TOUCHE: Coding Competition Software

Daniel Sanders

Introduction

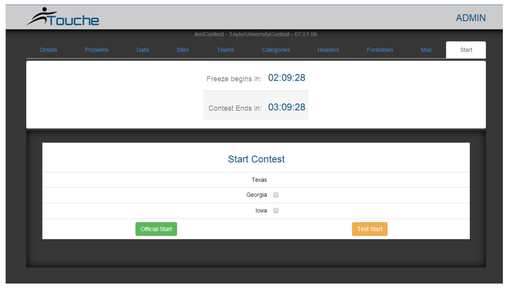
Just a little background on Touche. The project has been in development for many years now. It was originally written in the late 90's but was last updated in 2005. It's written in mostly PHP and HTML, with no CSS or modern design tools. It is web based software that runs coding competitions and allows for all the functionality you would expect during a competition. It is built around the three roles of the people that will be using the system: the admin, judges and teams. Each has a separate login and various pages that allow them to do what they need. The code we were given was working code but had been adapted and configured to run on a CSE server. Meaning when we received it simply putting it on a new virtual server was not enough to get it working. We had to spend hours and hours right at the beginning of the project configuring the machine to work correctly. But once we accomplished that it was time to start editing the code and making changes.

Initial Decisions

I was team lead on this project so it fell to me right off the bat to decide where we wanted to go with this project. We had been given the list of issues, possible bugs and questions that we could address. But just jumping into it and starting to fix things was not necessarily the right thing to do. Even before we got the server up and running I had decided to use Pivotal Tracker to monitor the issues and assign tasks. I had used it previously in ISD and felt like it was a very capable tool. So I set my team up with Pivotal Tracker and added in every one of the issues and bugs that we had been given. By working with Dr. Geisler I was able to organize the 28 different issues and set certain ones as high priority. Then we roughly ranked them according to importance and let team members start selecting tasks that would be suited to their skills. In addition to Pivotal Tracker I also worked to get Git setup and working. I had to think of what the best way would be to manage various code repos and merge code as we all worked on fixes. Eventually I decided on using various branches on GitHub as a central repo and then pushing and pulling from there as each person completed milestones. It was a bit rough at times but in the end worked quite well.

Moving "Start Contest"

The first big code change I made after getting everyone set up and things were working correctly was transferring the start contest code. With the original implementation of the site the admin would configure everything about the contest but then the judge would have to log in to his pages in order to start the contest. In addition to make things even more difficult the restart contest functionality was under the admin pages, so if the contest needed to be restarted the admin had to clear the old one and then have a judge start a new one. So one of the recommendations we were given was to unify this functionality. In thinking this problem over I thought it made most sense to just give the admin entire control over creating and restarting contests. This sounded simple but ended up being a bit more confusing than expected. The code uses a lot of .inc(include files) to accomplish things so this meant figuring out which includes were relevant to the start page, which weren't, and then transferring it over to the admin panel. This is the completed start page, under the admin panel.

