Software Engineer (Node)

Take-home Task

This task is intended to take 1 hour. We are looking for an indication of your thought process outside of an interview context, not a comprehensive and in-depth solution. We'll discuss your responses in the next stage of the interview along with any points you haven't had time to cover. If you find yourself spending longer than one hour, please note down what you would do if you had more time and we will discuss it in the next interview – do not worry about finishing.

When you have completed your solution, please push the code to Github, and send the repository URL to bmenissier@powerx.ai. If the project is private, please also add `bastienmenis` as a project collaborator. Feel free to ask any questions or clarifications at any time.

We are looking to evaluate:

- Your approach to feature implementation
- Your ability to write idiomatic Typescript code
- Your understanding of software development best practices
- The way you choose development priorities

Problem statement

Our company is building a new data integration. Data will be sent to our API in a plaintext format. We need to store this data and be able to retrieve it on-demand later. The API will be written in TypeScript with Express.

You are tasked with implementing the storage and retrieval logic through two API endpoints:

- `POST /data` — this endpoint will receive data in a plaintext data format. Each line represents one reading, which consists of a timestamp, a metric name, and a metric value. For example:

  ```
  1649941817 Voltage 1.34 1649941818 Voltage 1.35 1649941817 Current 12.0 1649941818 Current 14.0
  ```

  The API should parse this data, store it to the database and return `{ "success": true }` to the client. If the data is malformed, the API should return `{ "success": false }` without storing anything in the database.

- `GET /data?from=2022-01-01&to=2022-01-03` — this endpoint should retrieve two query parameters `from` and `to`, which will be ISO standard dates or date-times. The API should retrieve all data within the given date range, formatted as JSON like so:

  ```
  [ { "time": "2022-04-14T13:10:17.000Z", "name": "Voltage", "value": 1.34 }, { "time": "2022-04-14T13:10:17.000Z", "name": "Current", "value": 14 } ]
  ```

- `GET /data?from=2022-04-12&to=2022-04-14` — this endpoint should retrieve two query parameters `from` and `to`, which will be ISO standard dates or date-times. The API should retrieve all data within the given date range, formatted as JSON like so:

```
[ { "time": "2022-04-14T13:10:17.000Z", "name": "Voltage",
"value": 1.34 }, { "time": "2022-04-14T13:10:17.000Z", "name":
"Current", "value": 14 } { "time": "2022-04-14T00:00:00.000Z",
"name": "Power", "value": 18.76 }, ]
```

- The `GET` endpoint should also return an average `Power` reading for each day within the query range.

Codebase

The current API code can be found here: https://github.com/powerxai/starter-interview-swe-api. You can click the "Use this template" button in the Github interface to create your own copy of the repository to work on.

A basic API scaffolding has already been provided, which you are free to extend as you see fit. Please install additional libraries if you think they are appropriate. For simplicity, data can be stored in-memory using the implementation in `database.ts`. Feel free to change this implementation to suit whatever data structure you would like to use.

Sample Requests

To test your implementation, you may like to use these sample requests:

▼ **Add Data**

```
curl --request POST \ --url http://localhost:3000/data \ --header
'Content-Type: text/plain' \ --data '1649941817 Voltage 1.34
1649941818 Voltage 1.35 1649941817 Current 12.0 1649941818 Current
14.0'
```

▼ Add Data (malformed)

```
curl --request POST \ --url http://localhost:3000/data \ --header
'Content-Type: text/plain' \ --data '1649941817 Voltage 1.34
1649941818 1.35 Voltage'
```

▶ Get Data

Additional Considerations

- How can we test the code to be confident in the implementation?
- How can we make sure this code is easy to maintain for future developers?
- Our API needs to be high-performance — how can we measure the performance of our API?
- How could we optimise this code if the API receives many more `POST` requests than `GET` requests? What about if the API receives many more `GET` requests than `POST` requests?
- Would any of this logic need to change to scale to millions of simultaneous connections?