

**COM S/SE 3190: Construction of User Interfaces**  
**Fall 2024**

**Assignment 03: Product Catalog and Cart**

[Total Points: 100]

**Assignment publishing: Wednesday October 23<sup>th</sup>, 2024**

**Assignment Due: Wednesday October 30<sup>th</sup>, 2024, 11:59PM**

**1. Overview**

In this assignment you will develop a small cart application. Your client is a small store that sells a variety of products. The client requires 3 views.

1. A browse view:
  - a. Contains a finite list of “addable” items.
  - b. User can pick multiple types of items and 1 or more instances of each item.
2. A cart view:
  - a. Displays the list, quantity and value of selected items.
  - b. Displays the total value + tax if necessary.
  - c. A checkout form to accept user credit card information and shipping address.
3. A confirmation view:
  - a. A page that shows the order summary.

We want all three pages to be implemented using the single-page design. I.e. you will only need one **index.html** page and ideally one other **script.js** page. You can also add your own **style.css** page. But custom styling is optional. We expect you to use bootstrap or tailwindcss. You can also use the bootstrap icons css file. We will be reviewing in class how to transition from one view to another. Specifically, how we implemented event listeners and state variables.

The application is a single-page design developed using REACT. If the project is developed out of this requirement the grading of the assignment will be zero.

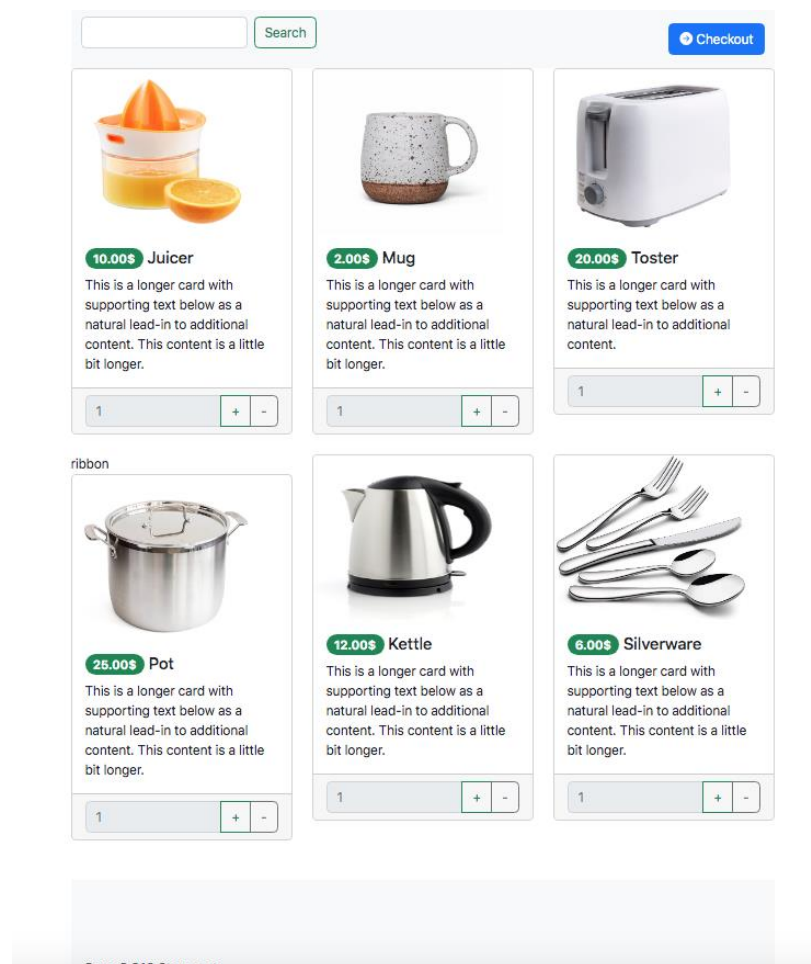
Feel free to add small touches to the webpage to enhance the view. You can use:

- Bootstrap icons, components etc.
- Feel free to play with font sizes, colors, especially with numeric values.
- You can customize the browse view, table and form components as you wish. But make sure you display all stated fields and values.
- Custom CSS to:
  - Enhance the look and feel with custom coloring, components, fonts etc.
- Just remember to maintain the underlying bootstrap/tailwindcss theme. Similarly, make sure your customization doesn't get in the way of the expected functionality outlined below.

