THE UNSCIENTIFIC LOVE COMPATIBILITY CALCULATOR API - Coding Assesment (BE)

THE UNSCIENTIFIC LOVE COMPATIBILITY CALCULATOR API

In this assessment the objective is to develop an initial REST API with a single method. Please use NodeJs for this.

The method will accept some data and multiple calculations will be performed on that data. The result of each calculation will be multiplied by a weight, and the sum of the weighted scores will be returned as a result.

We haven't made up our mind about it yet, but we think that in the final version the calculators are

1. Classic love compatibility with a weight of 0.2
2. A bijective function using the OCEAN personality test scores of both persons with a weight of 0.5
3. A bijective function using the dimorphic differences of both persons with a weight of 0.3

Test scores are normalised before being fed into an aggregator function.

Example

Calculator 1 returned a score of 89 out of a maximum of 100
Calculator 2 returned a score of 100 out of a maximum score of 200
Calculator 3 returned a score of 0 out of a maximum of 6.

The normalised scores are 0.89 and 0.50.

The aggregator will perform this calculation --> `0.89 x 0.2 + 0.5 x 0.5 + 0.0 x 0.3 = 0.428`

Your API method must then contain in its result the number `0.428`.

The Task

Since we don't want candidates to spend too much time on this assessment, in this version we want you to implement the first calculator. You can find a link to a paper demo below. You may check your implementation using one of the many online calculators available as well. We provide two videos where the first calculator is implemented by hand.

Helpful information

1. You can find a paper calculation demo of the classic compatibility scoring here https://www.youtube.com/watch?v=oFsLVG7EAZ4 and here https://www.youtube.com/watch?v=2P47bHsmqFU
2. You do not need to develop a UI. Please don't. We will test your API using Postman and that will be good enough.
3. Please read the instructions very carefully. While you do not need to develop any calculator but the first (we have purposely kept you in the dark about the other calculators), you do need to develop the concept of the aggregator as a minimum. You may hardcode the weights for expediency.
4. The sum of all the weights the aggregator uses always sum up to 1.0. If there's just 1 expected input then that input has a weight of 1.00
5. If the classic compatibility calculator returns a score over 100, use 100.
6. It should take you about 30-45 minutes to complete the coding part of the task, but please take your time to read through what is required.
7. If you're going to copy the calculator code from the internet (and who hasn't done that) that's fine. We respect your time and we really don't want you to re-invent the wheel. In fact if you copy it from the internet and tell us where you copied it from you make everyone's life easier.
8. If you make any assumptions, please state these in an accompanying README file. And If you could take all the time you needed to, tell us what you would have added.
9. The assessment will be evaluated on your ability to read the instructions, meet the requirements while still thinking ahead but not too far ahead because we haven't made up our made about a few things yet, readability and design of your code and any additional information you give per the next point.
10. Lastly, you may put any thoughts you have about the assessment in the same README file. We would especially like to know what you think this test is about.

Please zip your work and send it to the recruiter.

Thanks