Welcome to Algorithm Arena

Overview

Our project is an algorithm testbed. Users can solve coding problems like those on Leetcode or other coding problem platforms with a more gamified perspective.

In the future, we plan to have the user be able to also upload their own problems and start a competition session between multiple people. So stay tuned!

How to Run

Prerequisite

Because this is a server application instead of a single-user application, you will need to obtain authentication keys for some functionality (most of them) to work. You can contact one of us to request these keys

1. Albert Jing
2. Geoffrey Jing
3. Christo Polydorou
4. Nguyen Tran

The Convenient Way

On Mac and Linux

1. Install Docker. You can find instructions at Docker website.
2. Open the terminal to where you have cloned this repository.
3. Run command `` `chmod +x run.sh` ``. You only need to run this command ONCE and never again.
4. Run `` `./run.sh` ``. The browser will automatically open. You might need to wait 40-50 seconds before the page loads if this is your first time. Subsequent times will only require less than 5 seconds.
5. From the second time onward, you can just run `` `./run.sh` `` without steps 1-3.

On Windows

See The Docker Way right below.

The Docker Way

On All Systems

1. Install Docker. You can find instructions at Docker website.
2. Open the terminal to where you have cloned this repository.
3. Run command `docker compose up --build`.
4. Open your browser at `http://localhost:8080/`.

Functionalities

Testbed

At the home page, the user will be able to choose any of the ten hand-picked problems we have available with varying difficulties: easy, medium, and hard*. Once a problem is selected, we provide an IDE-styled codebox where users can write their coded solution to the problem, and then press submit and wait to see if their solution passed all the test cases. After pressing submit and once the solution is finished being evaluated, the user will either get a success statement telling them they passed and notifying how long their code took to run on average per test case, a fail statement letting them know which test case they failed on, their output and the correct output, or a different fail statement notifying the user that their solution timed out, which means it took too long to run their solution against our test cases.

*Note: the problem might take a second or two to load and appear when first clicked into its page.

Log-in System

At the home page, the user can also log in or sign up to track their progress. The database (which is AWS-hosted) will keep track of each individual's progress. The webpage also remembers your log-ins using cookies.

Once the user has solved enough problems to be on the leaderboard, the user will see their names and the problems they solved showing up on the leaderboard. Currently, our leaderboard only shows the top five users.

Our Multi-container Structure

Our project is structured into three containers:

1. React Main App (`react-app` folder)
2. Database (`Database` folder)

3. Evaluation (`Evaluation` folder)

React Main App

This container handles the front end and communications with other containers. It's the main piece that connects everything together.

Upon loading, the app calls a Database's endpoint to retrieve the problems and their difficulties. Displaying them into three categories on the homepage.

When the user chooses a problem, the app calls another Database's endpoint to retrieve the description of the problem, as well as the starter code.

When the user submits their code, the app sends the code in a JSON object to an Evaluation's endpoint that judges the code for its correctness and returns an average run time.

When a user logs in or signs up, the app also calls the respective Database's endpoints to update our AWS MySQL server. Similarly, when a user requests to see the leaderboard, the app also calls a Database's endpoint to retrieve the users on top of the leaderboard, then display it on the web browser.

Database Container

The database container handles all communication with our AWS MySQL server and our MongoDB server. Its endpoints perform

1. Getting a problem's description
2. Getting a problem's arguments (than can be used to generate the header's code)
3. Get test cases and the correct answers to supply the Evaluation container for each problem
4. Authenticate and sign up an user when the React main app sends a combination of username and password
5. Get the leaderboard
6. Update the leaderboard when the user solves a problem successfully

Evaluation Container

The evaluation container is the heart of our technology. When given the code of the user, it calls a Database's endpoint to get the test cases and the corresponding correct answers.

Then, it runs the user's code against each test input, comparing it with the expected answer, while counting down to guard against inefficient code that takes too long for a task or poorly written code that has stuck itself on a while loop.

Finally, it calculates the average run time for each test case, and returns the result to the React main app.

Correct Code for Testing

When playing around with the application, the user can access the correct solution to all of our problems at `./Q&A/solutions`.

Acknowledgement

Our leaderboard functionality wouldn't have been possible without the help of team Scramble (John, Sunny, Artem, and Roo). They built the React page for our leaderboard.