Online Judge

Problem Statement

An online judge platform is designed to host coding competitions where participants solve a series of coding problems within a set timeframe. Participants submit their code, which is automatically evaluated against hidden test cases to determine correctness and efficiency. The platform assigns scores based on these evaluations and ranks participants accordingly. The goal is to create a robust, scalable, and user-friendly system that supports multiple programming languages, provides real-time feedback, and ensures fair competition.. Examples of online judges include 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

Here are some key features expected in the design:

1. **User Authentication:** Implement secure user authentication and authorization, allowing users to register, log in, and manage their accounts.

2. **Admin Authentication:** Ensure a robust authentication system for administrators to manage the platform, oversee competitions, and handle user issues securely.

3. **Problem Filtering by Tags and Difficulty:** Allow users to filter and solve problems based on specific tags and difficulty levels. This feature helps users to focus on problems that match their skill level and areas of interest, making their learning and practice more efficient.

4. **Custom Input Execution:** Enable users to run their code on custom inputs successfully. Provide a feature for users to input their own test cases and see the output generated by their code in real-time, allowing for better debugging and validation of their solutions.

5. **Code Submission:** Create an intuitive interface for users to submit their code solutions, supporting multiple programming languages and enforcing specified input/output formats.

6. **Automatic Evaluation:** Automatically evaluate code submissions against predefined test cases to determine their correctness. Provide users with feedback on the status of their submissions, indicating whether they passed or failed the test cases.

7. **Admin CRUD Operations:** Allow administrators to perform CRUD (Create, Read, Update, Delete) operations on problems and test cases. This enables admins to manage the problem sets and test cases efficiently, ensuring the platform remains up-to-date and relevant.

High Level Design :

1. Database Designing

(i) **Problems**

(a) statement: string (CharField)
(b) name: string (CharField)
(c) P_ID : number (primary key)
(d) Sample input: string (CharField)
(e) Sample output: string (CharField)
(f) Difficulty: string (CharField)
(g) Tags: string (CharField)

(ii) **Solutions**

(a) P_ID: number (reference to the problem schema (Foreign Key))
(b) Email: string (reference to the user schema (Foreign Key))
(c) Code: string (CharField)
(d) verdict: string (CharField)
(e) date_and_time: date and time
(f) Language: string (CharField)
(g) Points: string (CharField)

(iii) **Test_cases**

(a) P_ID: number (reference to the problem schema (Foreign Key))
(b) input: string (CharField)

(c) output: string (CharField)

(iv) **Register User**

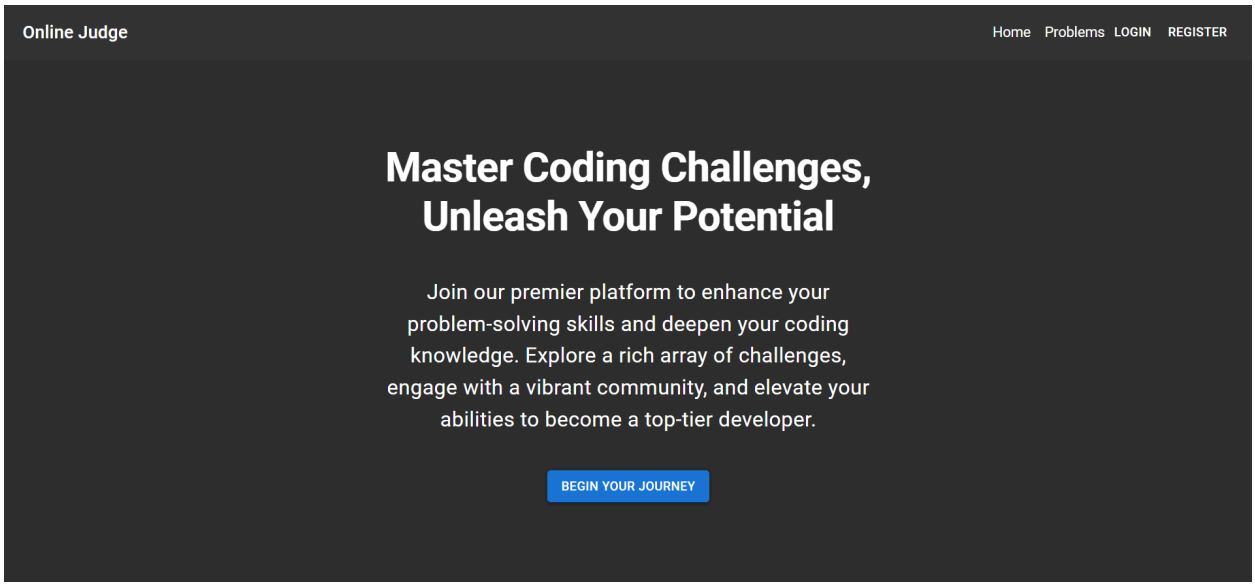
(a) Username: string (CharField)
(b) Password : string(CharField)
(c) Email : string(CharField) (primary key)
(d) Birth Year : number
(e) Country: string (CharField)
(f) Role: string (CharField)

