Online Judge

An online judge refers to a website that has a library of coding problems, with test cases and expected outputs for each problem (preferably hidden from the user), where the user has the option of writing code for a particular problem and submitting it. Upon submission, the output of the user code will be compared to the expected output and a verdict will be displayed to the user.

Expected Features:

1. Users should be able to login by providing username/email and correct password.
2. Users should be able to submit their code through either file upload or writing in an in-built text editor.
3. Problem Library should display a list of problems.
4. Users should receive a verdict for each of their submissions.

High Level Design:

1. Database Schema:
    a. Problem Library JSON structure:
        i. Problem Title : string
        ii. Problem Statement : string
    b. Test Cases JSON structure:
        i. Input : string
        ii. Output : string
        iii. Problem link : link to problem JSON document
    c. Submissions JSON structure:

i. Verdict : string
ii. Submission Time : DateTime
iii. Problem link : link to problem JSON document

d. User Authentication JSON stricture:
i. User ID : string
ii. Password : string
iii. Email : string
iv. Username : string

2. Front End Structure:
a. Home Screen :
i. Login/SignUp screen
ii. Problem Library
b. Problem Screen:
i. Text editor section
ii. Language select section
iii. File upload section
iv. Submission log section.
c. Submissions Screen:
i. Past Submissions section (with verdicts)
ii. Links to past submission code.

4. Back End Structure:
a. Home Screen:
i. Using express.js setup an API GET endpoint that returns all problems to the Front End from the Database
b. Problem Screen:
i. Using express.js setup an API GET endpoint that returns all problem data to the Front End from the Database
ii. Also setup an API POST endpoint that retrieves test cases from the database, evaluates the

solution and returns a Verdict to the Front End

c. Submissions Screen:
    i. Setup an API GET endpoint that returns previous submissions from the database to the Front End.

3. Security Flaws and solutions:
    a. Flaw: Large number of users submitting simultaneously.

       Possible Solution: Implementing a submission queue, which might increase waiting time for users but will prevent server from crashing due to too many requests.

    b. Flaw: User submitting a solution that takes very long to finish running.

       Possible Solution: Implementing a Time Limit Exceeded verdict, which prevents user code from running longer than a fixed duration the length of which would depend on the problem setter.

    c. Flaw: User submitting a very large solution file, causing slowdowns.

       Possible Solution: Implementing a size limit for user submissions, which prevents users from submitting files larger than a specified memory size.

    d. Flaw: User submitting a solution that uses too much memory upon running.

       Possible Solution: Implementing a Memory Limit Exceeded verdict, which prevents user code from using more server resources than necessary.