Connex One - Full Stack Developer Technical Test

Summary:

This test is designed to evaluate both technical skills in a real-world scenario and how you'd work as a team member at Connex One. The task is to build a simple API, and frontend application for displaying the data returned by the API.

Specifications for both the API and frontend applications can be found on the following pages.

One of the most important (and challenging) parts of development is working as part of a team. If you have any questions about requirements or implementation feel free to contact Daniel Kelly at In Technology Group.

Timescale / Deliverables:

- You should spend at most 4 hours on this task. In this time some candidates will have completed the task in full; some will have made progress and have come away with learning points and/or questions.
- The API and frontend application can either be separate decoupled projects, or built as one monolithic application, depending on candidate preference.
- Application code can be written in either JavaScript or TypeScript, depending on candidate preference.
- Where possible under time constraints, application code should be covered by linter checks, include additional testing as appropriate (unit, integration, or snapshot), and be error-free.
- General good practices for the software-development lifecycle should be used.
- Source code for the applications should be hosted on a version control platform (GitHub, GitLab, Bitbucket, etc.).
- A running instance of the application does not have to be hosted anywhere, but the project should be sufficiently documented that a reviewer could execute the project from their own machine.
- Please send details for the repository to: daniel.kelly@intechnologygroup.com

API Specification:

HTTP Verb	Endpoint	Specification
GET	/time	Responses returned from the API should validate against the following JSON schema; `{ "properties": { "epoch": { "description": "The current server time, in epoch seconds, at time of processing the request.", "type": "number" } }, "required": ["epoch"], "type": "object" }`
GET	/metrics	Response should serve all available Prometheus-format metrics for the API, including default recommended metrics (see configuration value *collectDefaultMetrics*).

The API should return a 403 code for all API requests that do not include header *Authorization* with value *'mysecrettoken'*.

Required technologies / packages:

- nodejs - https://nodejs.org/en/
- express - https://www.npmjs.com/package/express
- express-prometheus-middleware -https://www.npmjs.com/package/express-prometheus-middleware

Suggested technologies / packages:

- express-generator - https://www.npmjs.com/package/express-generator

Frontend Specification:

The frontend application requires one page, split into two sections horizontally.

On the left-hand side of the application, display:

- The most recently-fetched value for server time (retrieved by hitting endpoint */time*), displayed in epoch seconds.
- The difference between current client machine time and the most recently-fetched value for server time in epoch seconds, formatted in stopwatch format (i.e. *HH:mm:ss*; a difference of 32 seconds would be *00:00:32*, a difference of 0 seconds would be *00:00:00*).
- The displayed difference should update once per second. Eg. An initial difference of *00:00:00*would change after one second to *00:00:01*.

On the right-hand side of the application, display:

- A HTML preformatted code block containing the most recently-fetched value of all Prometheus metrics (retrieved by hitting endpoint */metrics*).

On loading the application for the first time, begin an API request to endpoints */time*and */metrics*to load current data values. Every 30 seconds, the frontend application should make API requests to endpoints */time*and */metrics*to load the latest data.

While a network request is occurring, the relevant half of the screen should indicate this (eg. a div covering it entirely with text ‘Loading’, or a spinner of some kind below the content).

All API requests made by the frontend should include header *‘Authorization’* with value *‘mysecrettoken’*to ensure requests do not receive a 403 response.

Required technologies / packages:

- React - https://github.com/facebook/react

Suggested technologies / packages:

- create-react-app - https://github.com/facebook/create-react-app