(v) Register Admin

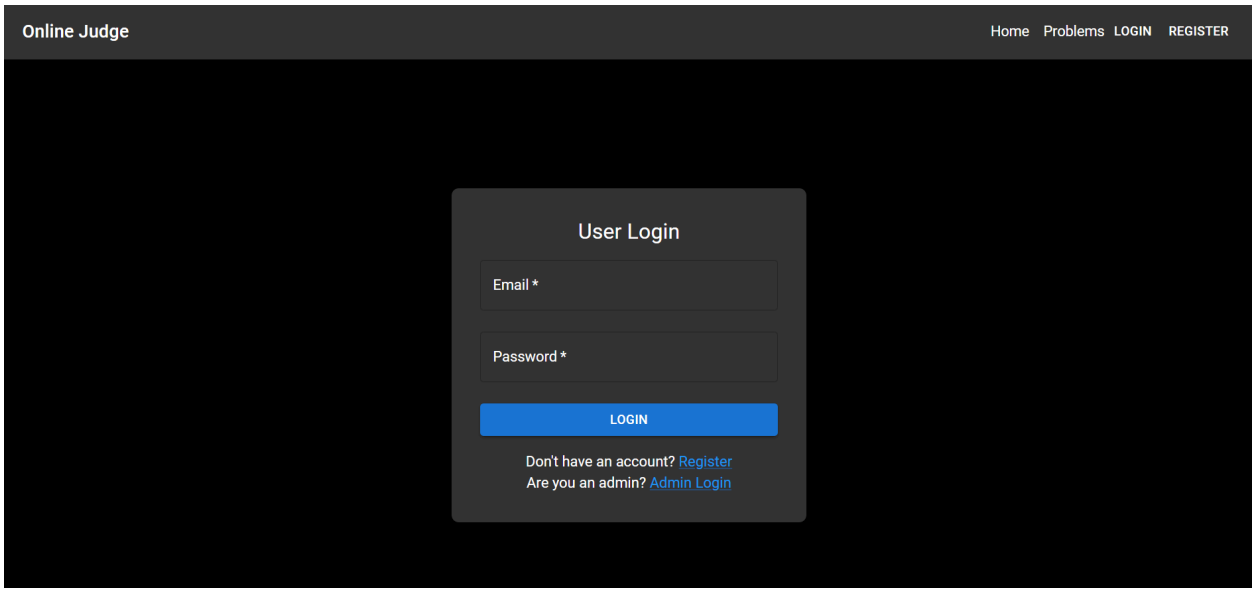
(a) Username: string (CharField)
(b) Password : string(CharField)
(c) Email : string(CharField) (primary key)
(d) Birth Year : number
(e) Country: string (CharField)
(f) Role: string (CharField)

