Dyno Therapeutics Coding Challenge

The purpose of this task is to assess your experience with a variety of software and cloud engineering tasks. The finished product will allow an end user to upload a file and then perform some lookups against it. You will have the opportunity to walk us through your implementation and talk about any tradeoffs and design decisions that you made along the way.

Your solution will allow a user to upload a gzipped file which contains data in FASTA format (https://en.wikipedia.org/wiki/FASTA_format). An example file has been attached. When a file has been uploaded, your solution automatically stores the data in a relational database of your choice, replacing any previously uploaded data. Please do not reload the file if it is identical to the last uploaded file. Also do not load the file if it exceeds 1mb in zipped size. You may allow the file to be uploaded however you see fit.

Your solution should include a REST API with the following 2 endpoints:

/exact_matches

This endpoint accepts json of the form { “sequence”: “some_sequence_here”} and returns all of the information about sequences in the database that exactly match the input sequence.

/hamming_matches

This endpoint accepts json of the form { “sequence”: “some_sequence_here”} and returns all of the information about sequences in the database within 1 hamming distance from the input sequence. (https://en.wikipedia.org/wiki/Hamming_distance)

To turn in the challenge, send us the url of the API, information on how to upload files, plus any code you have written for it (either zipped, or link us to a repo). Please use AWS or GCP to host your solution. Alternatively, a docker compose/terraform/ansible script or similar that would allow us to test it locally. We like to use Python, but you’re welcome to use a similar language if you’d prefer. Otherwise, you can complete this task however you wish.

Note: If you no longer have access to Amazon or Google’s free trial, **please reach out** and we will find a solution. The exercise should take only a tiny amount of computational resources.