HIGH LEVEL DESIGN

Project “Online Judge” by Rishikesh Dasari

Version : 01

Date : 09/07/2023

Author : D Rishikesh

Introduction

Overview:

The project – Online Judge is a basic online compiling and coding platform. It helps user to practice the Data Structure and Algorithm problems online on the platform itself. The user can submit their codes and evaluate them automatically. The current platform is designed to practice coding skills.

Objectives:

Online Judge comes with set of predefined problems along with specific inputs and expected output. It supports to code in various programming languages. The OJ provides the verdict for the submission of the code, indicates whether the written code has passed all the test cases/ failed some or encountered any errors. It enables the users to list them on the leader board to encourage them the competitive spirit and solve more problems to secure good ranking.

Scope

The scope of the OJ encompasses –

1. **User Management**: The user can be registered or logged to the web site through account registration mechanism with email IDs and logged through existed user account existed data information.

2. **Problem Management**: The platform OJ offers diverse set of problems categorized by difficulty level and programming language. It also provides users to create the problems to create, edit and delete problems along with the constraints, inputs and outputs.

3. **Code Verdict**: OJ enable the code can be written in multiple languages and interpreting/compiling them to the given test cases. It provides the verdict based on the submission of the code.

4. **Leader Board**: To engage and encourage the healthy competition, OJ maintain the leaderboards based on the performance of the different users.

5. **Administration and Maintenance**: OJ will include an administrative interface for system administration to manage user accounts, problems and overall system settings.

System Architecture

The System Architecture for Online Judge consists of several components that works together to provide desired functionalities to the users. They include –

1. **User Interface**:

It handles user interaction and present the OJ features through web-based interface. It includes web pages, forms and interactive elements allows users to register, log in, problem search, code submissions.

2. **Application Interface**:

This is the main component of the OJ which handles the user request, manage business logic and interactions. Django acts as server-side component and handles the user requests. Django views are responsible for UI functionalities. It includes modules such as user management, problem management, code submission and evaluation, scoring and ranking, and administrative functions.

3. **Database Design**:

The database used for building OJ is PostgreSQL/SQLite. The database contains different tables. The following indicates them –

1. Problems:
    a. Problem ID – Primary key (DB default generated)
    b. Problem Name – CharField
    c. Problem Description – CharField
    d. Code – CharField.
    e. Difficulty – CharField (Hard/Medium/Easy)
2. Solution:
    a. Problem ID – Foreign key
    b. Verdict – CharField
    c. Submission – Auto Date Time Field
3. Test Cases:
    a. INPUT – CharField
    b. Output – CharField
    c. Problem ID – Foreign Key

4. Leaderboard: (Optional)

    a. User ID – Foreign Key

    b. User Name – CharField

    c. Problem Count – INTField

5. User Credentials:

    a. User ID – Primary Key(System Auto Saved)

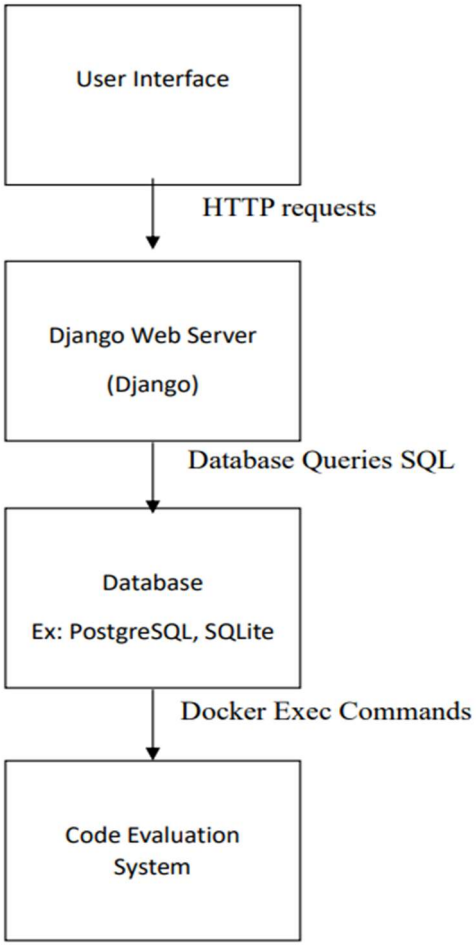
    b. User Name/email ID – CharField

    c. Password – CharField

4. **Execution Evaluation**:

- Docker: Utilizes Docker containers to create an isolated environment for code evaluation. A Docker container with the required compiler, such as GCC, is set up to execute the submitted code.

System Diagram:



User Flow

- User Authentication:
    a. If user registered UI requests a GET request to fetch problem details from the User table via the Django view.
    b. If new user UI send a POST request to create a new account to Django view.

- List Problems:
    a. UI requests a GET request to fetch problem details from the Problem table via the Django view.
    b. Django view retrieves the problem data and returns it to the UI, which displays the problem names.

- Show Individual Problem:
    a. UI requests a GET request to fetch the problem details from the Problem table via the Django view.
    b. Django view retrieves the problem data and returns it to the UI, which displays the problem description and a code submission box/form.

- Code Submission:
    a. UI includes a submit button below the code submission box.
    b. Upon submitting, the UI sends a POST request to the Django view.
    c. Django view:
        - Retrieves the test cases for the problem from the Test Cases table.
        - Executes the submission code in the Docker container and compares the output with the expected output.
        - Saves the verdict (e.g., correct, wrong answer) for the submission in the Solutions table.
        - Redirects to the leaderboard page.
- Leaderboard: (Optional)
    a. UI requests a GET request to fetch the number of verdicts (problem solved) from the Leaderboard table via the Django view.
    b. Django view retrieves the data and returns it to the UI, which displays the leaderboard.

Conclusion

This HLD outlines the architecture and major functionalities of the OJ built with Django. It includes a user-friendly UI, database for storing problem, solution, and test case data, and a code evaluation system using Docker and sandboxing techniques for secure execution of code submissions. Additional features and enhancements can be incorporated based on project requirements and further iterations of development