Coding Challenge: Implement State Management for a Shopping Cart in Angular

Problem Statement:

You are developing a simple **Shopping Cart** application using Angular. The goal is to manage the state of the shopping cart efficiently using **RxJS and NgRx**. Users should be able to **view products, add them to the cart, remove them, and update their quantity**, while ensuring the application's state is properly managed.

Requirements:

1. Displaying Products:

- Fetch a list of products from a backend API (e.g., `/api/products`).
- Each product should have a **name, price, and stock quantity**.
- Display the products in a grid or list format.

2. Adding Products to Cart:

- Each product should have an "Add to Cart" button.
- Clicking this button should update the state using **RxJS (BehaviorSubject) or NgRx (Actions & Reducers)**.
- If a product is already in the cart, increase its quantity instead of adding a duplicate entry.

3. Viewing and Managing the Cart:

- Display the list of items added to the cart.
- Allow users to **increase or decrease the quantity** of each item.
- Provide a "Remove" button for each item to delete it from the cart.
- Implement logic to prevent adding more than the available stock.

4. Checkout Functionality:

- Implement a "Checkout" button that submits the cart items.
- After successful checkout, **reset the cart state** and display a confirmation message.

5. Error Handling:

- If the API request fails, display an appropriate error message.
- Prevent adding out-of-stock items.

Key Angular Concepts to Use:

✅ **HttpClientModule** for making API requests.
✅ **RxJS (BehaviorSubject) or NgRx (Actions, Reducers, Selectors, and Effects)** for managing state.
✅ **Observables and async pipes** to handle asynchronous data.
✅ **Angular Forms (Reactive or Template-driven)** for modifying cart quantities.
✅ **Angular Components** to structure the application.

Challenge:

🚀 **Enhance the user interface** by styling the product grid and cart using CSS.
🚀 Implement **a "Save for Later" feature**, where users can move items out of the cart but not lose them.
🚀 **Optimize performance** by using **selectors in NgRx** or memoization techniques in RxJS.

Example API Response for Products:

```
[
  { "id": 1, "name": "Laptop", "price": 1200, "stock": 5 },
  { "id": 2, "name": "Headphones", "price": 100, "stock": 10 },
  { "id": 3, "name": "Smartphone", "price": 800, "stock": 3 }
]
```

Expected Functionalities:

✅ Users can fetch and view available products.
✅ Users can **add/remove/update** items in the cart dynamically.
✅ Users cannot **add more than the available stock** of a product.
✅ The cart updates **reactively** using state management.
✅ Clicking "Checkout" **resets the cart** and shows a confirmation message.