Nautilus Labs: SWE Take-Home Test

Thank you for taking the time to work on the Nautilus Labs take-home assignment. This assignment will let you demonstrate an important part of being a software engineer: writing code.

Problem Intro

As a software engineer at Nautilus Labs, you must be able to work well with time-series data. The Nautilus Platform receives over 100 values every ten minutes from every ship that we are optimizing. Every feature of our platform needs to retrieve, transform, store, analyze, and respond to queries about that data.

In this take-home assignment, we would like you to build a fully functioning, mini-version, of Nautilus Platform. Specifically, we want you to build an API server that will load a CSV file of ship data, process it, store it, and then listen and respond to queries about that data.

In addition, during the onsite interview we will have questions about the code you submitted for this assignment, and also ask you to add additional features to it.

Problem Details

Your API server will need to read in a CSV file and provide a REST API to handle queries on that data. (It's not necessary that you read the data into a database. Storing the data in memory is fine.)

Attached is a CSV file containing 100 rows with 3 columns from a real ship:

- **timestamp** - unix epoch time (seconds since 00:00:00 January 1 1970).
- **speed** - ship speed given in miles per hour.
- **fuel** - fuel consumption given in gallons per minute.

You will need to do some cleanup of the data once it is loaded. Since the **speed** and **fuel** values in the CVS are coming from real sensors, there can often be errors. In this case, there are some blank values. If a value is blank, then you need to make a best guess as to what that value could be. An easy solution is to use the value at the previous timestamp. A better solution is to take an average of the previous and next value.

The API must handle the following queries:

- GET: /total_distance?start=*START_TIME*&end=*END_TIME*
    - Response is json of the form: {total_distance: *RESULT*}
    - *RESULT* is the total distance traveled in miles from the *START_TIME* to the *END_TIME*.

- GET: /total_fuel?start=*START_TIME*&end=*END_TIME*
    - Response is json of the form: {total_fuel: *RESULT*}
    - *RESULT* is the total amount of fuel consumed in gallons from the *START_TIME* to the *END_TIME*.
- GET: /efficiency?start=*START_TIME*&end=*END_TIME*
    - Response is json of the form: {efficiency: *RESULT*}
    - *RESULT* is the average miles per gallon from the *START_TIME* to the *END_TIME*.

Submission Details

Your code must be readable, well-structured, commented, and include tests. Also include a README file that describes how to build and run your API server, and also include links to 3rd party libraries that you are using in your code. Code must be buildable and runnable from the command line, and not depend on an IDE.

Code must be written in (in order of preference): Go, Python, Scala, Java, C#, Javascript, C++, or Ruby

Final submission can be sent as a tar, zip, dropbox (or similar), or as a link to your public GitHub repo.

Please don't hesitate to reach out if you have any questions.