2. User Interface

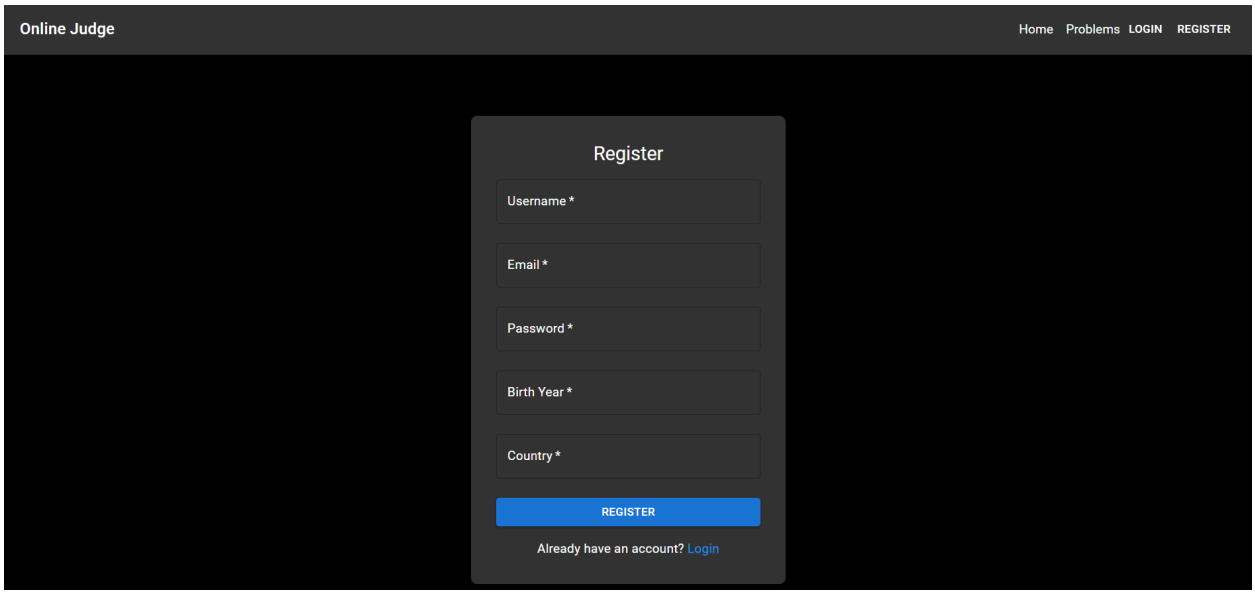
(A) Screen 1: Landing page of application



