**High Level Design of Online Judge**

1. **Database Designing:**

* **Collection 1: problems:**

Document Structure:

1.problem statement(string)

2.problem name(String)

3.problem id(numeric)

4.difficulty level(String)

* **Collection 2: solutions:**

Document Structure:

1.problem id(string)->foreign key

2.verdict(String)

3.solution id(numeric)

4.solution(String)

* **Collection 3: test cases:**

Document Structure:

1.problem id(string)->foreign key

2.input (String)

3.output(String)

4.difficulty level(String)

* **Collection 4: User login:**

Document Structure:

1. userId(string)

2.password(String)

3.email id(numeric)

4.Full Name(String)

5. Type(Admin/User)

**2. Web Server Designing :**

* **UI:**

**Screen 1:** Home Screen

Problem List

Login/Signup

**Screen 2:** Specific Problem Language selection

File Selection

Coding Arena

Verdict / Submission

**Screen 3:** LeaderBoard(Optional) List of top performers

● 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 (MongoDB)

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.

**3**.**EVALUATION SYSTEM:**

**DOCKER :**

Use special containers running on machines with high CPU

to run the submitted code. Code sand boxing is necessary

so that the executions.

doesn’t consume too much of the resources

should have the appropriate privileges set so that the code

doesn’t peek into system config

should have time limits set

**Other Features :**

Plagiarism Checks(using softwares like MOSS)

Cache Handling