Online Judge

Problem Statement:

An online judge is a web-based platform that provides a virtual environment for users to practice competitive programming and improve their coding skills. Users can also compete by registering for contests, where submitted code will be evaluated against pre-defined testcases and scores will be assigned based on the test results. Examples of Online Judge are Leetcode, Codechef, Codeforces etc.

Overview:

Designing a Full Stack Online Judge Using Mern Stack. Takes code from different users over the server. Evaluates it automatically as accepted or not accepted.

Features:

Key features expected in Online Judge design:

1. **User Registration:** Users should be able to register/create their account on the platform. Users will be able to submit their code only if they have logged in. User authentication and secure routing will be implemented using JWT.

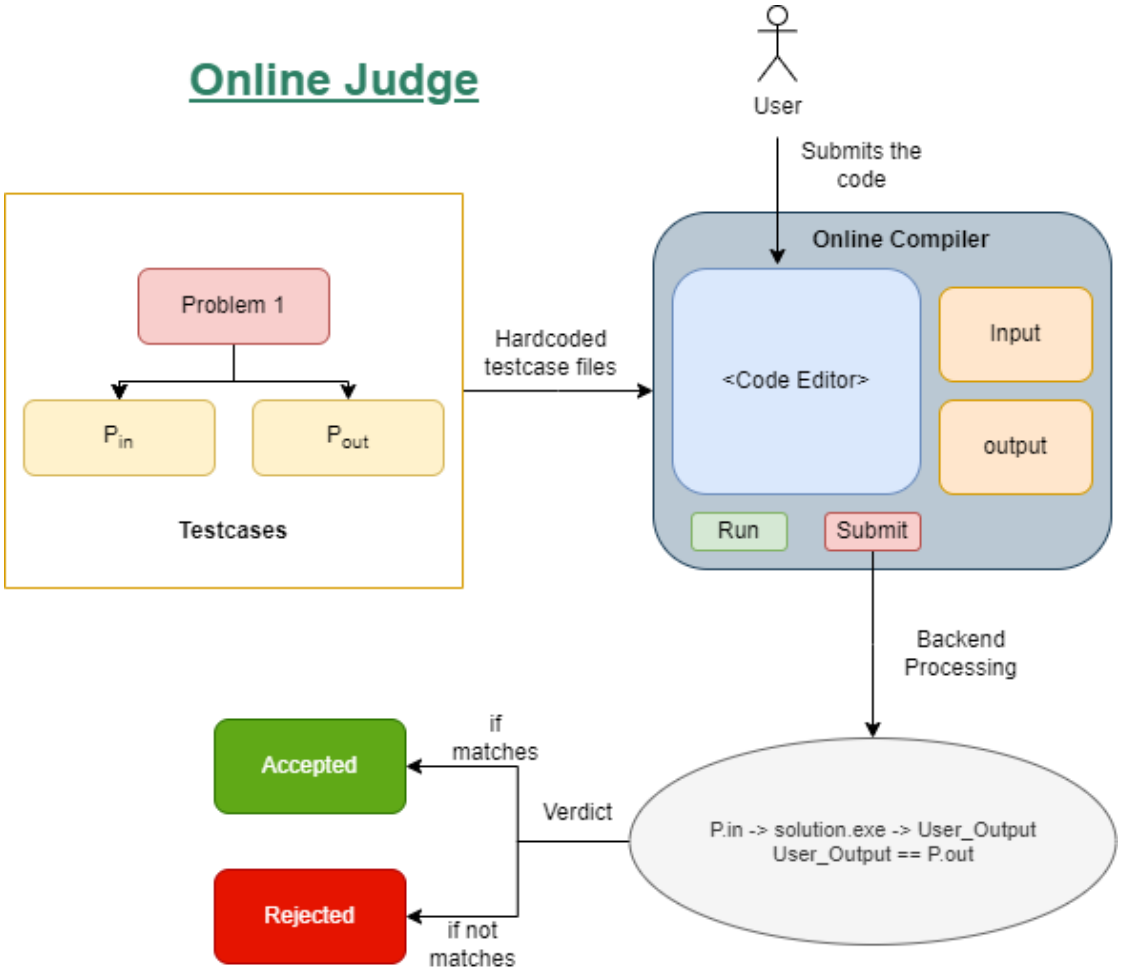
2. **Profile Management**: Participants should have access to their profile, which includes personal details and their participation history. This allows them to track their progress and view their past competition performances.

