```
import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { DataService } from '../../../core/data.service';

@Component({
  selector: 'app-add-product',
  templateUrl: './add-product.component.html',
  styleUrls: ['./add-product.component.css']
})
export class AddProductComponent {

  constructor(private dataService: DataService, private router: Router) { }

  // This function is called when the form is submitted
  onAdd(form: any): void {
    // Check if the form is valid before proceeding
    if (form.valid) {
      // Create a new product object from the form values, adding a unique ID
      const newProduct = {
        ...form.value,
        id: Date.now()  // Generate a unique ID using the current timestamp
      };

      // Call the service to add the product and handle the response
      this.dataService.addProduct(newProduct).subscribe({
        next: () => {
          alert('Product added successfully!');  // Notify the user of success
          this.router.navigate(['/inventory']);  // Navigate to the inventory page
        },
        error: (err) => {
          console.error('Error adding product:', err);  // Handle errors if any
          alert('There was an error adding the product.');
        }
      });
    } else {
      // If the form is invalid, alert the user to fill in all required fields
      alert('Please fill in all required fields.');
    }
  }
}


<h2>Add New Product</h2>
<form (ngSubmit)="onSubmit(productForm)" #productForm="ngForm">
  <div>
    <label for="productName">Product Name:</label>
    <input
      type="text"
      id="productName"
      name="productName"
      ngModel
      required
      minlength="3"
      #productName="ngModel"
    />
    <div *ngIf="productName.invalid && productName.touched">
      <small *ngIf="productName.errors?.['required']">Product name is required.</small>
      <small *ngIf="productName.errors?.['minlength']">Name must be at least 3 characters long.</small>
    </div>
  </div>

  <div>
    <label for="productDescription">Description:</label>
    <input
      type="text"
      id="productDescription"
      name="productDescription"
      ngModel
      required
      #productDescription="ngModel"
    />
    <div *ngIf="productDescription.invalid && productDescription.touched">
      <small *ngIf="productDescription.errors?.['required']">Description is required.</small>
    </div>
  </div>

  <div>
    <label for="manufacturer">Manufacturer:</label>
    <input
      type="text"
      id="manufacturer"
      name="manufacturer"
      ngModel
      required
      #manufacturer="ngModel"
    />
    <div *ngIf="manufacturer.invalid && manufacturer.touched">
      <small *ngIf="manufacturer.errors?.['required']">Manufacturer is required.</small>
    </div>
  </div>

  <div>
    <label for="productPrice">Price:</label>
    <input
      type="number"
      id="productPrice"
      name="productPrice"
      ngModel
      required
      min="0"
      #productPrice="ngModel"
    />
    <div *ngIf="productPrice.invalid && productPrice.touched">
      <small *ngIf="productPrice.errors?.['required']">Price is required.</small>
      <small *ngIf="productPrice.errors?.['min']">Price cannot be negative.</small>
    </div>
  </div>

  <div>
    <label for="productQuantity">Quantity:</label>
    <input
      type="number"
      id="productQuantity"
      name="productQuantity"
      ngModel
      required
      min="1"
      #productQuantity="ngModel"
    />
    <div *ngIf="productQuantity.invalid && productQuantity.touched">
      <small *ngIf="productQuantity.errors?.['required']">Quantity is required.</small>
      <small *ngIf="productQuantity.errors?.['min']">Quantity must be at least 1.</small>
    </div>
  </div>

  <button type="submit" [disabled]="productForm.invalid">Add Product</button>
</form>
```

