Lanranforces: A Better Online Judge

Executive abstract

Online judge system is a online platform for code judgment. This paper analyses some common drawbacks of existing online judge systems, and introduces the main features and advantages of Lanranforces — a better online judge system developed by our team. This paper also shows the use case diagram, ER digram and a simple preview of Lanranforces' web page.

Team

Li Haonan (11712510), Peng Ke (11612003), Chen Yiming (11712225), Zhu Xingyu (11711724).

Description

Motivation

Let's start with 2 stories that happened around us.

One day, Little Haonan wanted to do his Data Structure and Algorithm Design homework on an online judge system. Fortunately, his code passed all the test data in the system, and he got an "accepted" as a verdict. However, when he was chatting with his friends, he found that there is still a little bug in his code that was not found even by the system! This fact really disappointed him because he thought he solved a difficult problem but actually, he didn't. As you can see, this is a very common problem in our daily life, because even the test builder cannot take every possible mistake that a student could make into account.

And here is another story. Little Xingyu is one of the administrators on an online judge system and one day, he would like to publish a set of problems on it. He knows that there are many STL (standard template library) functions wrapped in C++/Java but he does not want students to use them. He thinks the best way to learn a complicated data structure is to write it by yourself, instead of just use wrapped STL functions! But the only way to ban these functions is to announce the prohibition in the QQ group, which is of low efficiency and reliability. This is another common problem that we could meet while using the online judge system.

We can provide a better way to do this.

Feature Description

Our solution is a better online judge system with some special functions. Our team decides to develop an online judge system that enables students to

- Make a hacking attempt

Making a hacking attempt means that when a student solved a problem, he can click and read other students' solution. After this, if he finds any potential bugs in the code, he can try to "hack" this solution. The student who wants to hack a solution needs to provide a test data that may trigger the bugs and if it does, it is a successful hacking attempt. All test data that triggered a bug successfully will be added to the data set in order to make it more perfect.

- Complete a code segment

  Instead of writing a complete program, some of the problems only require students to complete a single code segment, or in other words, finish an application programming interface. That is, some variables and arrays are given as parameters, and a result must be assigned as the return value of the function. In fact, this may be closer to our future work and will make students get more adapt to such a multi-person cooperation process.

And also, the online judge system will enable administrators to

- Ban some STL functions

  To ban a STL function means that when an administrator wants to publish a set of problems, he can set some constrains to the solutions submitted to the online judge system. For example, if the administrator wants students to write the data structure "queue" by themselves, he can add a compile command to the judger, and make the code segment `#include <queue>` invalid. Therefore, every solution with this code segment will get a "compile error" as a verdict.

In addition, our online judge system

- Supports multiple programming languages

  Unlike other online judge systems, our system supports many kinds of languages, like Java, C/C++, python3 and Kotlin, providing diversified programming choices for users. So if a student wants to learn a new programming language, try to solve some problems with it on our online judge system would be a pretty good choice.

- Supports special judge function

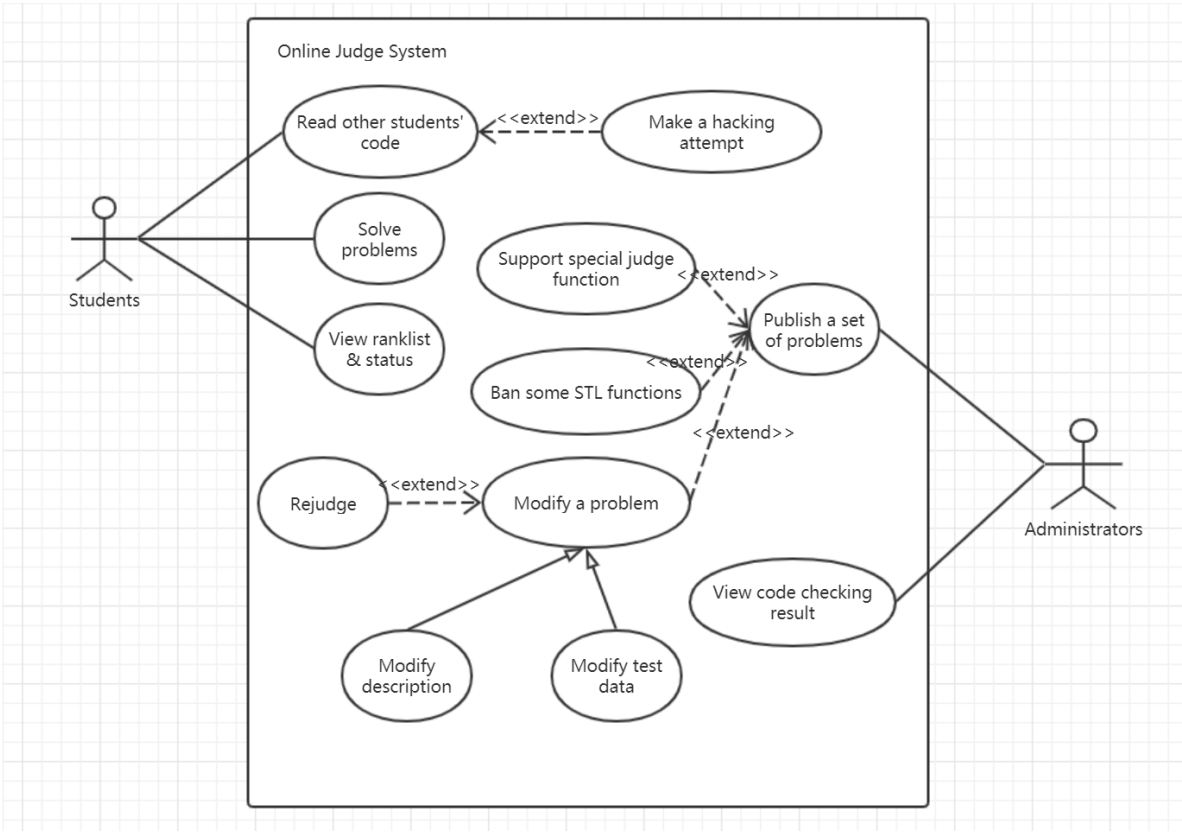
  As we know, some problems may have multiple solutions, and any of them could be considered correct. And students may need to deal with numerical precision while doing some problems related to math, so the output which is close enough to the correct answer could also be considered correct. In this case, we need special judge mechanism other than just comparing the output with the standard one. Our online judge system provide support to this special judge function.

