High Level Design for Online Judge (OJ)

1. **What is an OJ**?
   An online judge is a platform wherein users/participants can submit code for a given problem/question and the OJ then sends a verdict (Accepted or Rejected) to the user based on correctness against a hidden set of test-cases and the time-complexity of code.
   Examples: Codeforces, Atcoder, Codechef etc.

Problem:

Task is to design an Online Judge to simulate an OJ as closely as possible.

Approach to solve problem:

<u>Frontend</u>

Mainly using React along with HTML, CSS and JavaScript (JS)

- **Register Page**: The page should ask for user details and help the user create the account.
- **Login Page:** The page should accept the Email-ID and password and redirect the user to home page if details correspond to a valid registered user.
- An option to **reset password** in case user forgets his password by sending reset link to registered Email-ID

Home page:

- Submission History**:** Display list of all submissions and option to see them by accepted, rejected or other WA (wrong answer) verdict.
- Problem set:
    - Option to sort problems based on difficulty level (Rating)
    - Option to either open the problem for immediate solving or mark for solving later.

Problem Page for a given problem:

- Display the problem statement on the left/right side of screen as might be deemed fit.
- Constraints and output for given test-case (test-case 1)
- Code Editor to write the code for that problem
- Run button to test the code locally
- Submit button to send the final code for test against hidden test-cases.
- Accordingly give the verdict

Profile

- Display the user details
- Number of problems solved
- Option to change password by providing current password

Backend
(Using NodeJS, ExpressJS and JWT(JsonWebToken) authentication)

Purpose	API Endpoint	Request type
Register	/register	POST
Login	/login	POST
Forgot password	/forgotpassword?	POST
Profile	/api1/profile	GET
Change Password	/api1/changep	POST
View problem set	/api1/viewps	GET
View given problem	/api1/viewp/problem-id	GET
Code submission	/api1/code-sub/problem-id	POST

Database:

Mainly MongoDB (NoSQL database)

- About the user:
  - FirstName
  - MiddleName
  - LastName
  - Email-ID
  - Password
- Testcase collection
  - Problem-id
  - Test-case
  - Output of Test-case
- Problem
  - Problem-id
  - Problem-title
  - Problem-statement
  - Status of problem (solved/unsolved/marked for solving later)
  - Tags
- Submission
  - User-id (can be denoted via email-id or the one given by MongoDB)
  - Problem-id
  - Verdict (Accepted/WA/TLE/MLE etc)
  - Runtime (millliseconds)
  - Space taken (KB)
  - Test-case number where the code wasn't accepted (if not accepted)

Code evaluation:

1. **Docker** – Setting up docker container for necessary compiler
2. **Sandbox isolation** – Addressing issues like
    - DDOS (Distributed denial-of-service) attack,
    - Malicious scripts running beyond Time Limits

- Network isolation (not able to use external websites)
- Not able to access hidden tests
- Performing custom isolation

User roles

1. **Admin** – Has all functionalities of website
2. **User** – Can only solve problems
3. **Problem setter** – Can add new problems on behalf of admin

OJ Deployment

1. Using Amazon Web Services (AWS)
2. More planning to be done as project nears completion