Abdul Wahid Roll no. 4914

Overview

This document outlines the development of several essential features for a furniture e-commerce website, focusing on:

- Filter search section
- Pagination
- Dynamic routing
- Product listing page with dynamic data
- Product detail page
- Working category filters

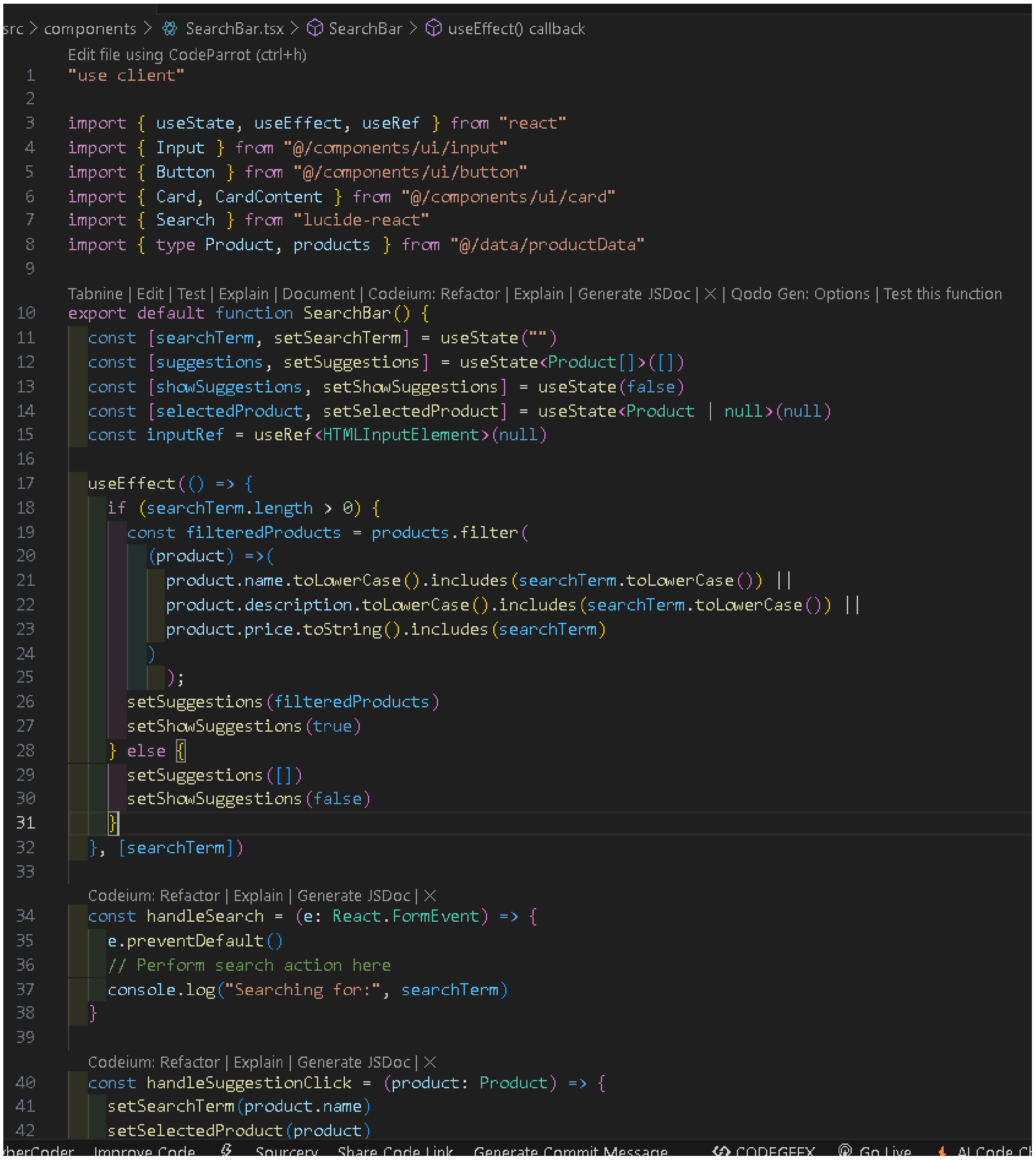
These components are designed to enhance the user experience by providing seamless navigation and efficient data handling.