3. **Code Submission:** Registered users should be able to submit their code for the problems in practice as well as contests.

4. **Competition Leaderboard:** Participants should be able to fetch the leaderboard of a specific competition. This leaderboard will display the rankings of participants based on their scores in that particular competition.

5. **Solution Evaluation and Scoring:** The platform should have a mechanism to evaluate the submitted solutions against the underlying test cases and generate scores. This evaluation process should be automated to ensure fairness and accuracy in scoring.

6. **Practice Problems:** The platform should provide practice problems that do not contribute to the scoring or rankings. These problems can be sorted by users either topic-wise or randomly according to difficulty.



Challenges:

- Handling a large number of concurrent users and submissions, especially during popular contests or peak times.
- Preventing malicious code execution, cheating, and ensuring data integrity.
- Ensuring that the judging system is accurate, efficient, and fair.

Solutions:

- Use scalable cloud services and load balancers to distribute the load across multiple servers. Implement caching for frequently accessed data to reduce database load.
- We will use Docker to run submitted code in isolated, secure environments to prevent any harmful operations.
- Design comprehensive test cases that cover edge cases and various scenarios to ensure thorough evaluation.

High Level Design:

1. **Database Designing (MongoDB will be used)**
    i. **Collection 1:** User Login/Signup
    Document Structure:
    FullName: string(CharField)
    Username: string(CharField)
    Password: string(CharField)
    Email: string(CharField)
    DOB: Date

    ii. **Collection 2:** Problems
    Document Structure:
    statement: string (CharField)
    name: string (CharField)
    code: string (CharField)
    difficulty: string (CharField, optional)

    iii. **Collection 3:** Submission/Solution
    Document structure:
    userId: reference to Users model (Foreign Key)
    problemId: reference to the Problem model (Foreign Key)
    verdict: string (CharField)
    submitted_at: date and time (Auto DateTime Field)

    iv. **Collection 4:** Testcases
    Document structure:
    input: string (CharField)
    output: string (CharField)
    problem: reference to the problem document (Foreign Key)

2. Web Server Designing

i. UI Screens:

Screen 1: Home Screen
Problem List
Login/Signup

Screen 2: Specific Problem
Language selection
File Selection
Code Editor
Verdict / Submission Log

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

ii. Functionalities:

- List Problems
  Template: A simple list consisting of names of each problem, linking it to the individual problem's page
  View: Use Express.js to GET request to fetch all problems from Problem table and return to UI

- Show individual Problem
  Template: Show Problem name and corresponding statement along with submission box to submit problem code in text format
  View: GET Request to fetch the problem details from the problem table and return to UI

- Code Submission
  Template: A submit button below the Code submission box in the "Show individual problem" template
  View: POST request to backend to handle execute following
  - Get the test cases(Input and output) for the problem from test cases table
  - Evaluate the submission code in your local compiler
  - compare the outputs from the compiler result to the output of the testcases in db
  - save the verdict for this submission in db

- redirect to leaderboard page

- Leaderboard (Result of each submission)
  Template: A list showing the verdict of the last 10 submissions
  View: GET request fetching the solution along with verdict for last 10 solutions from solution table

3. **Code Evaluation System**
   Docker:
   - setup a docker container for the specific compiler
   - Evaluate the code in the Docker container using the docker exec command
   - Code sand boxing / isolation is necessary so that the executions doesn't consume too much of the resources.