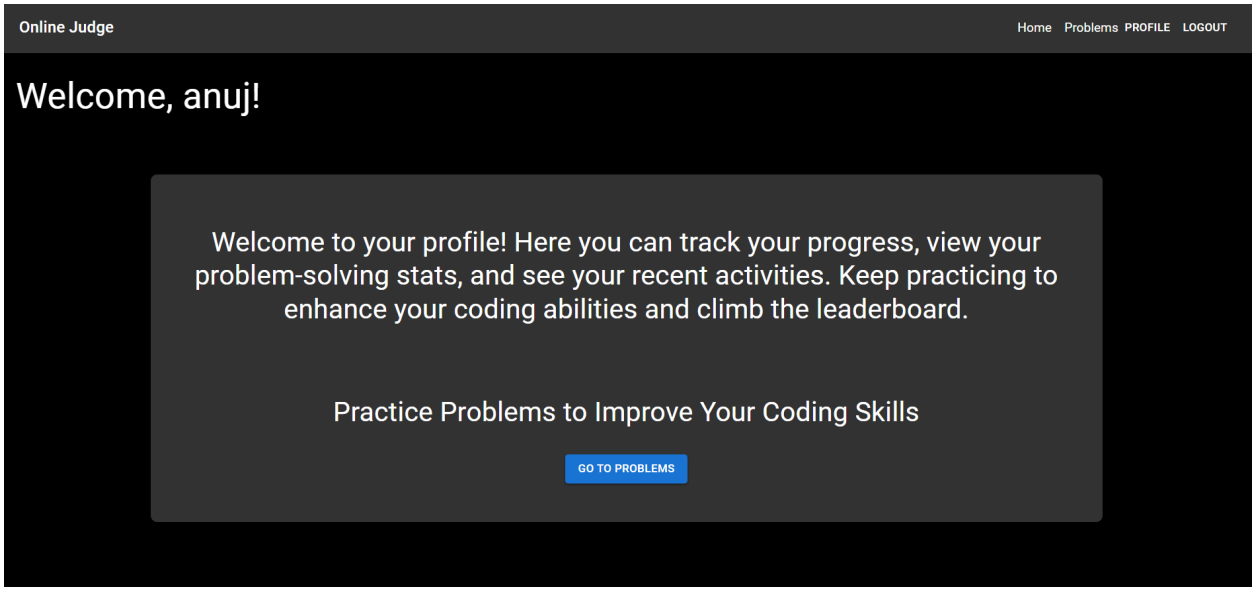
(B) Screen 2: User Login Page



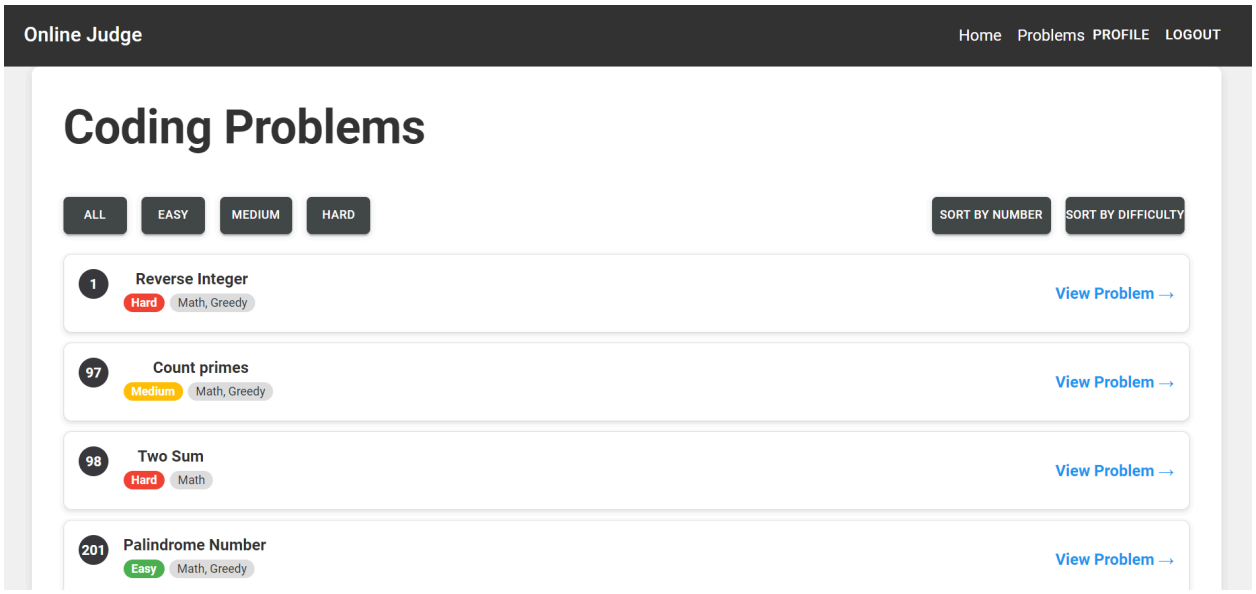
(C) Screen 3: User Registration Page



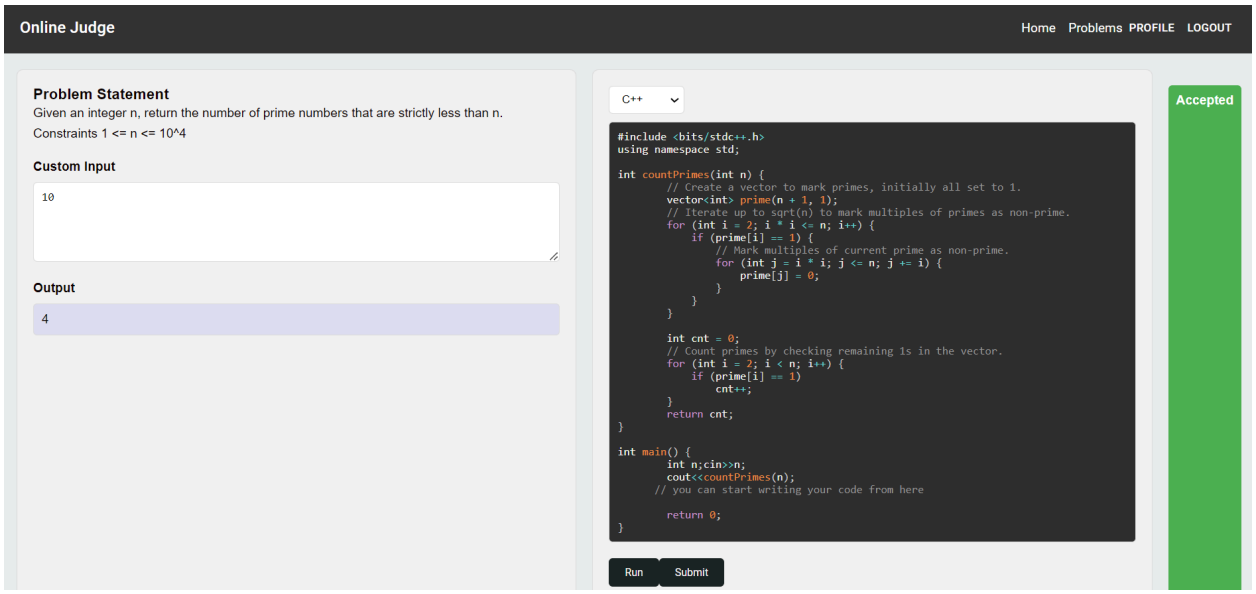