Filter Search Section Component

The Filter Search Section allows users to refine their search results based on specific criteria, such as:

- Furniture type (e.g., Chairs, Sofas, Recliners)
- Price range
- Availability (e.g., in stock or out of stock)
- Material type (e.g., Leather, Fabric, Wood)

Abdul Wahid Roll no. 4914

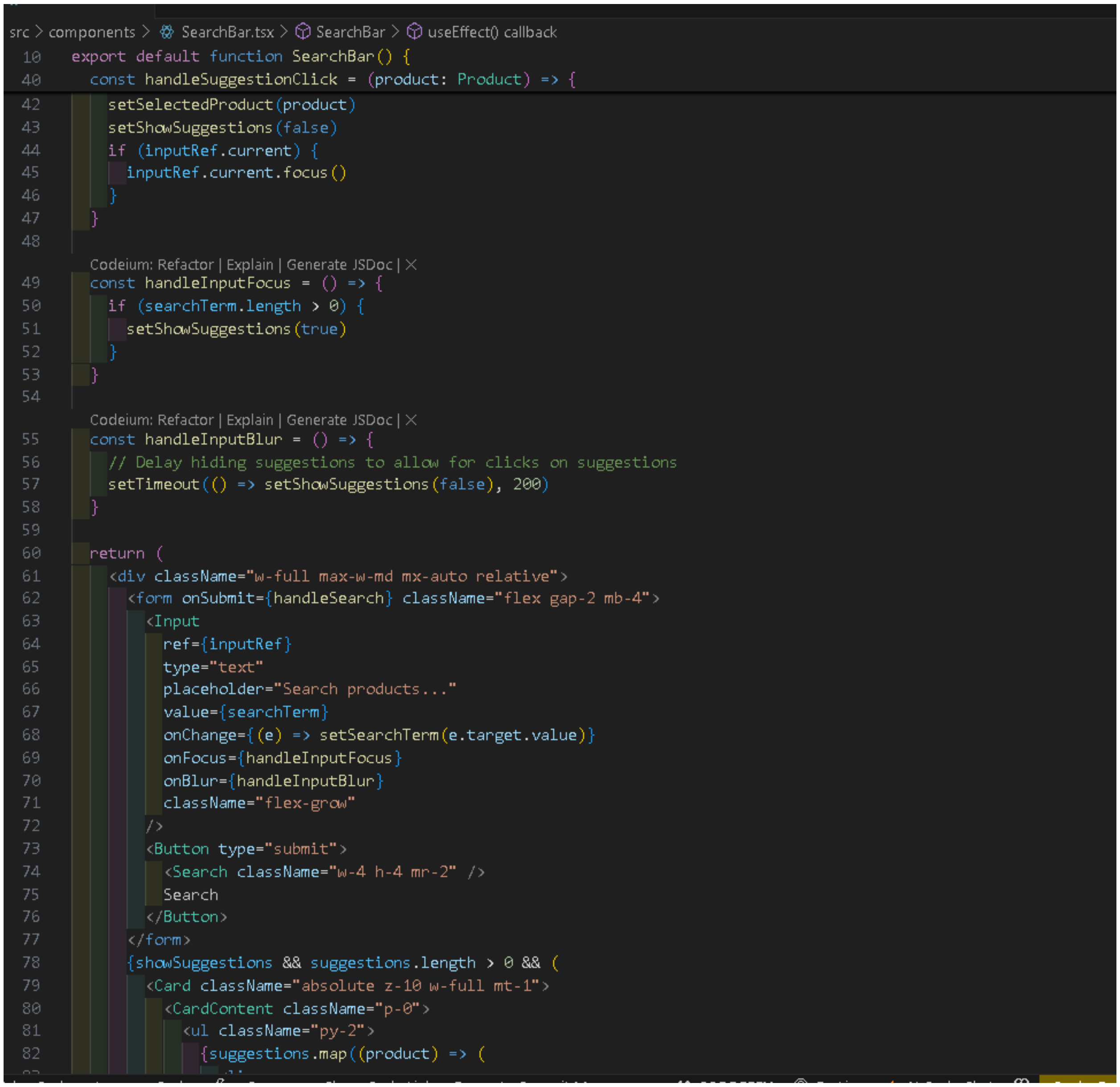


src > components > SearchBar.tsx > SearchBar > useEffect() callback

Edit file using CodeParrot (ctrl+h)