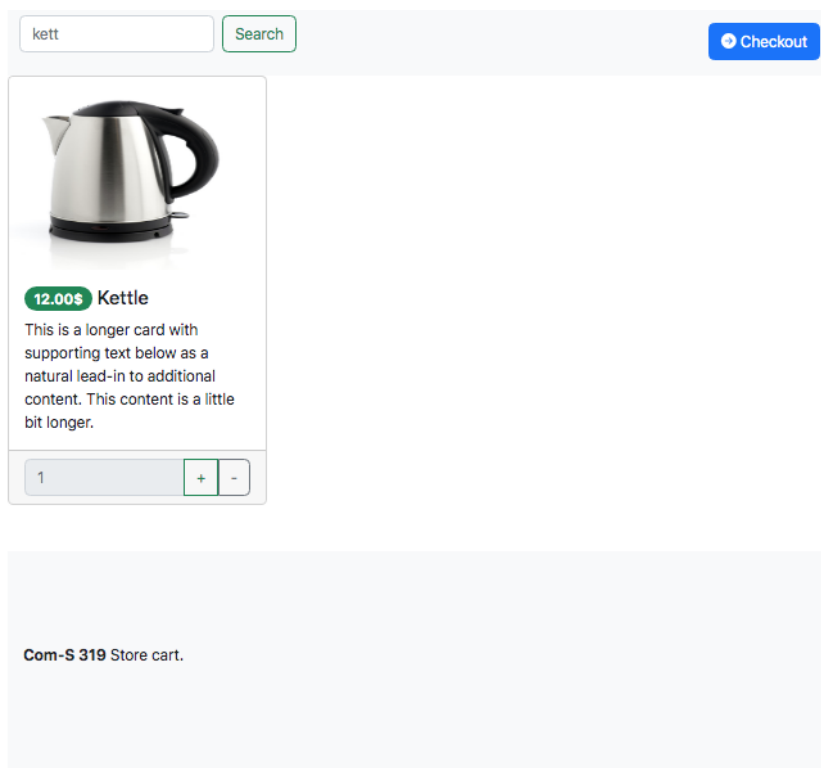
## 2. Design

### 2.1 Browse view (40 points)

- Your store must have a finite set of items. Anywhere between 6 –12 will suffice. In this example, the kitchenware store only sells 6 kinds of items.
- Feel free to sell other products.
- Notice how each item has “+” and “-” buttons directly below it. Each time you click the buttons, the quantity in the adjacent input field should increase and decrease accordingly. Assume there is no upper limit to your item count, but it shouldn’t be a negative number. So, watch out not to decrease below 0.
- Make sure you update a global “cart” variable every time item quantities change.





- Notice the search bar at the top. It should be functional.
  - Whenever a search pattern is inputted, all search cards containing the search substring as title should appear. At the same time, those whose titles don't contain the matching substring should fade away.
  - Feel free to add an additional “clear search” button.
  - In the example use case shown below, notice how typing the search substring “**kett**” is enough to filter the “**Kettle**”. Note that search strings are case **insensitive**.
  - **[Hint: Listen to the “input” event on the search input field.]**



## 2.2 Cart view (30 points)

- When clicking the “**checkout**” button, the user will be transported to the cart view. Remember we’re doing a single-page design, so there is no page navigation, just a <div> swap or the strategy that you consider more appropriate.

[Return](#)

Item	Quantity	Price
	Kettle	12.00\$
	Mug	4 x 2.00\$ = 8.00\$
Total		20.00\$ + 1.50\$ Tax = 21.50\$

### Payment information

Full Name

Email

Card

XXXX-XXXX-XXXX-XXXX

Address

1234 Main St

Address 2

Apartment, studio, or floor

City

State

Zip

Choose...

