Front-End Coding Exercise

v. 1.1.0

Overview

GroceryCo wants to develop a kiosk checkout system for customers in their supermarket stores. The client wants a system that allows customers perform a “checkout” based on prices and promotions defined by GroceryCo. They plan on installing kiosks in their stores as a supplement to cashiers.

Implement a prototype checkout system which handles their pricing schemes.

The goal of this exercise is to demonstrate your attention to detail and thinking from a design perspective. Create a solution that would be worthy to put into production and can be extended by other members of the team.

Details

Write a React application (quick start by using the Create React App github project at https://github.com/facebookincubator/create-react-app). The application should be able to be launched in a browser and carry out the checkout process. The application should include static data (e.g. JSON) containing an unsorted list of items which represent products from a customer’s initial basket state scanned at the kiosk. The kiosk has a limited inventory of products available, so the number of items in the customer’s basket cannot exceed that number at any time.

Example products:

- Apple
- Orange
- Banana

The checkout process should be able to accept the same item multiple times and in any order.
For example, an Apple might be scanned, followed by two Oranges, and then an Apple again.

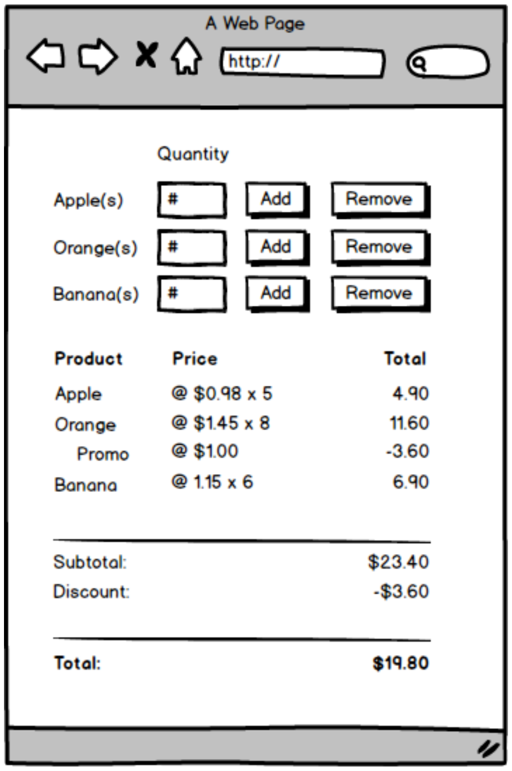
Prices for GroceryCo’s products are defined individually in their current price catalog.
For example, 'Apple' might cost $0.98, or ‘Bananas’ cost $1.15.

Products will occasionally be eligible for promotions for a limited time as defined by the marketing team.

Support the following promotion:

- **On Sale price.**
  Selected products are purchased at a discounted price which is less than regular price.
  For example, 'Orange': $1.00 Sale Price.

Display



Product	Price	Total
Apple	@ $0.98 x 5	4.90
Orange	@ $1.45 x 8	11.60
Promo	@ $1.00	-3.60
Banana	@ 1.15 x 6	6.90

Subtotal:	$23.40
Discount:	-$3.60
Total:	**$19.80**

The application should render into view a series of controls to add and remove products from the customer’s basket. Each control should contain:

- A read-only text field to show the name of the product
- An input field to set a numerical quantity of products being added to the basket
- An “Add” button to add products to the current basket state, respective of the control’s quantity and number of products left in the inventory
- A “Remove“ button to remove products from current basket state, respective of the control’s quantity

If the basket has reached the maximum available items in the kiosk’s inventory, the “Add” button should be disabled. If the customer has none of the products in their basket, the “Remove” button should be disabled. Your quantity input field should not allow the customer to enter in a number that exceeds the available items in the kiosk’s inventory.

As the basket is updated, render an itemized receipt containing the regular price for each item, discount or saving applied for a qualifying promotion, and the total price to be paid by the customer for the basket.

Advanced Requirement(s):
Support the following promotions:

- **Group promotional price based on the quantity purchased.**
  Products purchased which reach a specified quantity have a discounted price.
  For example, Buy 3 'Apple' for $2.00.
- **Additional product discount.**
  Products purchased which reach a specified quantity will discount an additional product of the same kind. For example, “buy one get one free” or “buy one, get one for 50% off”

Submission

Submit your solution via a GitHub repository. Include a README documentation with any necessary information (notes, limitations/design choices, or assumptions). Submissions are due one week or (7) days from receipt of this exercise.

Criteria

The application will be examined from these perspectives:

- **Architecture**
  - Does the code demonstrate good programming principles?
- **Appearance**
  - Does the site make good use of HTML and CSS to render a thoughtful layout
- **Readability**
  - Is the code readable to someone familiar with front-end development?
- **Maintenance**
  - How much developer effort is needed to extend and maintain the code?
- **Testing**
  - How well tested is the code?
- **Operational-ness**
  - Does the solution work correctly and efficiently?
- **Aesthetics**
  - Is the code elegant and nice to read?