- Uses Stanford Moss to do plagiarism detection

  We choose Stanford Moss as the tool to do plagiarism detection. Better than string comparing, Moss is able to find duplicate code segments with variable substitution and code refactoring. But Moss is not perfect as well, so Abstract Syntax Tree is also an option for us.

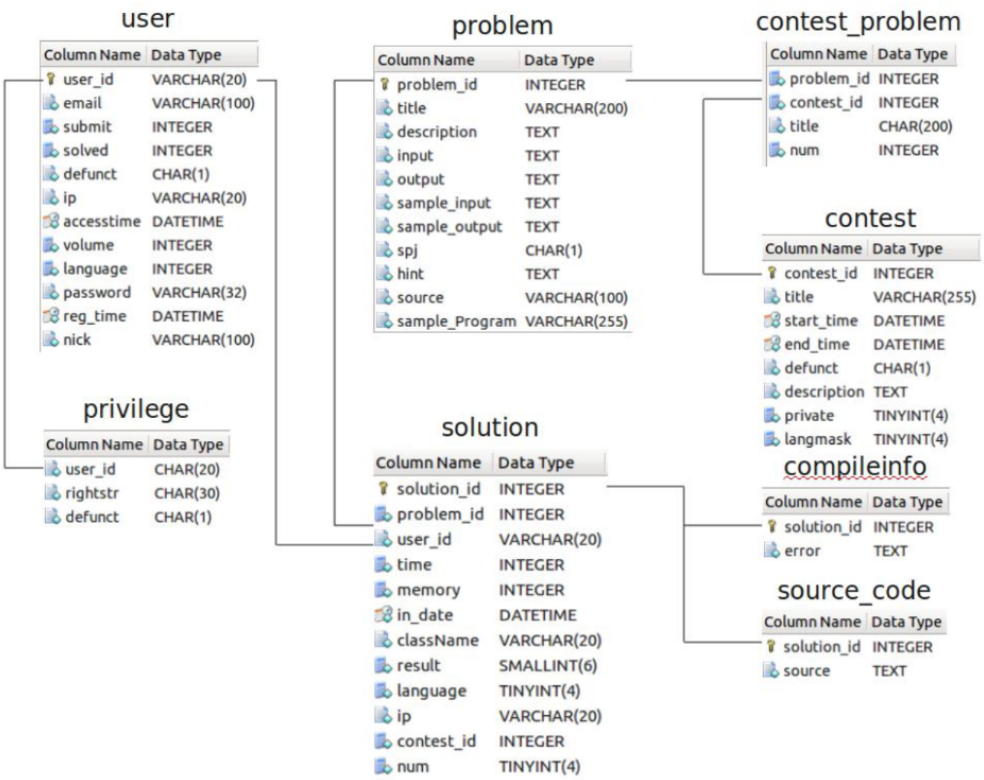
Unified Modeling Language

Here is a simple use case diagram of our online judge system. To make it clearer and prettier, we just ignore the use case “log in” since every use case includes it.



