Fullstack Take Home Assessment

This is a fairly open-ended coding challenge. Use your skills and experience to show us what you're capable of! Be as creative as you wish. Anything above and beyond the minimum requirements will work in your favor.

Scenario

You have to build a REST web service that exposes CRUD functionality to a database that stores carbon usage data for customers.
You are allowed to use any technology on the backend side (TypeScript or Python please) and any TypeScript-based framework/library in order to deliver a fully working web application.

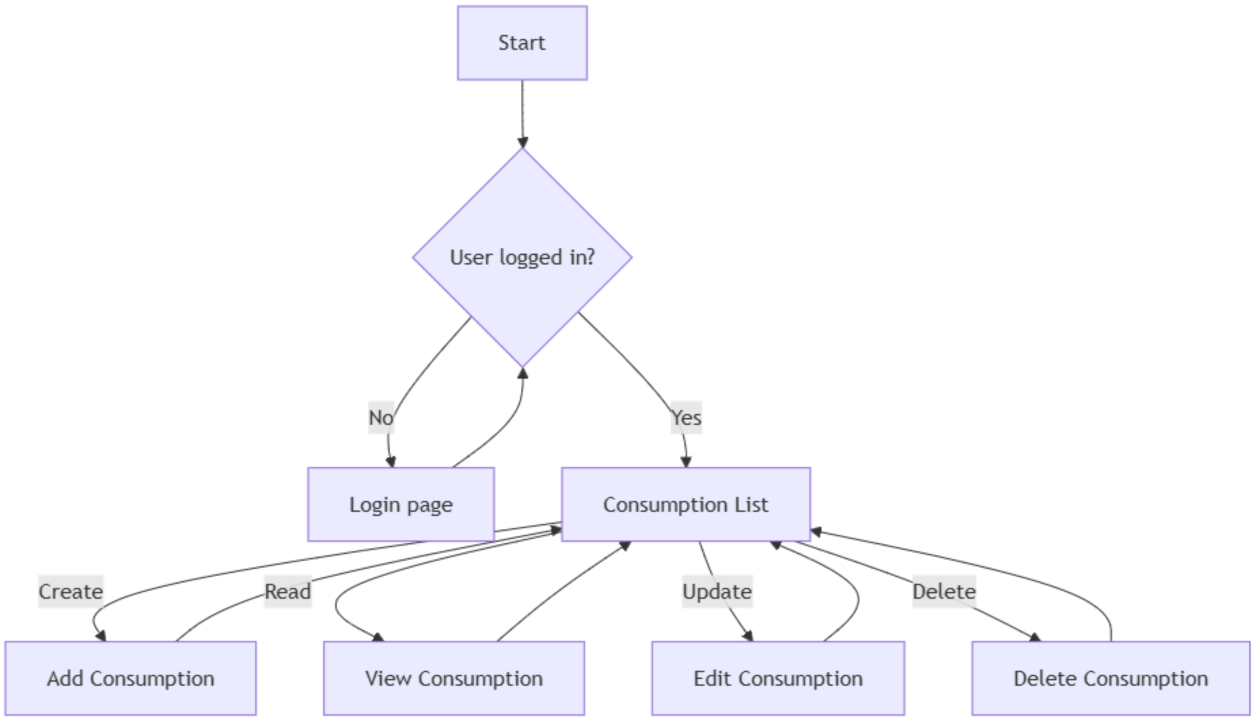
You may use open source libraries and components, but not no-code or low-code SaaS applications.

You will build:

- Small database to store carbon usage data for users - can be in-memory but a containerized image would be nice to show.
- Rest API using industry standards
- Single Page Application (SPA) consuming the above API in a modern framework such as React or Vue.

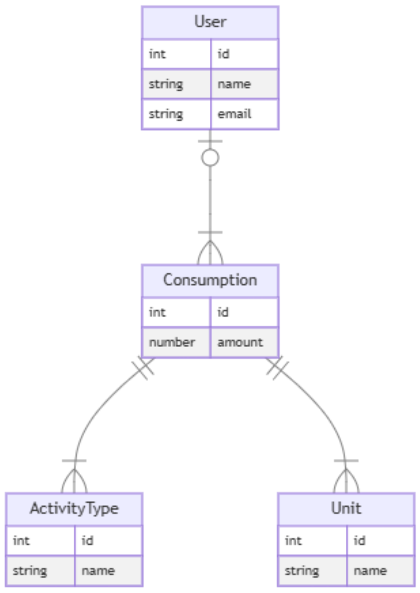
User flow

(very basic)



Database

The database should have, at least, this structure. Feel free to make your data structure more complex, but be prepared to explain it if it is not obvious.



Requirements

- Your files should be in a Github repository we can access.
- The backend could be made using a common web framework for TypeScript or Python.
- The frontend can be in any modern web framework, in TypeScript.
- Unit Tests should be available
- Your API should support all CRUD actions. Try to think about how a frontend might need to use this to allow users to submit, edit or delete their usage and retrieve it.
- The README in the root directory of your repository should provide instructions so that we can run and test it locally. We highly recommend you set it up with Docker to make both our lives easier.
- Please make sure your solution is documented from an implementation perspective (the how and why of some decisions, pain points and possible improvements)
- Please also include how long it took you to complete this task, and if you had any challenges that you had to overcome. You could also use some kind of project management / ticket tracking tool like github issues, trello, jira, etc and show us how you broke this work down into workable pieces and accomplished your goal.

Optional

- Your API is secured with token-based authentication (such as JWT).
- Modern UI choices
- Your API supports pagination, sorting and a filter by time range when retrieving.
- How else can you impress us? Just make something cool and show-off-worthy!