(D) Screen 4: User Profile Page



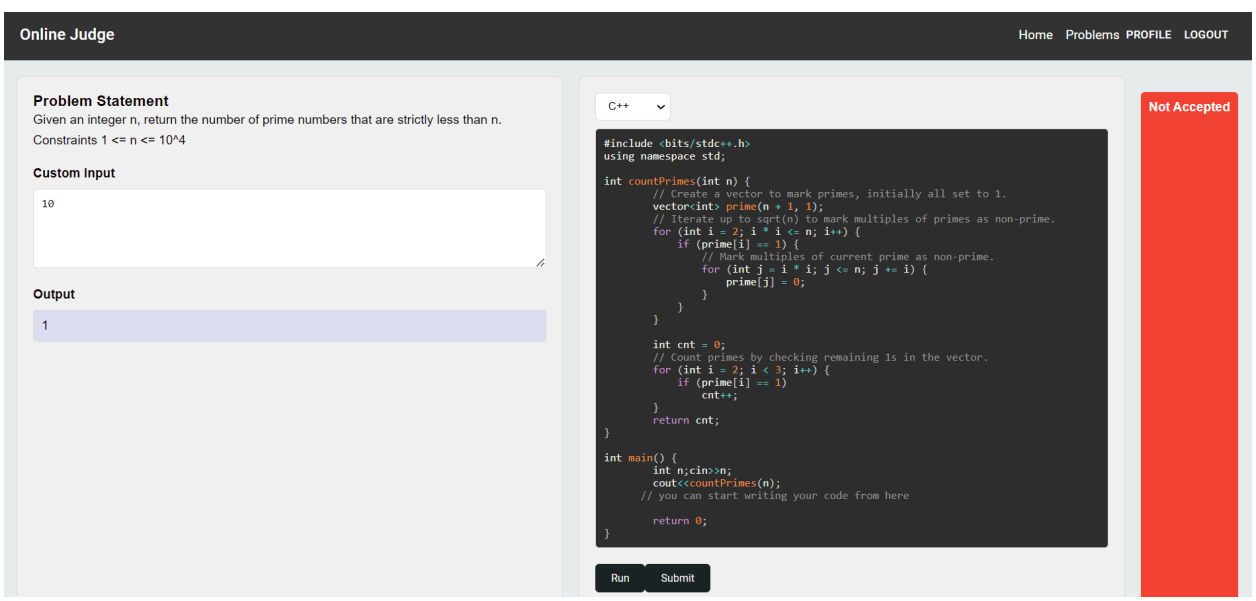
(E) Screen 5: User Problemset Page



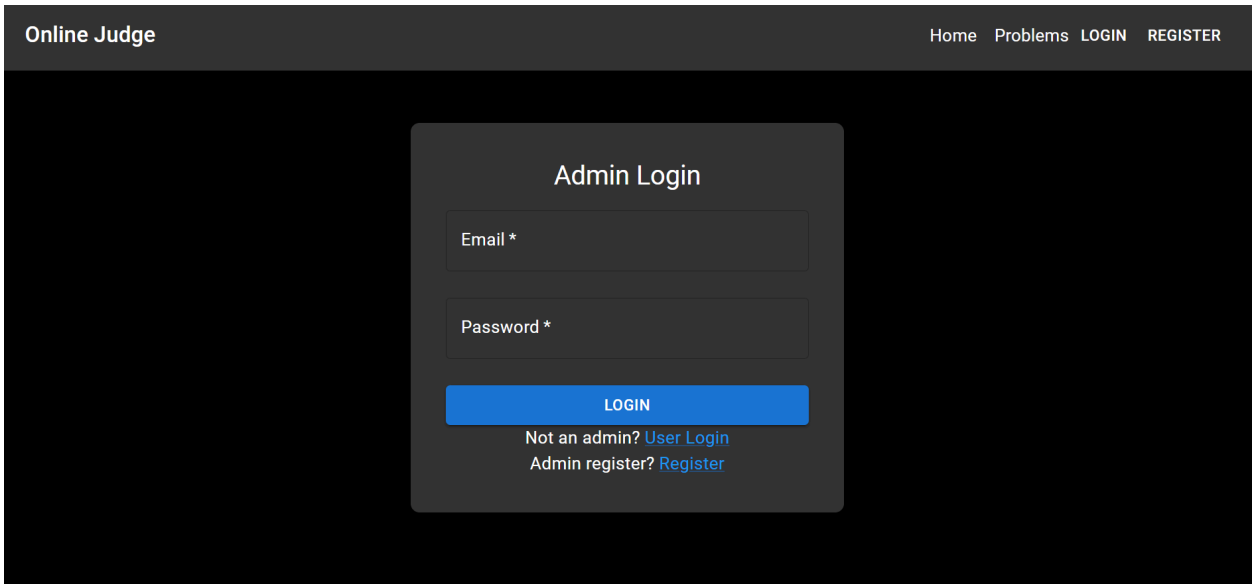