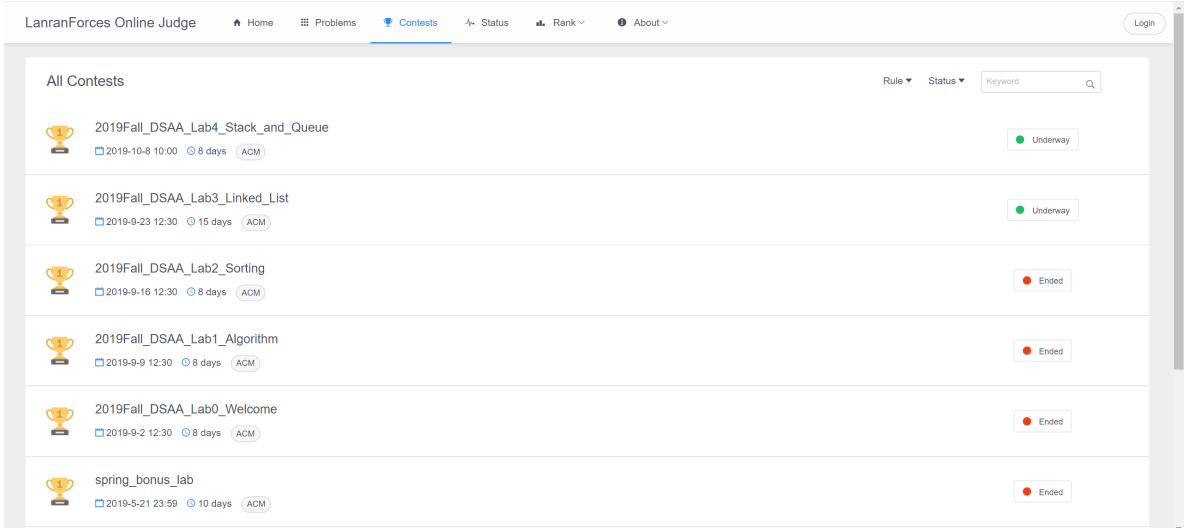
Database ER diagram

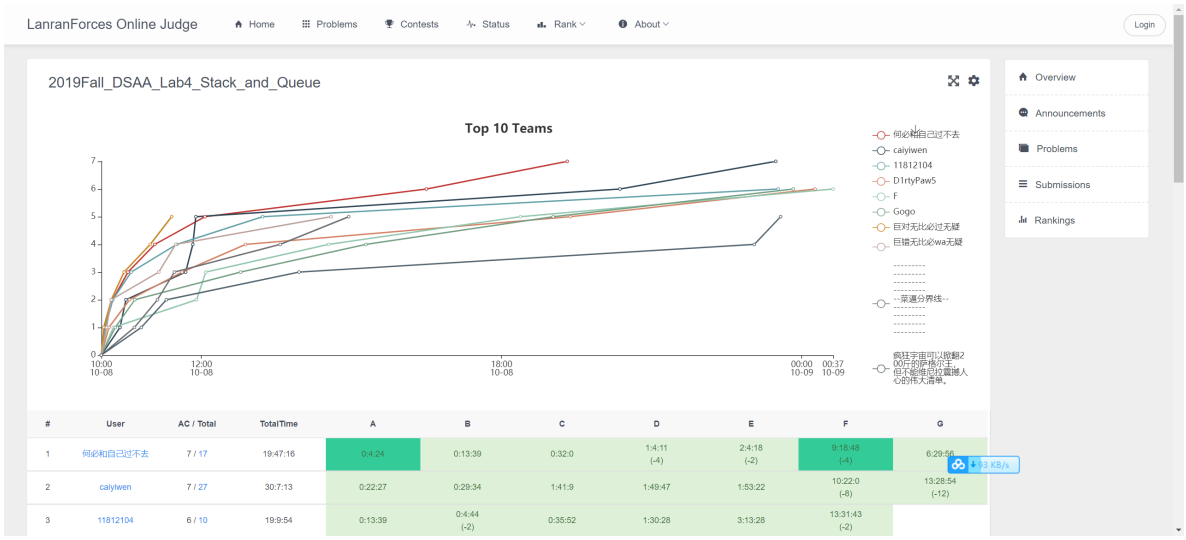
The picture showing below is a simple ER diagram of our database. We may add or modify a few tables in our future work.



Preview

The front-end of our online judge system is based on qdu online judge. But this is the first version, and we would make a lot of changes to it.





Timeline

- Back-end: The user part is finished now, and the manager part will be finished in the 7th week (person in charge: Li Haonan).
- Front-end: Continuous changes will be made until the end of the project. Several new functions (like the function of determining whether the code length exceeds limit) will be added (person in charge: Peng Ke, Zhu Xingyu).
- Judger: This part will be finished in the 8th week (person in charge: Chen Yiming).

Starting from the 9th week, performance optimization and the support for multi-process & multi-tester will take up most of our workload.