in the second.

The first time each team logs in they are given a page that displays all four of their team members, each associated with a radio button. All the user has to do is select which member will NOT be participating and click submit.

All that happens is that there is a field in the database that gets set. Anyone with database access can see it. This was not that high of a priority item so simply having it in a place that can be found is plenty.

The way that this was implemented is mainly in the user login page. It queries the database upon login to look at the "non-participant" value for the specified team. If the value is still the default value then it redirects to the page that asks the team to set the value. If the value has already been set then the team is redirected to the main page for them straight from the login page. This is how they only see the page one time.



It is a very simple page. Hitting the submit button would set the "non-participant" field to "chell". The fourth name is the alternate, and the default value to be selected..

The logic is that we needed to know if the sub was "subbed-in". This simply treats all four players the same and asks which is not participating. We can know whether there was a sub by the result. If the sub did not participate then they will be the "non-participant". If they were subbed then whoever they went in for will the the "non-participant".

Future Improvements

- There are more places where the shortened problem name could be put in. It was something that I was doing while the server was down. It was not a very high priority thing so I moved on when I could.
- The non-participant value could actually be put somewhere other than the database. One thing that could be done is that it could be added to the e-mail which can be sent out. An e-mail with all of the information about the contest can be sent to the teams and the admin. The database is sent along with the e-mail, so the non-participants are technically included, but this is not ideal. It would be easier to have a specific text file that has all of the values rather than forcing the admin to wade through the database.
- It would be ideal to have a case in the "non-participant" functionality that accounts for teams without alternates. I had this working, but it broke the day before the presentation. All that I did was check to see if the "alternate" field was blank, and if it was then set the non-participant field in the database to "none".

Conclusions

I learned a lot through this project. I did not know any PHP coming into this project. I brushed up on my HTML and SQL as well. Those are things that I could have easily learned on my own though. The most valuable thing that I took from this was how frustrating it can be to work with messy code. I had not seen the product of not following best coding practices before. It was certainly annoying (but completely understandable in this case). A third thing that I learned is how rough gitHub can be. Merging is rough. If I had the project to do again I would try to get into more of the server setup. I have not worked in linux as much as I should have, and have little or no server exposure. I played to my strengths for the good of the group. I do not regret that , but cannot help but wonder. If you read all this then you are truly special. Thanks for reading.

Touche

For More Information contact Dr. Geisler.