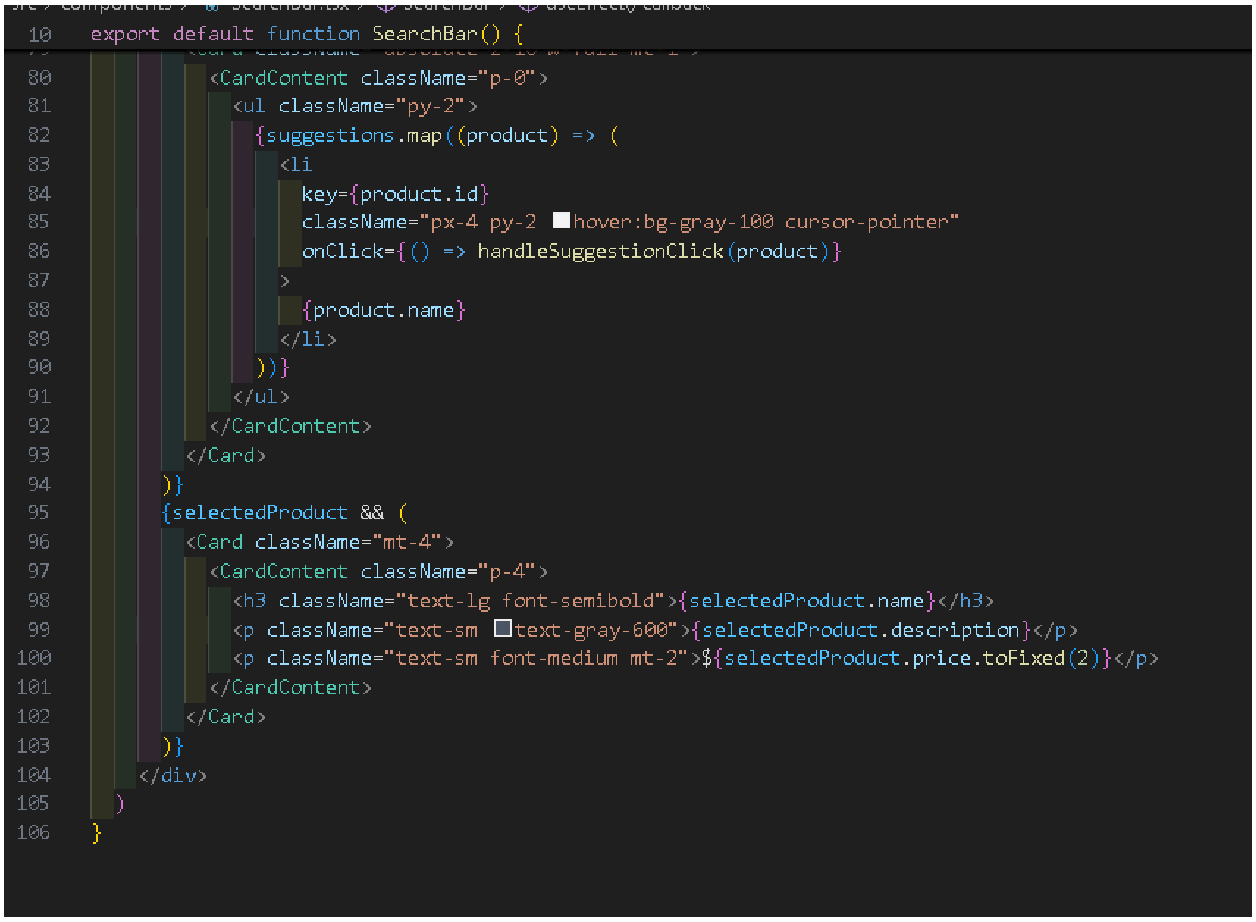
```tsx
"use client"

import { useState, useEffect, useRef } from "react"
import { Input } from "@/components/ui/input"
import { Button } from "@/components/ui/button"
import { Card, CardContent } from "@/components/ui/card"
import { Search } from "lucide-react"
import { type Product, products } from "@/data/productData"

export default function SearchBar() {
  const [searchTerm, setSearchTerm] = useState("")
  const [suggestions, setSuggestions] = useState<Product[]>([])
  const [showSuggestions, setShowSuggestions] = useState(false)
  const [selectedProduct, setSelectedProduct] = useState<Product | null>(null)
  const inputRef = useRef<HTMLInputElement>(null)

  useEffect(() => {
    if (searchTerm.length > 0) {
      const filteredProducts = products.filter(
        (product) =>(
          product.name.toLowerCase().includes(searchTerm.toLowerCase()) ||
          product.description.toLowerCase().includes(searchTerm.toLowerCase()) ||
          product.price.toString().includes(searchTerm)
        )
      );
      setSuggestions(filteredProducts)
      setShowSuggestions(true)
    } else {
      setSuggestions([])
      setShowSuggestions(false)
    }
  }, [searchTerm])

  const handleSearch = (e: React.FormEvent) => {
    e.preventDefault()
    // Perform search action here
    console.log("Searching for:", searchTerm)
  }

  const handleSuggestionClick = (product: Product) => {
    setSearchTerm(product.name)
    setSelectedProduct(product)
```

