Application Overview

We'll implement:

- **A product list**: Users can add products to the cart.
- **A cart display**: Shows items in the cart and updates dynamically.

---

2. Generate Components and a Service

Generate the components:
bash
CopyEdit
```
ng generate component product-list
ng generate component cart
```

1.

Generate a service to manage the state:
bash
CopyEdit
```
ng generate service cart
```

2.

---

3. Set Up the Cart Service

The service will manage the cart state using **RxJS BehaviorSubject**.

`src/app/cart.service.ts`
typescript
CopyEdit
```
import { Injectable } from '@angular/core';
import { BehaviorSubject } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class CartService {
  private cartItems = new BehaviorSubject<any[]>([]); // Initial state
is an empty array
  cartItems$ = this.cartItems.asObservable(); // Expose the observable
for subscriptions

  constructor() {}

  // Add a product to the cart
  addToCart(product: any): void {
    const currentCart = this.cartItems.value; // Get the current state
    this.cartItems.next([...currentCart, product]); // Update the
state
  }

  // Remove a product from the cart
  removeFromCart(product: any): void {
    const currentCart = this.cartItems.value;
    const updatedCart = currentCart.filter(item => item.id !==
product.id);
    this.cartItems.next(updatedCart);
  }

  // Clear the cart
  clearCart(): void {
    this.cartItems.next([]); // Reset state to an empty array
  }
}
```

---

4. Create the Product List Component

This component displays a list of products and allows adding them to the cart.

`src/app/product-list/product-list.component.ts`

typescript
CopyEdit

```
import { Component } from '@angular/core';
import { CartService } from '../cart.service';
```

```
@Component({
  selector: 'app-product-list',
  templateUrl: './product-list.component.html',
  styleUrls: ['./product-list.component.css']
})
export class ProductListComponent {
  products = [
    { id: 1, name: 'Product 1', price: 100 },
    { id: 2, name: 'Product 2', price: 200 },
    { id: 3, name: 'Product 3', price: 300 }
  ];

  constructor(private cartService: CartService) {}

  addToCart(product: any): void {
    this.cartService.addToCart(product);
  }
}
```

`src/app/product-list/product-list.component.html`

html
CopyEdit

```
<h2>Product List</h2>
<ul>
  <li *ngFor="let product of products">
    {{ product.name }} - ${{ product.price }}
    <button (click)="addToCart(product)">Add to Cart</button>
  </li>
</ul>
```

---

5. Create the Cart Component

This component displays the cart's contents and allows users to remove items or clear the cart.

`src/app/cart/cart.component.ts`

typescript
CopyEdit

```
import { Component, OnInit } from '@angular/core';
import { CartService } from '../cart.service';

@Component({
  selector: 'app-cart',
  templateUrl: './cart.component.html',
  styleUrls: ['./cart.component.css']
})
export class CartComponent implements OnInit {
  cartItems: any[] = [];

  constructor(private cartService: CartService) {}

  ngOnInit(): void {
    // Subscribe to the cart state
    this.cartService.cartItems$.subscribe(items => {
      this.cartItems = items;
    });
  }

  removeFromCart(product: any): void {
    this.cartService.removeFromCart(product);
  }

  clearCart(): void {
    this.cartService.clearCart();
  }
}
```

`src/app/cart/cart.component.html`

html
CopyEdit

```
<h2>Shopping Cart</h2>
<ul>
  <li *ngFor="let item of cartItems">
    {{ item.name }} - ${{ item.price }}
    <button (click)="removeFromCart(item)">Remove</button>
  </li>
```

```
</ul>
<p *ngIf="cartItems.length === 0">The cart is empty!</p>
<button *ngIf="cartItems.length > 0" (click)="clearCart()">Clear
Cart</button>
```

---

6. Update the Root Template

Include both components in the root template to display the product list and the cart.

`src/app/app.component.html`

html
CopyEdit

```
<div>
  <app-product-list></app-product-list>
  <app-cart></app-cart>
</div>
```

---

7. Run the Application

Start the Angular application:

bash
CopyEdit

```
ng serve
```

Visit `http://localhost:4200` to interact with the shopping cart.

Key Concepts Demonstrated

1. **BehaviorSubject**:
   - Manages the state of the cart and allows components to react to changes.
   - Initial state is set as an empty array (`BehaviorSubject<any[]>([])`).
2. **State Sharing**:
   - Both `ProductListComponent` and `CartComponent` share the cart state through the `CartService`.
3. **Reactive Updates**:

- Components subscribe to the `cartItems$` observable to automatically update their UI when the cart changes.

4. **Separation of Concerns**:
    - State logic is encapsulated in the service, while the components focus on UI.