Front-End Code Challenge

This test is designed to assess your skills as a front-end Javascript/CSS/HTML5 developer.

1. Mission

You will be building a basic shopping cart experience which leverages browser technology exclusively.

2. Requirements

2.1. Functionalities

You must create a shopping site front-end with the following functionality:

1. A product grid of 6 products that has the following features:
    1. Indicator when a product has low stock (< 3 items available)
    2. Indicator when a product is out of stock (0 items available)
    3. ‘‘Add to Cart” button which adds the product to the user’s shopping cart if it’s in stock
    4. Clicking on the product should open a basic product information page with a bigger image, long description and price information.
2. A shopping cart which can be opened on any page in a popup. Users should be able to change the quantity of the product in their cart and remove items. Quantity increases must be validated against available stock.
3. A checkout page which accepts the customer’s name, address and a payment option.
    1. Clicking the “pay” button should decrease the stock of each item and send the customer to a confirmation page.
    2. Think about any data validation, and what information we should collect on this page.
4. An order confirmation page which shows the user what they have bought, where it will be shipped and gives them the option to go back and start shopping again from the product grid page.

2.2. Project Scope

1. Documentation: description of the design, source code structure and an installation guide
2. React Toolkit: you can use any react boilerplate you want and give us why you chose it **(Preferably create-react-app)**
3. CSS framework: you may use any UI library you want to create a visually consistent experience **(Preferably Bootstrap 4 SASS)**

4. API Service: data for the app should be hard-coded in JSON files which are loaded by the front-end app, instead of a backend API. You are responsible for writing the sample data in a format compatible with your app.
5. A live running version of the app **(Preferably Heroku)**

3. Extra Credit

1. Use browser technologies to store the stock level of products even when the browser is closed and reopened.
2. Use browser technologies to store user information to pre-fill fields in the checkout page when the user returns.
3. Prototype out a payment method selector and payment information fields on the checkout page.
4. Write unit-tests for testing component views & logics code.
5. Use CI/CD tool for running automated tests & deployment **(Preferably Travis-ci)**