(F) Screen 6: Specific Problem Page with custom input and verdict as accepted



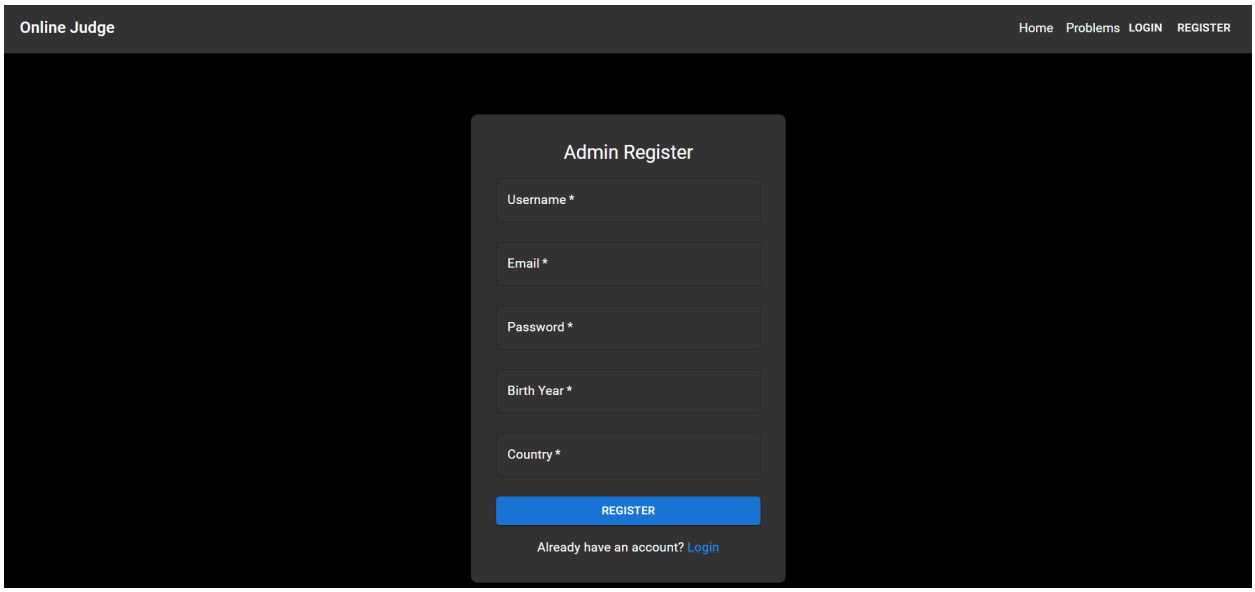
(G) Screen 7: Specific Problem Page with custom input and verdict as not accepted



