# HLD for Online Judge

## Problem Statement:

An Online Judge is a platform where users can solve coding problems and submit their solutions, receiving a verdict based on the accuracy and efficiency of their code.
Examples of Online Judge are : leetcode, Codeforces, Codechef, Hackerrank 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.

## Tech Stack Used: MERN Stack

**Advantages of using MERN Stack-**

1. For a smooth development of any web application or mobile app, it supports MVC (**Model View Controller**) architecture;

the main purpose of this architecture is to separate the presentation details with the business logic.

2. It covers all the web development stages starting from front-end development to backend development with JavaScript.
3. It is an open-source framework mainly used to develop web-based or mobile applications and is supported by the community.
4. It is very fast and efficient compared to MEAN Stack and mostly suitable for small applications, whereas MEAN Stack is suitable for developing large applications.

# Frontend :

**ReactJS, HTML, CSS.**

**Login/Register Page:**

- **Register:**
  - Use an email ID to register.
  - Create a username.
  - Set a password.
  - Re-enter the password for confirmation.
- **Login:**
  - Use your username and password to log in.
- **Forgot Password:**

    - An option to send a password reset link to your registered email ID.

**Home Page:**

- **Tabs (located at the top, side by side):**
    1. **List of Problems:**
        - Problems categorized by difficulty: Easy, Medium, Hard.
        - Options for each problem:
            - Solve
            - Review Later
            - Mark as Done (automatically marked as done after successful submission)
    2. **Submission History:**
        - A list which displays time of submission for each problem.
        - Shows verdict, time taken for the code to run, and space taken by the code.
    3. **Profile:**
        - Shows username and registered email ID.
        - Displays total number of successful submissions.
        - Options to add name and college information.

**Code Editor Page:**

- **Left Side:**
    - Problem statement
    - Few test cases with their input and output
- **Right Side:**
    - Code editor to write the code
    - Custom input field to test the code

    - Run button to execute the code and check the output in a terminal
    - Submit button to get the verdict
    - On wrong submission, displays the test case where the code failed.

# Database :

**MongoDB**

**Users Collection:**

- **Fields:**
    - **username:** Unique username.
    - **password:** Hashed password.
    - **email:** User's email.
    - **name (optional):** User's full name.
    - **college (optional):** User's institution.

**Problems Collection:**

- **Fields:**
    - **problemId:** Unique identifier.
    - **description:** Problem statement.
    - **title:** Problem title.
    - **code:** Numeric identifier.
    - **problemTags:** List of tags.
    - **status:** Problem status (done/unsolved/review later).

**Testcases Collection:**

- **Fields:**
    - **problemId:** Reference to the associated problem.

    - **input:** Test case input.
    - **output:** Expected output.

**Submissions Collection:**

- **Fields:**
    - **userId:** Reference to the user.
    - **problemId:** Reference to the problem.
    - **testCaseId:** Reference to the specific testcase.
    - **timestamp:** Submission timestamp.
    - **verdict:** Submission outcome.
    - **time taken:** Execution time (ms).
    - **space taken:** Memory usage.

# Backend :

**NodeJS, ExpressJS, JWT**

Backend API Overview

**List Problems API Endpoint**

- Endpoint: `/api/problems`
- Request Type: GET
- Description: Fetches all problems from MongoDB.

**Code Submission API Endpoint**

- Endpoint: `/api/submissions`
- Request Type: POST
- Description: Handles code submissions, evaluates against test cases.

### User Authentication API Endpoints

- Register Endpoint:
  - Endpoint: `/api/auth/register`
  - Request Type: POST
  - Description: Registers a new user.
- Login Endpoint:
  - Endpoint: `/api/auth/login`
  - Request Type: POST
  - Description: Authenticates a user, issues JWT.
- Forgot Password Endpoint:
  - Endpoint: `/api/auth/forgot-password`
  - Request Type: POST
  - Description: Initiates password reset process.

### Profile Management API Endpoints

- Profile Information Endpoint:
  - Endpoint: `/api/profile`
  - Request Type: GET
  - Description: Fetches user profile information.
- Update Profile Endpoint:
  - Endpoint: `/api/profile/update`
  - Request Type: PUT
  - Description: Updates user profile information

# Code Evaluation System

### 1) Docker

a) Setup a docker container for the specific compiler (eg:- docker container with GCC installed:https://hub.docker.com/_/gcc) docker run —name my-gcc -d gcc

b) Evaluate the code in the Docker container using the docker exec command(https://docs.docker.com/engine/reference/commandline/exec/)

**2) Sandbox**

- Use Isolate for isolation. https://github.com/ioi/isolate
- Address security challenges: DDOS attacks, malicious code, and scripts running beyond time limits.