**Final Project: Paradise Nursery Shopping Application**



**Estimated time needed:** 110 minutes

**Introduction**

In this final project, you will create a shopping cart application for an online plant shop which offers a variety of house plants.

The *Paradise Nursery* shopping cart features will include:

* A Landing page with a button linking to the product listing page
* A navigation bar with links to the landing, product listing, and shopping cart pages
* A card for each plant that showcases the different plants along with their images, name, description, cost and an **Add to cart** button.
* A minimum of two sections describing the plants in that section. For example, "Aromatic Plants" and "Medicinal Plants".
* A cart page which displays the products in the cart.
* The cart should have a card for each type of plant in the cart. Each card should have the thumbnail, the unit cost, the cost for all of the plants of that type and buttons to increase and decrease the quantity along with **Delete** button.
* A **Continue Shopping** and **Checkout** buttons

You will implement the knowledge and skills you gained from working on the practice project to handle dynamic functionalities, like the show cart quantity in the icon on the navbar and updating the cost of all of the items in the cart when the user updates the number of items.

**Project Submission**

To submit your application for grading, you will need to deploy it. You will need to provide the URL as part of the peer review project. You can use GitHub Pages to host. You can find the instructions from the *Setting Up the GitHub Environment* lab earlier in this module.

You may choose to host it elsewhere and provide that link instead. Just make sure your peers can access the application easily using the URL you provide.

When you deploy, be sure to make a note of the application’s URL.

**Learning objectives**

After completing this lab, you will be able to:

* React Components: Create functional React components using component composition and nesting.
* State Management with Hooks: Implement React Hooks, specifically the useState and useEffect hooks. You will manage component-level state using hooks to control the visibility of elements.
* Redux Integration: Integrate Redux within an application using Redux concepts like actions, reducers, and the store.
* Rendering Dynamic Data: Dynamically render data fetched from an array of objects into the UI. You will map over arrays to generate lists of components.
* Handling Events and Conditional Rendering: Handle user events such as button selection and trigger corresponding actions.

**Prerequisites**

* Basic knowledge GitHub and your own GitHub account
* Understanding of React function components, props, hooks and React Redux Toolkit
* Web browser with a console (such as Chrome DevTools or Firefox Console)

**Task 1: ProductList component Layout**

The product page will allow your users to shop for the different plants you sell. Each plant will display on its own "card" with its related data stored in the plant object. You will store the plant objects in an array. Follow these steps for the array and plant objects.

1. Display the Plant Array

* Navigate to the ProductList.jsx component and you will see an array named plantsArray with the plants details.
* Each plant object contains the categories, properties name, image URL, description, and cost.

1. Display Plant Details within div tag with class name **product-grid**.

* Utilize array methods to map over the plant array.

*Hint: use the map() method to iterate array.*

* Render each plant's details on the page, including name, image, description, and cost.

1. Display an **Add to Cart** button for each plant.

Display plants and add to cart button solution

* Include above code within class name **product-grid**.

1. Create one variable named addedToCart for state management using the **useState** hook to track which products are added to the cart.

Sample solution for useState hook

1. Add to Cart Functionality

* Create the handleAddToCart function to implement the functionality for adding a plant to the cart when the user selects the **Add to Cart** button. This function should take one parameter that contains the information of the selected plant. This information should then be dispatched to the addItem inside the function component CartSlice.
* Additionally, reflect the product has been added to the cart. Update the setAddedToCart state to by setting the product name as a key and its value to true.

Sample solution for add to cart

*Note: Make sure that you import the addItem reducer from CartSlice.jsx*

1. The handleAddToCart() function will carry the details of that plant which user want to add in the cart. And the plant details to the cart at a global level using CartSlice.jsx.
2. Make sure that you save these changes by pushing your code to your GitHub repository.