Backend Development Assessment

Requirements

1. You can use any backend framework to solve this problem. (Solutions in **Django** or **Express.JS)** will be preferred.
2. Use OOPs concepts wherever possible
3. Follow general coding guidelines for syntax, code formatting etc.

Submission

1. Develop an application to solve the given task.
2. Include a **requirements.txt/package.json** file containing the list of all used dependencies.
3. Push the code along with the output file to a public git repository Github.
4. Submit the link of that public repository in the Google Form provided.

Evaluation Standards

We would be looking for -

1. How precise and robust is the code written for the API
2. How many internal test cases does the API pass
3. Is the code modular and readable

Task

- **To calculate the similarity among the candidates that attempted a survey**

Consider a survey with **20** questions. Each question has 10 options named as 1, 2, 3 .. 10.

Consider **10** candidates that filled the survey. A candidate can select any one of the given options for a question, or they can leave it blank.

Your task is to determine the similarity percentage for the candidates based on their responses. For example if two candidates have selected the same option for all the questions, their responses are 100% similar. If most of the options selected by the two candidates are the same, their responses are highly similar. If the options selected by the candidates are very different, their responses are highly dissimilar. If any one candidate has not submitted the response for one question, do not include that question while calculating the similarity in the responses of that candidate and other candidates.

This task can be divided into the following sub-tasks:

1. Create the relevant endpoints for achieving the following tasks
    a. List all the available surveys
    b. Create a survey with 20 questions
    c. Submit a response for a survey from a user
2. Then create an API endpoint to get the similarity among the responses of candidates. Include the following features in the API
    a. When called directly, it should return the similarity among different candidates.
    b. **Filters**: The API should be able to filter the results based on the people such that only the similarity between the responses of the passed candidate and everyone else is returned.
    c. **Pagination**: The API should give paginated results such that if there are 10 responses from 10 users, we should get the first 5 objects in the first page, then subsequently, 5 results per page.
    d. **Searching**: We should be able to search among the list of candidates such that the similarity between the responses of those users should come, whose name consists of the search text (irrespective of the sentence case).

Our motive is to check how you approach a problem, think of a solution and implement the same in a given time constraint. So focus on writing clean and precise code that solves the problem. All the best !!