ONLINE JUDGE

Problem Statement

Design an online judge platform where users can submit their code for evaluation and receive a verdict based on provided problem statements. The platform will feature multiple coding challenges and an integrated online compiler. Users will write and test their code using custom or predefined test cases.

Overview

Develop a full-stack online judge platform with the MERN stack . It will automatically evaluate code submissions and be deployed on the cloud for universal accessibility.

Key Features

- **User Authentication**: Participants must register on the website to access problems.
- **Problem Submission and Evaluation**: Users receive a verdict based on their submitted solutions.
- **Discussion Forums**: Create forums or discussion boards where users can discuss problems, solutions, and coding techniques.
- **Leaderboard**: Ranking users based on the number of problems solved.
- **User Profile Tracking**: Tracking and displaying user progress and achievements.
- **Custom Test Cases**: Allow users to define and run their own custom test cases to further test their solutions.
- **Multi-Language Support**: Expand support for a wide range of programming languages beyond the basics.
- **Badge and Achievement System**: Award badges or achievements based on user performance or milestones achieved.
- **Tutorial Integration**: Provide integrated tutorials or hints for each problem to help users learn and improve their coding skills.
- **Notifications**: Notify users about new competitions, submissions, or updates via email.

Frontend Design

Home Page

- **Purpose**: Display a list of problems, an overall leaderboard, and options for user authentication.
- **Components**:
    - **Header**: Contains options to log in/sign up.
    - **Problem List**: Displays available problems.
    - **Leaderboard**: Shows users ranked by the number of problems solved and attempts taken.

Login/Signup Page

- **Purpose**: log in using their username and password or sign up for a new account.
- **Components**:
    - **Header**: Logo and navigation links to the home page.
    - **Login Form**: Fields for username, password, and submit button.
    - **Sign-Up Form**: Fields for name, username, password, and create account button.

Problem Page

- **Purpose**: Provide the problem description and allow users to write, test, and submit their code.
- **Components**:
    - **Problem Description**: Includes input/output explanation and example test cases.
    - **Code Editor**: Window for writing code in the chosen programming language.
    - **Input Window**: Separate area for entering custom test cases.
    - **Output Window**: Displays results of code execution.
    - **Buttons**: 'Run' button to execute the code and 'Submit' button to check against hidden test cases.
    - **Code Collaboration Button**: Redirects to a page for real-time code collaboration.

User Profile Page

- **Purpose**: Display user's basic details and track their progress.
- **Components**:
    - **Profile Information**: Shows name, username, and basic details.
    - **Progress Tracking**: Graphical representation of user's progress and achievements.
    - **Achievements**: Displays badges or achievements earned.

Discussion Forums

- **Purpose**: Allow users to discuss problems, solutions, and coding techniques.
- **Components**:
    - **Forum Threads**: Lists discussions with topics and replies.
    - **Thread Creation**: Form to create new discussion topics.
    - **Reply Form**: Allows users to respond to existing threads.

Notifications

- **Purpose**: Notify users about new competitions, submissions, or updates via email.
- **Components**:
    - **Notification Bell Icon**: Indicates new notifications.
    - **Notification Panel**: Lists recent notifications with details.

Code Collaboration Page

- **Purpose**: Facilitate real-time code collaboration between users.
- **Components**:
    - **Shared Code Editor**: Allows multiple users to edit the same code simultaneously.

Tutorial Integration

- **Purpose**: Provide integrated tutorials or hints for each problem to help users learn and improve their coding skills.
- **Components**:
    - **Tutorial Section**: Displays hints or explanations related to the current problem.

Backend Design

Node.js: Build an application using Node.js. This will efficiently handle data storage.

User Authentication: Use JWT-based authentication to secure API endpoints and ensure secure management of user sessions.

REST APIs

Login Endpoint

- **Endpoint:** /login
- **Method:** POST
- **Request Body:**
    - username: string
    - password: string
- **Response:**
    - success: boolean
    - token: string
    - user:
        - id: string
        - username: string
        - name: string
- **Functionality:** Authenticates the user credentials, generates a JWT token upon successful login, and returns user details.

Signup Endpoint

- **Endpoint:** /signup
- **Method:** POST
- **Request Body:**
    - username: string
    - password: string
    - name: string
- **Response:**
    - success: boolean
    - user:
        - id: string
        - username: string
        - name: string
- **Functionality:** Registers a new user with provided details and returns the user's

information.

List Problems Endpoint

- **Endpoint:** /problems
- **Method:** GET
- **Response:**
  - problems:
    - id: string
    - title: string
    - description: string
    - difficulty: string
- **Functionality:** Fetches and returns a list of available problems with basic details.

Fetch Problem Details Endpoint

- **Endpoint:** /problem/:id
- **Method:** GET
- **Parameters:**
  - id: string (Problem identifier)
- **Response:**
  - id: string
  - title: string
  - description: string
  - input: string
  - output: string
  - exampleTestCase:
    - input: string
    - output: string
- **Functionality:** Retrieves and returns details of a specific problem identified by :id.

Submit Solution Endpoint

- **Endpoint:** /submit
- **Method:** POST
- **Request Body:**
  - problemId: string
  - code: string
  - language: string
- **Response:**
  - success: boolean
  - verdict: string

    - details:
        - passed: integer
        - failed: integer
        - total: integer
- **Functionality:** Accepts user's code submission, evaluates it against problem test cases, and returns the verdict along with execution details.

User Profile Endpoint

- **Endpoint:** /profile
- **Method:** GET
- **Response:**
    - user:
        - id: string
        - username: string
        - name: string
        - solvedProblems: integer
        - attemptedProblems: integer
- **Functionality:** Fetches and returns the user's profile details including solved and attempted problems statistics.

Database Design

Users:

- _id: ObjectId
- username: String
- password: String
- name: String
- solvedProblems: Array
- attemptedProblems: Array

Problems:

- _id: ObjectId
- title: String
- description: String
- input: String
- output: String
- exampleTestCase: { input: String, output: String }
- testCases: Array

Submissions:

- _id: ObjectId
- userId: ObjectId
- problemId: ObjectId

Full Tech-Stack Flow

Frontend:

- **React.js**: Build components like ProblemList and ProblemDetails for displaying problems and their details.
- **React Router**: Enable navigation between pages such as home, problem details, and user profile.

Backend:

- **Mongoose**: Interface with MongoDB to handle data for Users, Problems, and Submissions.

Database:

- **MongoDB**: Store Users, Problems, and Submissions with structured collections.

Code Execution:

- **Docker**: Containerize the MERN stack for consistent environments across development, testing, and production.