OJ Task Breakdown

High level

- Basic skeleton of Django webserver(Task 0) Writing your first Django app, part 1 | Django documentation | Django (djangoproject.com
- Desing your database(Task 1)
    - no. of tables (preferably 3:- problems, solutions & test cases)
        - Problems
            - Statement : CharField
            - Name : CharField
            - Code : CharField
            - Difficulty : CharField - Hard/Easy/Medium (Optional)
        - Solutions
            - Problem : Foreign Key
            - Verdict : CharField
            - Submitted At : Auto DateTime Field
        - Test Cases
            - Input : CharField
            - Output : CharField
            - Problem : Foreign Key
- Desing webserver(Task2)
    - No. of UI screens and their purpose. (Should not be more than 3, preferably 2)
    - Design UI for each screen(note: keep it simple)
    - Define and execute Django views (python methods to handle each of the functionalities included in the UI)
    - Your django app should have the following functionalities
        - List Problems
            - Template: A simple list consisting of names of each problem, linking it to the individual problem's page
            - View: GET request to fetch all problems from Problem table and return to UI
        - Show individual Problem
            - Template: Show Problem name and corresponding statement along with submission box to submit problem code in text format
            - View: GET Request to fetch the problem details from the problem table and return to UI
        - Code submission (this can be in the same screen as "show individual problem")
            - Template: A submit button below the Code submission box in the "Show individual problem" template

        - View: POST request to backend to handle execute following
            - Get the test cases(Input and output) for the problem from test cases table
            - Evaluate the submission code in your local compiler from Django view. (This will require you to call the system command to the compiler from your django backend)
            - compare the outputs from the compiler result to the output of the testcases in db
            - save the verdict for this submission in db
            - redirect to leaderboard page
    - Leaderboard(Result of each submission)
        - Template: A list showing the verdict of the last 10 submissions
        - View: GET request fetching the solution along with verdict for last 10 solutions from solution table
- Code evaluation system (Task 3)
    - docker
        - 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
        - Evaluate the code in the Docker container using the docker exec command(https://docs.docker.com/engine/reference/commandline/exec/)
    - Sandbox
        - Isolate -> https://github.com/ioi/isolate
        - Libsandbox -> https://github.com/openjudge/sandbox