Online Judge Platform

Problem Statement: An Online Judge platform is designed for coding enthusiasts and competitive programmers to enhance their coding skills through problem-solving. Users can solve coding challenges, submit their solutions, receive feedback on correctness and efficiency, and engage in friendly competition. Popular platforms include LeetCode, HackerRank, and Codeforces.

Features:

1. **Problem Browsing and Viewing:**
    - Users browse a list of coding problems.
    - Detailed descriptions include problem statement, name, code, difficulty level, and example test cases.
2. **User Authentication:**
    - Secure login and signup ensure safe access with encrypted user data.
3. **Code Submission and Evaluation:**
    - Users submit code solutions.
    - Real-time evaluation in a secure Docker sandbox.
    - Verdicts ("Accepted", "Wrong Answer") based on predefined test cases.
4. **Submission History and Logs:**
    - Accessible logs of recent submissions.
    - Details: verdicts, execution time, memory usage, programming language.
5. **Profile Management:**
    - Update personal info, change passwords, view submission stats.
6. **Leaderboard:**
    - Dynamic display of top performers.
    - Based on successful submissions and problem-solving efficiency.

High-Level Design:

1. Database Design

Collection: Problems

- Problem statement.
- Problem name.
- Problem ID.
- Difficulty level.
- Example test case input.
- Example test case output.

Collection: Solutions

- Associated problem.
- Solution result ("Accepted", "Wrong Answer").
- Submission timestamp.
- Execution time (milliseconds).
- Programming language.

Collection: Test Cases

- Test case input.
- Expected output.
- Associated problem.

Collection: Login/Signup

- Unique user identifier.
- Encrypted user password.
- User's email address.
- User's date of birth.
- User's full name.

2. Web Server Design

UI Screens

1. **Home Screen**
   - Problem List
   - Login/Signup
2. **Specific Problem Screen**
   - Language Selection
   - File Selection
   - Coding Arena
   - Verdict / Submission Log
3. **Leaderboard Screen** (Optional)
   - List of Top Performers
4. **Show Submissions Screen**
   - List of Recent Submissions for a Specific Problem

Functional Requirements

1. **List Problems**
   - Frontend:
     - React UI with problem names linked to problem pages.
   - Backend:
     - Express.js API fetches all problems from MongoDB.

2. **Show Individual Problem**
    - Frontend:
        - React template displays problem details and submission box.
    - Backend:
        - Express.js API fetches problem details.
3. **Code Submission**
    - Frontend:
        - Submit button in "Show Individual Problem" template.
    - Backend:
        - Express.js API handles POST requests, evaluates code using Docker, compares outputs, saves verdicts, and returns data.
4. **Show Submissions**
    - Frontend:
        - React UI displays recent submissions with details.
    - Backend:
        - Express.js API fetches recent submissions.
5. **Leaderboard**
    - Frontend:
        - React UI lists the verdicts of the last 10 submissions.
    - Backend:
        - Express.js API fetches solutions and verdicts.

3. Evaluation System

Code Execution and Sandbox Environment

Docker Setup:

- Use Docker for isolated execution environments.
- Deploy on high CPU machines for efficient computation.

Security Measures:

- Run containers with restricted privileges.
- Sandbox to prevent unauthorized access.
- Enforce resource limits (CPU, memory, disk).

Additional Features

Plagiarism Checks:

- Integrate MOSS for plagiarism detection.
- Automate checks for code similarities.
- Generate plagiarism reports.

Cache Handling:

- Cache evaluated submissions.
- Implement cache invalidation strategies.
- Optimize system performance.

Advantages:

1. **Full Stack Consistency:** Simplifies development with a unified JavaScript-based stack for frontend and backend, aiding in seamless integration and easier maintenance.
2. **React for Interactive UI:** Facilitates efficient creation of responsive user interfaces, crucial for managing complex UI elements and enhancing user experience.
3. **Express.js for Backend:** Provides a robust framework for building RESTful APIs, essential for handling user authentication, data retrieval, and submission evaluations.
4. **MongoDB for Database:** Offers flexibility in managing unstructured data and scalability for large volumes, aligning well with JavaScript objects and intuitive data manipulation.
5. **Real-time Data Handling with Node.js:** Supports concurrent connections and real-time updates, ideal for features like live submission updates and leaderboard management.

Disadvantages:

1. **Learning Curve:** Requires familiarity with JavaScript frameworks (React, Node.js) and asynchronous programming, potentially extending initial development time.
2. **Scalability Challenges with MongoDB:** May face limitations with complex queries and transactions compared to traditional relational databases as data complexity grows.
3. **Security Considerations:** Node.js and MongoDB require careful configuration to mitigate risks like DDoS attacks and injection vulnerabilities, demanding robust security practices.
4. **Ecosystem Maturity:** While popular, the MERN stack's ecosystem might lack extensive libraries and tools compared to more established stacks, potentially requiring more custom development.
5. **Complexity for Simple Applications:** Overkill for basic CRUD applications, adding unnecessary complexity and overhead to development efforts.
6. **Node.js Performance Limitations:** Efficient for I/O-bound tasks but less optimal for CPU-bound operations, which can impact performance under heavy user loads or intensive computations.