**Online Judge**

**Problem Statement:**

­­An Online Judge is a platform where users can put their problem-solving skills to the test by solving various problems with difficulty ranging from easy to hard.

The user can perform the following actions:

* Select a problem from the problem list on home page.
* See the set the problems they have solved or attempted from the problem list.
* Go to the page of a specific problem where they can read the problem and attempt it.
* Get the verdict of their submitted solution by running the user-code against a pre-determined set of test cases specific to the given problem.
* Other than submitting, users should also be able to run their code against their own custom test case.

**UI screens/routes:**

* Home page (after login): Has a list of problems and a filter option to sort them based on solved status or difficulty.
* Problems page: When clicked on any problem on the Home page, the problem page is opened which has the problem name, description, input field to code, a console to provide custom test case, a run button that runs the code with provided test case, a submit button and a dynamic banner to display the verdict.
* Leaderboard page: Display top 10 on the leadership board based off total scores. [OPTIONAL]

**Database Designing:**

* Users:
* UserId(default)
* total\_score: float
* Problems:
* problemId
* description: CharField
* difficulty: CharField
* solved\_status: CharField: solved/unsolved
* Score: float
* Test cases:
* ProblemId (foreign key)
* Input: CharField
* Output: CharField
* Submission:
* problemId (foreign key)
* Timestamp: timestamp
* Verdict: CharField

**Web Server Designing:**

* List Problems:

**Frontend**: Create a simple list UI in React that displays the names of each problem and links them to individual problem pages.

**Backend**: Define an API endpoint in Express.js that handles a GET request to fetch all problems from the database and return them to the frontend.

* Show Individual Problem:

**Frontend**: Design a template in React to display the problem name, statement, and a submission box for problem code in text format.

**Backend**: Define an API endpoint in Express.js to handle a GET request to fetch the problem details from the database and return them to the frontend.

* Code Submission:

**Frontend**: Include a submit button below the code submission box in the "Show Individual Problem" template.

**Backend**: Define an API endpoint in Express.js to handle a POST request from the frontend. This endpoint should execute the following steps:

* Retrieve the test cases (input and expected output) for the problem from the database.
* Evaluate the submission code using a local compiler or interpreter from the backend. You can use child\_process or a similar library to call the system command for compilation or execution.
* Compare the outputs from the compiler/interpreter to the expected outputs of the test cases.
* Save the verdict for this submission (e.g., "Accepted," "Wrong Answer," etc.) in the database.
* Return the verdict and any other relevant data to the frontend.
* Leaderboard:

**Frontend**: Create a list UI in React to display the verdicts of the last 10 submissions.

**Backend**: Define an API endpoint in Express.js to handle a GET request for fetching the solutions along with the verdicts for the last 10 submissions from the database.

**Code evaluation system (Task 3)**

**Docker**

We use special containers running on machines with high CPU to run the submitted code. Code sand boxing is necessary so that the executions don’t consume too much of the resources. These executions should have the appropriate privileges set so that the code doesn’t peek into system config and time limits should be set to detect any malicious attacks that could happen from the resulting code. To do this, we can follow the given steps:

* Setup a docker container for the specific compiler (eg:- docker container
* with GCC installed[: https://hub.docker.com/\_/gcc](:%20https:/hub.docker.com/_/gcc))
* Docker run —name my-gcc -d gcc
* Evaluate the code in the Docker container using the docker exec command (<https://docs.docker.com/engine/reference/commandline/exec/>)