☐ Check me out

[Order](#)

- The cart view has a **return** button. When clicked, it should return us to the **browse** view.
  - Note that upon return, we should see all the items listed (no previous filters) with the correct quantity values in place.
- The **cart** view must show a list of selected items alongside their quantities and prices.
  - Don’t forget to add up the **total** value and tax (tax is optional).
  - Choose whatever tax formula you want.
- Finally, the page must have a **payment** form.
  - Full **name**, **email**, **card**, **address1**, **city**, **state**, **zip** are all required fields! Furthermore, they require validation!
    - Validate email and credit card fields. Also validate zip code. i.e. make sure it only accepts **5**-digit numeric strings. The other required fields need to have an arbitrary string value to be considered valid.
  - Address2 is an optional field.

- The checkbox serves no purpose, so feel free to remove it.
- When clicking “**order**”, we should navigate to the **confirmation** view.
- Remember to test your navigation robustness. i.e:
  - If you click “**return**”, you’ll go back to the previous browse view with the correct quantities set and no search string in the search field!
  - Click “**return**” followed by “checkout” and you should land back on the same cart view. (i.e. no changes in quantities or prices)

## **2.2 Confirmation view (20 points)**

- We leave the design of the confirmation view up to you.
- The only requirement is to display the following info:
  - Purchased items, preferably with pictures, and total purchase amount, optionally with price breakdown.
  - User information (Preferably redacting the first 12 digits from the user's credit card)
  - A button that will navigate you back to a fresh browse view. Don't forget to reset your “cart” object!
- Tips on designing the confirmation page:
  - Stick with the underlying bootstrap / tailwind theme.
  - Season the view with whatever custom style you think will enhance the look!

## **3. What to Submit:**

The task for grading can be heavily complex because it requires reconstructing the whole React application. To help in this task, you are going to make a video explaining the application from a technical and procedural points of view. The video must not last more than 3 minutes. The video will show your screen and the voices of the students explaining the different aspects, technical and procedural.

- Introduction slide: Start your video with a PowerPoint slide that introduces your team.
- The technical part will explain the application already running and will show an exploration of react directories and files, the content of the files including the images and text, html, css, json, modules, and JavaScript files.
- The procedural part will explain the way a final user or client should use to explore the products, searching, and how pass from one view to another by inputting data or clicking buttons. Any event that the application displays through data entry, the student must explain it.
- Format for the video : MP4.
- Include these files :  
Before creating a ZIP file, you will enter the folder application and remove the folder **modules**. That folder contains the Javascript libraries and other software (~300MB).

The remaining files are :

package.json  
package-lock.json  
README.md

public <-folder

src <- folder. This folder contains JavaScript files, image folder, text, etc

- Only these files and folders will be included in the ZIP file with name

**“TeamNumber\_Assignment03.zip”**

You will include the video with format MP4:

- Brief video of the Web site functioning with a duration of no more than 3 minutes.

#### **4. The Rubric to grade this assignment is as follows:**

**This Assignment has a total value of 100 points.**

**Late submission will deduct 5 points per day late.**

##### **Browse view 40 points :**

- Has a Browse view with minimum 6 items (image and text) ? \_\_\_\_\_ / 18
- Has a JSON file with reference to images and text ? \_\_\_\_\_ / 10
- Has a Search functional input ? \_\_\_\_\_ / 4
- Has “+” and “-“ functional buttons ? \_\_\_\_\_ / 4
- Has a Checkout functional button ? \_\_\_\_\_ / 4

##### **Cart view 30 points:**

- Has a list of selected items alongside their quantities and prices? \_\_\_\_\_ / 18
- Has a Payment Information section ? \_\_\_\_\_ / 4
- Has a Return functional button ? \_\_\_\_\_ / 4
- Has an Order functional button? \_\_\_\_\_ / 4

##### **Confirmation view 20 points :**

- Has a list of selected items alongside their quantities and prices? \_\_\_\_\_ / 12
- Has the User information? \_\_\_\_\_ / 4
- Has a button that will navigate you back to a fresh browse view? \_\_\_\_\_ / 4

##### **Video 10 points :**

- Has a brief video of the Web site functioning with a duration of no more than 3 minutes? \_\_\_\_\_ / 10

The description of this assignment includes requirements described above mentioning deduction points, such as : the application must be developed in React, not including modules, etc.