Final Report for USM Online Judge

Cory Brown
Lucas Thompson
Group Project, Software Engineering II CSC424
Computer Science Department, University of Southern Mississippi

Submitted to—
Dr. Jonathan Sun
Computer Science Department, University of Southern Mississippi

Table of Contents

Section **Page**

Statement of Problem

The University of Southern Mississippi College of Computing would like an Online Judge website that gives students a portal to complete challenging programming exercises.

Background

Online Judge websites allow a user to select a programming exercise and submit code that fulfills the exercise's tasks. The exercises can cover a broad range of topics but typically include: string manipulation, bit manipulation, data structures, and many types of algorithms (searching and sorting of arrays, trees, and graphs). The exercises usually employ constraints of speed and/or memory usage.

Original Design

Overview

The design will be a straightforward dynamic website that uses a complementary database. The main focuses will be a pleasing user experience and strong data security.

Backend

Providing the website services will be a Virtual Private Server running a LAMP stack consisting of Linux, Apache, MySQL, and Python. The server will be hardened with a firewall and the user account in charge of compiling user code will have limited permissions so to prevent access to the internet or a hijacking of the server through malicious code. In addition to the database there will be a user submission save area.

In summary: VPS, LAMP, limited user access, storage for submissions

Frontend

The frontend will be powered by a python module named Flask. It will take http requests and forward to the correct page handling classes which will then create the response to send to the user's browser. Python and Flask will also handle connecting to the database and providing a cache system to reduce database hits and speed up the website performance. Flask comes with a html templating system installed, Jinja2, and we will be templating all the html for the site and using CSS styling

In summary: Python, Flask, Jinja2, HTML5, CSS

Final Implementation

We followed our initial design very closely. During implementation we did make use of several public open source libraries for the flask platform.

Login

For our login system, we decided to not store use credentials in our on site database. Rather than deal with security, we decided to use the open standard for authorization (OAuth) provided by several other websites. In the final implementation, we included login credentials from the websites Facebook, Google, and Github. A user can login simply by granting our application access to their profile. Our website then sends a query to the relevant website using OAuth to request the users email. If the user is logged into their relevant account, it sends a response back to our website with their email attached. We then log the user in using this email. This gives us the benefit of letting these well known websites handle security, so we can focus on other aspects of the website.

Profiles

Once logged in, a profile is created for the user. The user can enter their name, an email, their location, their homepage, their company, and their school. The user then has the option of making this profile available for other users to view. The user also has the option to delete their account.

Statistics

Once a user has completed any of the exercises. Statistics are generated on their attempts. The exercise, the time it took to run the users code for that exercise, the number of attempts on that exercise, whether their last attempt was a success of failure, and a link to the source code for the last successful attempt are provided.

User Search

We offer users the ability to search for other users profile via a search bar. A user can be searched by any term available in their profile such as name, email, location, company, or school. All user profiles are indexed from the database to make this search fast and to allow searching of partial and multiple terms. Any profile that matches a searched term is returned in the search and the user has the option of visiting any of them.

Exercise List

We provide a list of exercises for a user to attempt. Once an exercise is selected, the details of the exercise are displayed. Below those details is a text editor that the user can type or load their code into to submit. Once the code is submitted, it is compiled, run, and the output is checked. Depending on the output of the code, the relevant statistics for that user are updated.

Text Editor

We provide a text editor for users to enter their code live. The text editor provides syntax highlighting, formatting, search, and various other IDE type features. The user has the option of loading a file from their computer and have it displayed in the text editor. Once the code has been entered into the text editor via typing or loading a file, it can be submitted for checking.

Compiler

When a user submits code for an exercise, it is uploaded to a directory on our server. We have a CRON job running that constantly checks this upload directory. If it finds a file, it compiles it, verifies the results, and posts it to a results directory. It also saves the source code to a storage directory. The website then retrieves the file from the results directory and displays the results to the user.

Admin Tools

There is also a page that administrators can use to make the addition and editing of page content and exercises easier. Each page has a form where the text content can be edited and submitted easily to change a web page's text. Also, there is a form for easily editing a current exercise or adding a new exercise.

Summary

In summary, our project was a success. We have basic functionality plus several other optional features. Before going live, there is one large issue and several small bugs that would need to be worked out. The only major hurdle remaining before our site would be ready for the general public is a security feature to scan any submitted code. It is a bad idea to directly run an anonymous users code. The compilation process has restricted privileges, but another step would need to be taken.

Our security plan was to convert every compiled file to its assembly file representation prior to running a user's program. That assembly file would then be scanned for system calls that should not be allowed. For example, no system calls for opening a file should be allowed, so if those were found, we would throw an error.

The implementation of this would be easy, and we already have a class in the code designated for it. However, compiling the list of system calls would be a large task, and we were unable to complete it by the end of the project.

Once this was accomplished, with some minor tweaks to our code and design, all our website would need is a large amount of content.