Abdul Wahid Roll no. 4914



src > components > SearchBar.tsx > SearchBar > useEffect() callback

```
export default function SearchBar() {
  const handleSuggestionClick = (product: Product) => {
    setSelectedProduct(product)
    setShowSuggestions(false)
    if (inputRef.current) {
      inputRef.current.focus()
    }
  }

  Codeium: Refactor | Explain | Generate JSDoc | ×
  const handleInputFocus = () => {
    if (searchTerm.length > 0) {
      setShowSuggestions(true)
    }
  }

  Codeium: Refactor | Explain | Generate JSDoc | ×
  const handleInputBlur = () => {
    // Delay hiding suggestions to allow for clicks on suggestions
    setTimeout(() => setShowSuggestions(false), 200)
  }

  return (
    <div className="w-full max-w-md mx-auto relative">
      <form onSubmit={handleSearch} className="flex gap-2 mb-4">
        <Input
          ref={inputRef}
          type="text"
          placeholder="Search products..."
          value={searchTerm}
          onChange={(e) => setSearchTerm(e.target.value)}
          onFocus={handleInputFocus}
          onBlur={handleInputBlur}
          className="flex-grow"
        />
        <Button type="submit">
          <Search className="w-4 h-4 mr-2" />
          Search
        </Button>
      </form>
      {showSuggestions && suggestions.length > 0 && (
        <Card className="absolute z-10 w-full mt-1">
          <CardContent className="p-0">
            <ul className="py-2">
              {suggestions.map((product) => (
```



```
export default function SearchBar() {
          <CardContent className="p-0">
            <ul className="py-2">
              {suggestions.map((product) => (
                <li
                  key={product.id}
                  className="px-4 py-2 hover:bg-gray-100 cursor-pointer"
                  onClick={() => handleSuggestionClick(product)}
                >
                  {product.name}
                </li>
              ))}
            </ul>
          </CardContent>
        </Card>
      )}
      {selectedProduct && (
        <Card className="mt-4">
          <CardContent className="p-4">
            <h3 className="text-lg font-semibold">{selectedProduct.name}</h3>
            <p className="text-sm text-gray-600">{selectedProduct.description}</p>
            <p className="text-sm font-medium mt-2">${selectedProduct.price.toFixed(2)}</p>
          </CardContent>
        </Card>
      )}
    </div>
  )
}
```

Features:

- **Real-time Filtering:** Users can see the filtered results update instantly without refreshing the page.
- **Responsive Design:** Optimized for both desktop and mobile devices.
- **Performance:** Uses debouncing techniques to improve search performance.

Technologies Used:

- **Frontend:** React components for interactivity.
- **Backend:** GROQ queries to fetch filtered data from Sanity CMS.

