**Online Judge:**

**Problem Statement:**

An online judge platform is a platform that allows programmers to submit code solutions to programming questions. The platform should automatically evaluate the correctness and efficiency of submitted code against a set of test cases(some revealed and some unrevealed). Additional features should include user authentication, leaderboard, and the ability to host coding contests. Examples for online judge include Leetcode, Codeforces, etc.

**Overview:**

Designing a Full Stack Online Judge Using Mern Stack.

This platform contains different problems which helps improving coding skills of the programmers. Users generally solves the problems in different programming languages (C++, Python, Java) and submits the code. Then the application will evaluate the code and based on the result for the test cases it gives the verdict.

**Features :**

Some key features expected in the platform :

1. **User Registration :**

Participants should be able to register for competitions by providing their personal details such as name, email and password.

1. **Problem Repository :**

A comprehensive library of coding problems consists of different difficulty level, different topics, etc.

3. **Solution Submission :**

Allows users to submit solutions in multiple programming languages. The system compiles and runs the code against a set of test cases.

4. **Automatic Evaluation :**

Provides instant feedback on submissions, checking for correctness, performance, and edge cases. Displays verdict such as "Accepted", "Wrong Answer on some ‘X’th test case", "Time Limit Exceeded", etc.

5. **Leaderboards :**

Leaderboard showcase top performers within specific contests. Participants will be able to fetch it.

6**. Profile Management :**

Participants will have access to their profile, which includes personal details and their participation history. This allows them to track their progress and view their past competition performances.

7. **Practice Problems :**

Platform provides some set of problems which can be served as practice for the competitions and doesnot contribute to leaderboards and rankings.

**Challenges and Solutions :**

1. **Challenge :**

Ensuring the secure execution of user-submitted code to prevent malicious attacks.

**Solution :**

This can be tackled by using Docker. We use Docker to create isolated containers (Sandboxing). Each container is assigned a specific amount of memory and resources.

1. **Challenge :**

High traffic during contests can cause server overload and slow response times.

**Solution :**

To handle high traffic during contests and prevent server overload, we can implement a message queue system. By queuing the tasks, we can manage and distribute the load more effectively. Distribute incoming traffic across multiple servers using load balancers. Implement auto-scaling groups to automatically add or remove instances based on traffic load.

1. **Challenge :**

Maintaining a high quality, diverse set of problems.

**Solution :**

Encourage reviews from the users to expand the problem set. Continuously update and rotate problems to keep the content fresh and challenging.

1. **Challenge :**

An unauthorized person gets access to manipulate the verdicts and output on the server.

**Solution :**

Isolating our core logic using Custom Isolation. Can be implemented this API using Docker.

**High Level Design :**

1. **Database Design :**

* Collection 1 : Problems

statement: string (CharField)

name: string (CharField)

code: string (CharField)

difficulty: string (CharField)

tags : string(CharField)

* Collection 2 : Solutions

problem: reference to the problem document (Foreign Key)

user\_id : reference to the user document

verdict: string (CharField)

submitted\_at: date and time (Auto DateTime Field)

* Collection 3: test\_cases

input: string (CharField)

output: string (CharField)

problem: reference to the problem document (Foreign Key)

* Collection 4: Users

UserId: string (CharField)

Password : string(CharField)

Email : string(CharField)

DOB : Date

FullName : string(CharField)

Role : string (CharField)

* Collection 5 : Submissions

Solution : reference to the solution document

Status : string

1. **Web Server Design :**

* **UI:**

Screen 1: Home Screen

Problem List

Login/Signup

Screen 2: Specific Problem

Language selection Dropdown

File Selection

Tags

Difficulty Level

Verdict / Submission

Screen 3: LeaderBoard

List of top performers.

* List Problems :

Frontend: A simple list UI in React to display problem names and links to individual problem pages.

Backend: An API endpoint in Express.js to handle GET requests, fetch problems from the database, and return them to the frontend.

* Show Individual Problem:

Frontend: A template in React to display problem details, tags, difficulty level, and a submission box for code.

Backend: An API endpoint in Express.js to handle GET requests, fetch problem details from the database, and return them to the frontend.

* Code Submission:

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

Backend: An API endpoint in Express.js to handle POST requests from the frontend. The endpoint will:

- Retrieve test cases for the problem from the database.

- Evaluate the submitted code using a local compiler.

- Compare the outputs to the expected test case outputs.

- Return the verdict and other relevant data to the frontend.

* Leaderboard:

Frontend: A list UI in React to display the verdicts of recent submissions.

Backend: An API endpoint in Express.js to handle GET requests, fetch solutions and verdicts for recent submissions from the database, and return them to the frontend.

1. **Evaluation System :**

* **Docker:**

` - Use special containers with high CPU to run submitted code.

- Sandboxing ensure executions do not consume excessive resources by setting limits.

- Set appropriate privileges to prevent code from accessing system configurations.