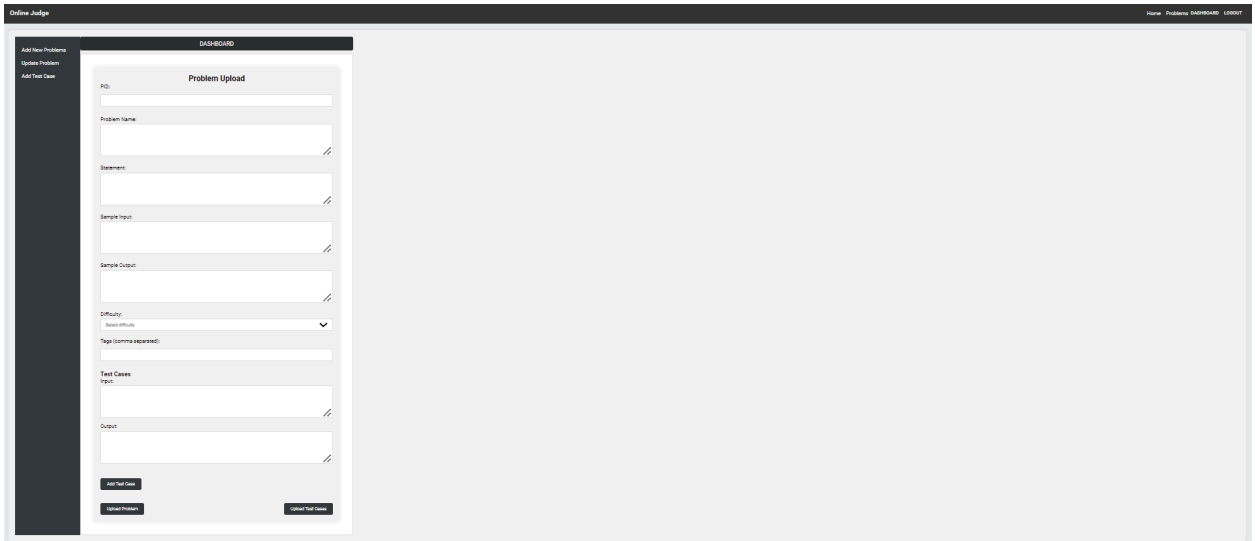
(H) Screen 8: Admin Login Page



(I) Screen 9: Admin Registration Page

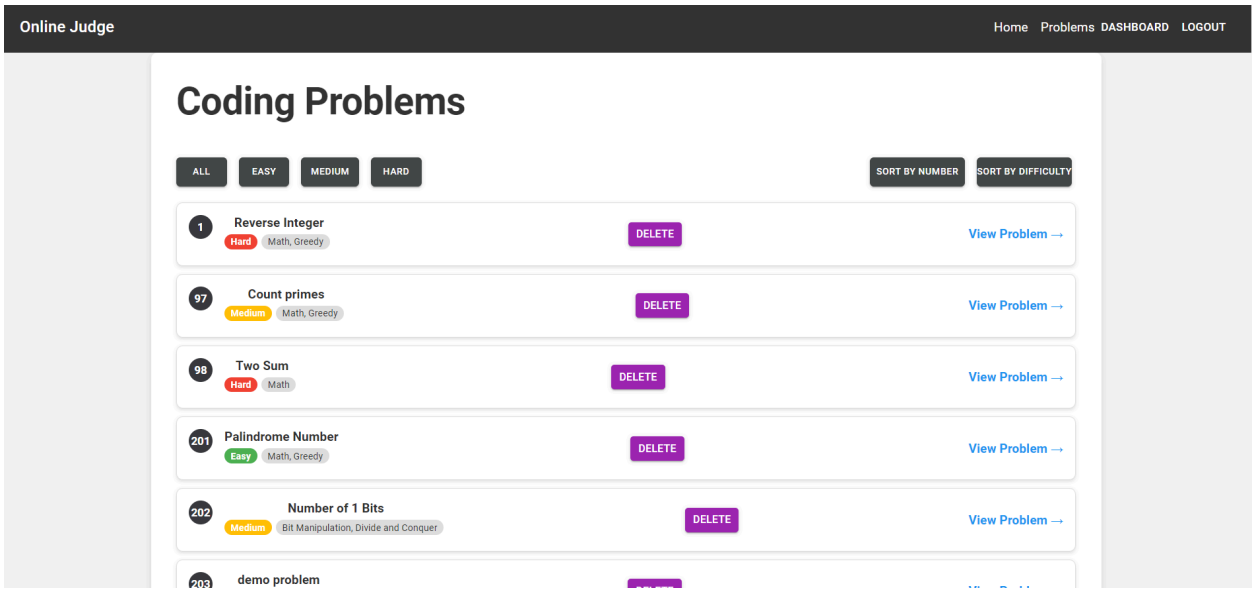


(J) Screen 10: Admin Dashboard (Create New Problems and Test Cases)



(K) Screen 11: Admin Dashboard (Update existing Problem)

(L) Screen 12: Admin Problemset



3. Web Server Designing

1. Problems List

Frontend:

- React Component for Problem List: Develop a React component that dynamically displays a list of coding problems.
- Navigation to Individual Problem Pages: Ensure each problem in the list is a clickable link that directs users to a dedicated page for that specific problem, allowing users to view problem details and submit their solutions.

Backend:

- API Endpoint in Express.js: Implement an API endpoint using Express.js to retrieve the list of all problems stored in the MongoDB database. This endpoint will facilitate the fetching of problem data for the frontend to display, ensuring the problem list is always up-to-date.

2. Show Individual Problem

Frontend:

- React Component for Problem Details: Develop a React component that displays the detailed information of a specific coding problem, including the problem statement, input/output specifications, sample test cases, and constraints.

- Code Submission Interface: Incorporate a submission box within the component where users can write and submit their code solutions. Ensure the interface is user-friendly and supports multiple programming languages.

  Backend:
- API Endpoint in Express.js for Problem Details: Create an API endpoint in Express.js to fetch the detailed information of a single problem from the MongoDB database. This endpoint will enable the frontend to request and display comprehensive problem data for users to solve.

3. Code Submission

Frontend:
- Submit Button Integration: Integrate a submit button within the individual problem template that allows users to submit their code solutions. Ensure the button triggers a submission event when clicked.