Pagination

Pagination was implemented to improve the user experience by:

- Dividing the product listing into smaller, more manageable pages.
- Displaying navigation buttons (e.g., Previous, Next) for easy browsing.

Features:

- Dynamically calculates the total number of pages based on the number of products.
- Shows a limited number of pagination links to avoid clutter.
- Highlights the current page for better visibility.

Implementation:

- **API Integration:** The backend API returns paginated data based on the requested page number and page size.
- **Frontend Logic:** React handles dynamic rendering of pages and pagination links.

Dynamic Routing

Dynamic routing ensures scalability and improves the navigational flow of the website. Examples include:

- **Product Listing Page:** /products
- **Product Detail Page:** /products/[id] (e.g., /products/12345 for a specific piece of furniture)
- **Category Filter Pages:** /categories/[category] (e.g., /categories/sofas)

Benefits:

- Enables sharing of specific product details or filtered results through unique URLs.
- Seamless integration with the Next.js router for server-side rendering (SSR) or static site generation (SSG).

Product Listing Page with Dynamic Data

The Product Listing Page fetches dynamic data from Sanity CMS and displays a grid of available furniture items with key information, including:

- Furniture name
- Price
- Thumbnail image
- Short description

Key Features:

- **Dynamic Data:** Automatically updates when new products are added to the database.
- **Lazy Loading:** Loads images and data as the user scrolls, reducing initial load time.

Example:

```
fetch('/api/products')
```

```
 .then((response) => response.json())
 .then((data) => setProducts(data));
```

Product Detail Page

The Product Detail Page provides detailed information about a specific piece of furniture, including:

- High-resolution images
- Full description
- Specifications (e.g., dimensions, materials, weight capacity)
- Purchasing options

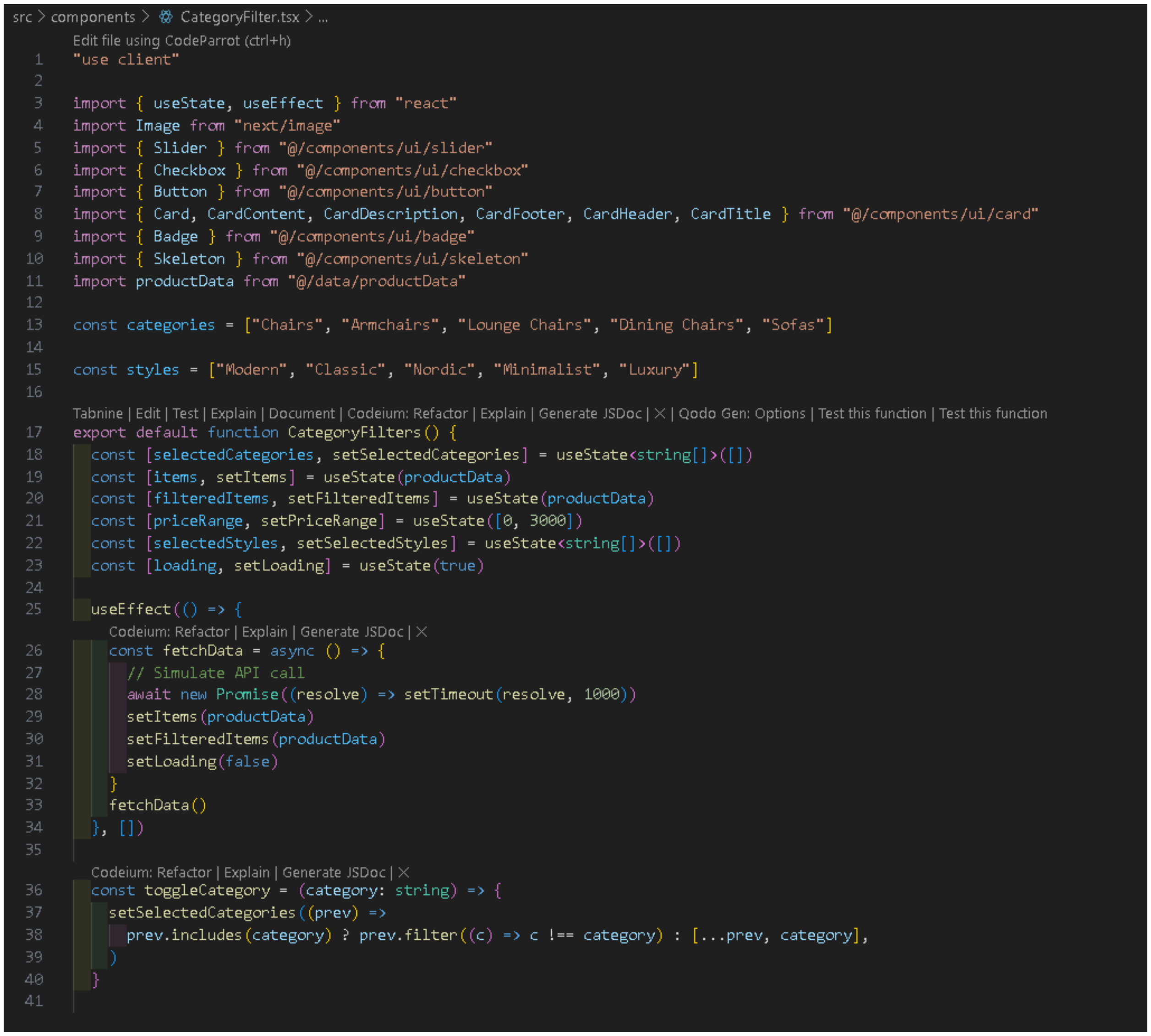
Implementation:

