Welcome to the React Beer E-Commerce Challenge!

In this challenge, we aim to create a small part of a beer e-commerce web-app for mobile devices using JavaScript and React. The goal is to build two simplified versions of the following pages:

- A PLP (Product Listing Page) displaying a list of products.
- A PDP (Product Details Page) displaying detailed information for each product, and allowing the user to see price and inventory (i.e., number of items in stock) for different size-variants of each product.

The visual specification for these pages is found in the following link:
https://www.figma.com/file/7YLJQSm9fBWzqbs9UqzPnO/Web-developer-Challenge-2?node-id=405%3A367&t=aEeRJiFSK3eFij3X-0

In addition, you are asked to create a very simple backend server, with two API endpoints, that serves the products and size-variants information needed by the front-end. The data for these endpoints comes from two files included with this challenge:

- A file named `products.js` containing a list of all products and their details.
- A file named `stock-price.js` with inventory quantities and price information for the size-variants of the different products. Keep in mind that prices in this file are in cents.

Technical requirements for the frontend:

- The PLP should be served at the url `/products`.
- The PDP for a given product should be served at a url in the format: `/product/productId-productBrand`. For example, the product with ID 127 of the brand "Modelo Especial," should be served at the url: `/product/127-modelo-especial`.
- The PDP should check for updated stock and price information every 5 seconds.
- When styling the PLP, you can use any library you want.
  However, when styling the PDP, *do not use any library (like Tailwind, Material-UI, Bootstrap, etc.)*, *but you can use SAS.*
- All error messages, that you think should be shown to the user, should be reported by `Window.alert()`.
- The CTAs that engage parts of the application that are not implemented in the challenge (e.g., the "Add to cart" CTA) should display a `Window.alert()` with the relevant information to be submitted (if any).

- ***Most importantly, write the code in an elegant, simple, and modular way, organizing things similar to how you would do in a real application for a real client.***

Technical requirements for the backend:

- Create an API endpoint served by a `GET` request to the url `api/products` that returns the list of all products and their details.
- Create an API endpoint served by a `GET` request to the url `api/stock-price/[sku]` that returns the price and number of items in stock for a given product size-variant, identified by its SKU. For example, for the SKU 10041, the url should be: `api/stock-price/10041`.
- Make sure you *do not* modify the "`products.js`" and `stock-price.js` files (except for changing prices/stock values inside `stock-price.js` while you are testing the 5-seconds update requirement for the frontend. Note that to make such a stock/price change effective you should restart the server (i.e., the server *does not* need to detect changes to `stock-price.js`).

Please make sure that you do the following:

- ***Do not use*** any framework (like Next.js, etc.). You can, however, use any NPM libraries you wish.
- *Carefully follow all the specifications given above*, and the visual specifications given in the Figma link. Note, however, that you do not have to use the exact same fonts as in the Figma, and any close enough font will do.
- Separate the frontend and backend code into two completely separate project folders, each with its own `package.json` file.
- Include a README.md file in each of the frontend and backend project folders, with *clear and full instructions* on how to install and run that part of the challenge.
- Any instructions, error-messages, comments, etc. *should be written only in English*. Please do not use anywhere any other human language (like Spanish).
- Use the *latest version* of Node.js.
- Write the project in JavaScript, *not* in TypeScript.

Good luck!