Assignment Guidelines

Why are we giving you this assignment?

If you are reading this, we think you could be a good fit for our team. For the next step, we'd like to assess your coding, testing, and architectural design skills by asking you to create a web service. Completing this challenge should normally take between half to a full day to complete, depending on your background and experience. Please read the instructions in this document *carefully* and good luck!

Requirements

Technologies

- Code should preferably be written in Node/JavaScript/TypeScript and/or Python, as these are our primary languages.
- You must make use of Docker (https://www.docker.com/) and docker-compose (https://docs.docker.com/compose/) to package your system so it can run easily. We review assignments as they come in, while working on our core products, so we don't have much time to debug your service. Also, please avoid packaging Windows-based Docker containers.
- The entire service *must* be runnable with a single docker-compose up command.

Architecture

Problem Statement

We want to create a document search application. Here are the features expected to be developed by you.

1. Users should be able to add documents.
2. Users should be able to search for a word and it should return the best matching document.

Expectations

1. Create an API for managing documents. (add/list).
2. Create an API for returning the best 3 documents that match the queried keyword.

Assumptions

1. The application must support pdf documents. However, you are free to support other document types too.
2. The client does not have any specific requirement for the *best matching* document.
   It is up to you to architect this and provide justification for your design.
3. Assume that document once added will not be updated or deleted.
4. Since processing PDF files could be time-consuming, this service should have a long-running job architecture. Possible architectures would include queue-based workers or webhooks, but you are free to design it as you see fit.
5. You do *NOT* need to make any frontend UI for this service.
6. You do *NOT* need to deploy or host this service anywhere - it just needs to be runnable and testable on any normal computer - but we do encourage you to think about how you might host and scale your service and discuss it in your README.

Testing

- You should provide basic unit/integration tests for your service and they should be easily runnable.
- You are NOT required to implement all corner cases, but we do encourage you to think about them and discuss them in your README.

Documentation

You must provide a README file containing the following information:

- How to run and test your service
- What API endpoints are available and how to use them - *It should be easy to run and use the service*. Please provide example images, curl requests, etc.
- An explanation of your architecture, e.g. What components exist in the system? How are they connected?, What libraries/dependencies/tools did you choose and why?
- Any improvements you would make to the service or tests that go beyond the scope of the assignment - e.g. What would need to change to put your service into production? How could it be scaled up or down? How will it handle a high load of requests? What can you do to monitor and manage your services ?

Also, please use a git repository or similar to keep track of your progression/changes. We would like to see some of your progression as you complete this assignment. A private local repository should be fine, which you can create by calling git init in your folder and making commits to it. Then you can submit a zip file of the whole folder (which would include the .git folder in it).

Submission

When you have finished the assignment, please send it to engineering-assignment@cogent.co.jp either as an attached unencrypted ZIP file or as a link to the Github/Bitbucket repo if applicable. Make sure to state your name in the email and README.

How we review

We want to evaluate not just your code and architecture, but also your ability to think about future improvements and current shortcomings of your service. Overall, we look at the following things:

- Organization, readability, and consistent style of your code
- Design choices as well as discussion/consideration of those choices and future improvements.
- Test cases, and discussion/considerations of more complicated or rare potential test cases

It is okay if you don't have enough time to finish everything - some things will have to be set aside to your README discussion, but it is important that your service is fully runnable and testable, even if it is missing features or specific test cases.