Backend Coding Challenge

The role of a software engineer at Ginkgo requires integration of internal and external data services as well as data persistence infrastructure. This requires versatile development skills and knowledge of secure, scalable patterns for service-to-service communication and data modeling.

In our software, we generally work with DNA and Proteins. DNA is transcribed to RNA, which then can be translated to Protein. The objective of this exercise is to create a web application that can determine whether a particular DNA strand matches any part of the DNA sequences that could encode a protein in a well-known set.

This exercise involves implementing a fairly simple sequence alignment algorithm but requires the integration of multiple components and deployment so that we can see your great work. Your solution should be closer to a minimal viable product (something to build on) than a finished production in regards to feature completeness.

It's strongly suggested that you use the Biopython library rather than try and implement your own alignment search algorithm. Ginkgo uses Django and React (and we prefer you use them for this exercise), but if you feel more comfortable using another language or web framework, please do.

Application Overview:

- A web application serves a simple browser Javascript client that takes a DNA sequence as a form input.
- Upon submission of a DNA sequence, an asynchronous alignment is started to find a Protein that contains the submitted sequence.
- Proteins from the provided list of genomes[1] are searched until one is found containing the provided sequence. The list will be much larger in real life. Note that a single genome may contain multiple proteins.
- The browser client allows for additional submissions before a (the first) submission has completed.
- When a submission is complete, the web client is updated with the results, which include the name of the protein and where the sequence was found in the protein's sequence. Simple text-based results is all that is required.
- A user can close their browser, and upon subsequently revisiting the web application, their previous submissions and any active submissions are reloaded automatically.
- Users can see their previous searches and respective results.

Requirements:

- The only required technologies are Python and Javascript.

- A service is required for the logic; calling a command line interface via a shell invocation or invoking an external, third party web server are not desired solutions.
- Running all required processes for local development should be automated; one command. For example, if a webserver, task runner and database are required, then a single command should be able to launch all of these for the purposes of development.

We encourage questions, but please submit them in one email.

When ready for review, please submit:
1. A single Github repo containing your solution
2. The URL of the running application.

---

[1] NC_000852, NC_007346, NC_008724, NC_009899, NC_014637, NC_020104, NC_023423, NC_023640, NC_023719, NC_027867