- Submission Handling: Implement logic to capture the user's code and send it to the backend for evaluation upon clicking the submit button. Provide feedback to the user about the submission status.

  Backend:
- API Endpoint for Code Submission: Establish an API endpoint in Express.js to receive code submissions from the frontend. This endpoint will handle the submission process, including code evaluation and response.
- Code Evaluation Mechanism: Utilize a local compiler or interpreter to execute the submitted code in a secure, isolated environment. Compare the output of the code against predefined test cases to determine its correctness.
- Submission Verdict Storage: Store the results of the code evaluation (verdict) in the database. Return the verdict and any relevant feedback to the frontend, allowing users to see the outcome of their submission.

4. Dockerization and Containerization

Incorporating Dockerization and containerization ensures a secure, efficient, and scalable environment for code evaluation. These features enhance the overall user experience by providing fast and reliable feedback on code submissions, while maintaining system integrity and fairness. Each code submission is executed in an isolated environment to ensure fair resource allocation, system security, and consistent execution.

Code Sandboxing:

- Run each code submission in its own isolated Docker container.
- Prevent code from consuming excessive resources and affecting other submissions or system operations.

Security and Privilege Management:

- Configure Docker containers with strict security policies to restrict access to system configurations and sensitive files.
- Isolate the execution environment to enhance security and prevent malicious activities.

Scalability:

- Add more containers across different servers to increase the system's capacity.
- Manage a large number of code submissions efficiently, even during busy periods, without slowing down.