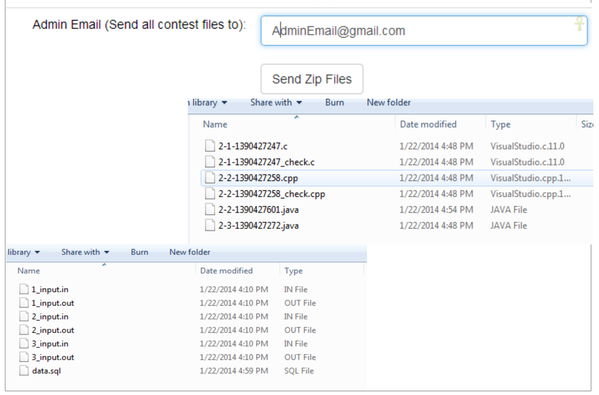


Bug Fixes

The next thing that was worked on was various bug fixes and very small changes. This included removing the option to upload Postscript files to the problem descriptions. This code was so old that it gave an option for HTML and Postscript. That was no longer necessary though now that PDF upload had been implemented so we removed it. The second bug was a minor issue with some checkboxes on the setup_contest.php page not staying checked. The fix was as simple as making sure the code refreshed and updated with the correct database value that tied to those checkboxes. The last bug that was fixed was with the possible error status's the judge could set on a submission. For some reason the "Error (Reason Unknown)" and the "Accepted" option were swapped. Meaning that the Accepted option was set to red text and marked the submission as wrong instead of the Error option. By switching these in the database it was able to quickly resolve the issue.

Add Email Functionality

The second major feature that I implemented was the send files to team and admin functionality. The previous version of the code claimed to have this feature enabled but when tested it didn't actually do anything. Essentially this feature takes the emails that the teams entered and the email the admin enters for himself, and sends a bunch of contest files to each of them. For the teams it takes every input, output file and all judged files of that team. It then tars and zips them and emails them to the address tied to the team. The admin part of this feature is exactly the same except instead of just grabbing the files for one team it takes everything from all the teams, tars them and send it all to the admin email. The feature uses mutt and then sends the mail using the server "sendmail" functionality. I had to do a bit of research and found that "sendmail" on it's own cannot handle attachments, and obviously in this case we need to attach the zipped files. So I had to set it up with mutt and install it on our server. I caught various errors with the original code and I believe that's why it wasn't working. Once I implemented this feature some other members of the team also added in an extra file to the tar ball. The code now dumps the entire database to a file and sends it to the admin. This feature was originally not slated to be worked on because of time constraints but it got moved up near the end of the project. The reason this feature became so important was because of the Live CD design. When we put this code on a Live CD the contest is entirely run from the CD, and nothing persists from one contest to the other. Once you eject the disk everything is wiped. So for archival purposes and to keep records of past contests it was vital that we implemented some way to email the important contest data from the system. Now with this feature in place it means when the contest is finished the admin could select this option and have every file from the contest in case he needed to look back at it. In the future this feature could even include more files, like the standing page or a log that lists all submissions and other activity during the contest. Here is the email interface on the misc.php tab, and the files structure when the files are unzipped and untared.



Improvements

In looking back at this project overall I think our team did an amazing job. We took on a project that had not been updated in almost 8 years. We started with code that was severely out of data and did not have the correct file structure that matched the working server version. Then on top of it we had to set up our own server to work correctly with the files. But with all those obstacles my team did a great job of adapting and turning out a very well done project. With that being said though there are as always a couple things I would have tried to be better at. The first thing is doing better at managing which issues get worked on first. I did a good job of ordering them during the first couple days, but when it came time for people to take certain tasks I wasn't always strict enough in assign them. As a result one of the high priority tasks, of improving compiler access for the teams so they can work with compiler errors, didn't get implemented. In our defense though we didn't have a ton of time and a lot of features took longer than expected.

The only other thing I would have done differently was more extensive testing. I by no means feel like we are releasing an untested product, but I think we could have ran it through a bit stricter testing. Especially when it came to actual contest judging and confirming that the computer judge was working as it is supposed to. We didn't a chance to test if any unique cases would cause the compiler or queue to break. And while yes in an actual contest the chances that anything deviates from out test cases is rare, I still would have liked to have tested it.

Conclusions

Other than the few things I mentioned above I feel very proud of the product we have completed. Our goal throughout was to make improvements but also document everything we did so in the future other contest administrators or the next developers to work on it would have an easier time setting it up and getting it configured. I tried to keep in mind the entire project how this will be used in an actual contest and how administrator, judged and team participants would respond to our changes. And I hope my confidence is proven correct as this code is used in future contests.

Acknowledgements

Matt Goldsberry
Tyler Garcia
George Harwood
Alexander Wagner
Caleb Beaverson