Product detail

```
<h2>Item Information</h2>
<p><strong>Product Name:</strong> {{ item?.name }}</p>
<p><strong>Product Description:</strong> {{ item?.description }}</p>
<p><strong>Made By:</strong> {{ item?.manufacturer }}</p>
<p><strong>Price (USD):</strong> {{ item?.price }}</p>
<p><strong>Available Stock:</strong> {{ item?.quantity }}</p>
```

```
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { By } from '@angular/platform-browser';
import { ProductDetailComponent } from './product-detail.component';
import { DataService } from '../../../core/data.service';
import { HttpClientTestingModule } from '@angular/common/http/testing';
import { ActivatedRoute } from '@angular/router';
import { of } from 'rxjs';

describe('ProductDetailComponent', () => {
  let fixture: ComponentFixture<ProductDetailComponent>;
  let component: ProductDetailComponent;

  // Mocking ActivatedRoute to simulate route params
  const mockActivatedRoute = {
    snapshot: { paramMap: { get: jest.fn().mockReturnValue('1') } }
  };

  // Mocking the DataService to return a predefined product
  const mockDataService = {
    getProducts: jest.fn().mockReturnValue(
      of([
        {
          id: 1,
          name: 'Product 1',
          description: 'Sample description',
          manufacturer: 'Manufacturer 1',
          price: 100,
          quantity: 10
        }
      ])
    )
  };

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ProductDetailComponent],
      imports: [HttpClientTestingModule],
      providers: [
        { provide: ActivatedRoute, useValue: mockActivatedRoute },
        { provide: DataService, useValue: mockDataService }
      ]
    }).compileComponents();

    fixture = TestBed.createComponent(ProductDetailComponent);
    component = fixture.componentInstance;
    fixture.detectChanges(); // Trigger initial change detection
  });

  describe('component rendering and behavior', () => {
    it('should create the ProductDetailComponent instance', () => {
      expect(component).toBeTruthy();
    });

    it('should display the "Product Details" heading', () => {
      const headingElement = fixture.debugElement.query(By.css('h2'));
      expect(headingElement).toBeTruthy();
      expect(headingElement.nativeElement.textContent).toBe('Item        Information');        //
Updated to match the revised version
    });

    it('should render the product name', () => {
      const nameElement = fixture.debugElement.query(By.css('p:nth-child(2)'));
      expect(nameElement).toBeTruthy();
      expect(nameElement.nativeElement.textContent).toContain('Product 1');
    });

    it('should render the product description', () => {
      const descriptionElement = fixture.debugElement.query(By.css('p:nth-child(3)'));
      expect(descriptionElement).toBeTruthy();
      expect(descriptionElement.nativeElement.textContent).toContain('Sample description');
    });

    it('should display the manufacturer name', () => {
      const manufacturerElement = fixture.debugElement.query(By.css('p:nth-child(4)'));
      expect(manufacturerElement).toBeTruthy();
      expect(manufacturerElement.nativeElement.textContent).toContain('Manufacturer 1');
    });

    it('should show the product price', () => {
      const priceElement = fixture.debugElement.query(By.css('p:nth-child(5)'));
      expect(priceElement).toBeTruthy();
      expect(priceElement.nativeElement.textContent).toContain('100');
    });

    it('should display the available quantity of the product', () => {
      const quantityElement = fixture.debugElement.query(By.css('p:nth-child(6)'));
      expect(quantityElement).toBeTruthy();
      expect(quantityElement.nativeElement.textContent).toContain('10');
    });
  });
});
```

Product lis

```
<h2>Product Catalog</h2>
<app-search-filter (search)="onFilter($event)"></app-search-filter>
<ul>
  <li *ngFor="let item of filteredItems">
    {{ item.name }}
    <button (click)="viewItemDetails(item.id)">View</button>
    <button (click)="editItem(item.id)">Edit</button>
    <button (click)="removeItem(item.id)">Delete</button>
  </li>
</ul>
<button routerLink="/inventory/add-item">Add Item</button>
```

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { DataService } from '../../../core/data.service';
import { AuthService } from '../../../core/auth.service';

@Component({
  selector: 'app-item-list',
  templateUrl: './product-list.component.html',
  styleUrls: ['./product-list.component.css']
})
export class ItemListComponent implements OnInit {
  items: any[] = [];
  filteredItems: any[] = [];

  constructor(
    private dataService: DataService,
    private authService: AuthService,
    private router: Router
  ) { }

  ngOnInit(): void {
    this.dataService.getProducts().subscribe((data) => {
      this.items = data;
      this.filteredItems = data;
    });
  }

  onFilter(query: string): void {
    this.filteredItems = this.items.filter(item =>
      item.name.toLowerCase().includes(query.toLowerCase())
    );
  }

  viewItemDetails(id: number): void {
    this.router.navigate([`/inventory/item-detail/${id}`]);
  }

  editItem(id: number): void {
    this.router.navigate([`/inventory/edit-item/${id}`]);
  }

  removeItem(id: number): void {
    // Ensure the user is logged in before proceeding with deletion
    if (!this.authService.isLoggedIn()) {
      alert('You need to be logged in to delete an item!');
      this.router.navigate(['/auth/sign-in']); // Redirect to login page
      return;
    }

    // Proceed with the deletion process if authenticated
    this.dataService.deleteProduct(id).subscribe(() => {
      this.items = this.items.filter(item => item.id !== id);
      this.filteredItems = this.filteredItems.filter(item => item.id !== id);
    });
  }
}
```

Routing module

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { AboutPageComponent } from './features/about/about.component';

const appRoutes: Routes = [
  { path: '', redirectTo: '/catalog', pathMatch: 'full' },  // Default route redirect to catalog
  { path: 'about', component: AboutPageComponent },
  { path: 'auth', loadChildren: () => import('./features/auth/auth.module').then(m => m.AuthModule) },
  { path: 'catalog', loadChildren: () => import('./features/inventory/inventory.module').then(m => m.InventoryModule) },  // Renamed 'inventory' to 'catalog'
  { path: '**', redirectTo: '/catalog' }  // Catch-all route to redirect to catalog
];

@NgModule({
  imports: [RouterModule.forRoot(appRoutes)],  // Setup for the app's routing
  exports: [RouterModule]
})
export class AppRoutingModule { }
```