- Fetches data dynamically using the product's unique ID.
- Includes a "Back to Listing" button for easy navigation.

Working Category Filters

The Category Filter allows users to browse furniture based on predefined categories, such as:

- Chairs
- Sofas


src > components > CategoryFilter.tsx > ...
Edit file using CodeParrot (ctrl+h)
Tabnine | Edit | Test | Explain | Document | Codeium: Refactor | Explain | Generate JSDoc | × | Qodo Gen: Options | Test this function | Test this function
Codeium: Refactor | Explain | Generate JSDoc | ×
Codeium: Refactor | Explain | Generate JSDoc | ×

```tsx
"use client"

import { useState, useEffect } from "react"
import Image from "next/image"
import { Slider } from "@/components/ui/slider"
import { Checkbox } from "@/components/ui/checkbox"
import { Button } from "@/components/ui/button"
import { Card, CardContent, CardDescription, CardFooter, CardHeader, CardTitle } from "@/components/ui/card"
import { Badge } from "@/components/ui/badge"
import { Skeleton } from "@/components/ui/skeleton"
import productData from "@/data/productData"

const categories = ["Chairs", "Armchairs", "Lounge Chairs", "Dining Chairs", "Sofas"]

const styles = ["Modern", "Classic", "Nordic", "Minimalist", "Luxury"]

export default function CategoryFilters() {
  const [selectedCategories, setSelectedCategories] = useState<string[]>([])
  const [items, setItems] = useState(productData)
  const [filteredItems, setFilteredItems] = useState(productData)
  const [priceRange, setPriceRange] = useState([0, 3000])
  const [selectedStyles, setSelectedStyles] = useState<string[]>([])
  const [loading, setLoading] = useState(true)

  useEffect(() => {
    const fetchData = async () => {
      // Simulate API call
      await new Promise((resolve) => setTimeout(resolve, 1000))
      setItems(productData)
      setFilteredItems(productData)
      setLoading(false)
    }
    fetchData()
  }, [])

  const toggleCategory = (category: string) => {
    setSelectedCategories((prev) =>
      prev.includes(category) ? prev.filter((c) => c !== category) : [...prev, category],
    )
  }

```

Key Features:

- **Dynamic Querying:** GROQ queries fetch filtered data based on the selected category.
- **Interactive UI:** Clicking on a category updates the listing page without a full page reload.

Conclusion

These features collectively enhance the functionality and usability of the furniture e-commerce website. By implementing efficient filtering, pagination, dynamic routing, and detailed product pages, the platform delivers an engaging and user-friendly experience for potential customers.