

# **Testing, ErrorHandling, and Backend Integration Refinement-Urban-Stitch**

Prepared By: Mustafa Qazi

Student ID: 00010241

First published: PKT 11:00 PM, January 20, 2025

Document Revision Information

Version	Date	Amendment	Author
1.0	11:00 PM, January 20, 2025	Initial release	Mustafa Qazi
2.0	3:00 PM, January 29, 2025	Modified product structure, resolved sanity backend product integration issue	Mustafa Qazi

## Overview

I tried to implement Key Features to enhance functionality of e-commerce website, Urban Stitch ensuring dynamic and user-friendly experience

## Features

- Product Listing Page
- Product detail page
- Add to Cart functionality with Redux
- Pagination with ShadCN UI
- Responsive design with Tailwind CSS

## Technical Workflow

- **Product Listing Page Workflow**
  - Fetched dynamic data from Sanity CMS to populate the product grid
  - Integrated category filters, search bar, and pagination to allow users to browse and locate products seamlessly
- **Category Page**
  - Dynamically fetched products based on their categories using Sanity CMS.
  - Rendered category-specific product listings with pagination for better user navigation.
- **Product Detail Page**
  - Implemented individual product detail pages with accurate dynamic routing using Next.js.
  - Each detail page renders data dynamically from Sanity CMS
- **Add to Cart Workflow**
  - Developed a slice using Redux Toolkit to manage cart state
  - Configured store.ts to include persistence middleware for saving cart data in local storage
  - Connected the toast notification component to trigger upon successful cart actions
- **Responsive design**
  - Thoroughly tested and adjusted the layout for different screen sizes, ensuring consistent functionality and aesthetics.
- **Toastify Notifications**
  - Created a client-rendered component to manage toast notifications
  - Added success messages to confirm user actions, such as adding items to the cart.

## Product Listing Code

```
app > product > [id] > ⌘ page.tsx > ⌘ star
"use client";
import Image from "next/image";
import { FaStar } from "react-icons/fa";
import { Minus, Plus } from "lucide-react";
import { useEffect, useState } from "react";
import { client } from "@sanity/lib/client";
import { urlFor } from "@sanity/lib/image";
import CustomerTestimonials from "@components/AllReviews";
import Top_sell from "@components/arrivals";
import { BreadcrumbCollapsed } from "@components/Breadcrupm";
import { useDispatch } from "react-redux";
import Toastify from "@app/cart/toastify";

// Adding key prop in star array
let star = [
  <FaStar key={1} />,
  <FaStar key={2} />,
  <FaStar key={3} />,
  <FaStar key={4} />,
  <FaStar key={5} />,
];

interface Iproducts {
  image: string[];
  discountPercent: number;
  isNew: boolean;
  name: string;
  description: string;
  price: number;
  _id: string;
  colors: string[];
  sizes: string[];
}

export default function SlugPage({ params }: { params: { id: string } }) {
  const [product, setProduct] = useState<Iproducts | null>(null);
  const [cartItem, setCartItem] = useState<any>(null);
```

```

}

export default function SlugPage({ params }: { params: { id: string } }) {
  const [product, setProduct] = useState<Iproducts | null>(null);
  const [cartItem, setCartItem] = useState<any>(null);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(false);
  const dispatch = useDispatch();

  useEffect(() => {
    const fetchProduct = async () => {
      try {
        setLoading(true);
        const products: Iproducts[] = await client.fetch(
          `*[ _type == 'products' ] {
            image: image.asset->url,
            category,
            discountPercent,
            isNew,
            name,
            description,
            price,
            _id,
            colors,
            sizes
          }`
        );
        const slug = products.find((item) => item._id === params.id);

        if (!slug) {
          setError(true);
        } else {
          setProduct(slug);
          setCartItem({
            id: slug._id,
            name: slug.name,
```

## Dynamic routing Code

```
import Image from 'next/image';
import Link from 'next/link';
import { FaStar } from 'react-icons/fa';

interface Iproducts{
  title:string,
  price:string,
  id:number
  rating?:string,
  old_price?:string
  img_url:string
}

let product:Iproducts[] = [
  {
    title:"T-Shirt with Tape Details",
    id:1,
    price:"$120",
    img_url:"/product1.png"
  },
  {
    title:"Skinny Fit Jeans",
    id:2,
    price:"$240",
    img_url:"/product2.png",
    old_price:"$260"
  },
  {
    title:"Checkered Shirt",
    id:3,
    price:"$180",
    img_url:"/product3.png"
  },
  {
    title:"Sleeve Striped T-Shirt",

```

```

    // Adding key prop in star array
    let star = [
      <FaStar key={1} />,
      <FaStar key={2} />,
      <FaStar key={3} />,
      <FaStar key={4} />,
      <FaStar key={5} />,
    ];
    export default function Products(){
      return(
        <div className="w-full h-full sm:h-[500px] mt-10">
          <h1 className="text-3xl md:text-4xl font-bold text-center">NEW ARRIVALS</h1>
          <div className="flex-col md:flex-row justify-center items-center md:justify-between px-0 mt-10">
            {
              product.map((data)=>{
                return(
                  <div key={data.id}>
                    <Link href="/products/${data.id}" />
                    <div className="w-[180px] h-[180px] md:w-[280px] md:h-[280px] bg-[#f0f0f0] rounded-[20px]"
                      <Image src={data.img_url} alt={data.title}
                        className="w-full h-full rounded-[20px]"
                        width={180} height={180}></Image>
                    </div>
                    </Link>
                    <div>
                      <p className="text-lg mt-2 font-bold">{data.title}</p>
                      <div className="flex text-yellow-400">
                        {star.map((icon, index) => (
                          <span key={index}>{icon}</span>
                        ))}
                      </div>
                      <p className="font-bold mt-1">{data.price} <span className="text-gray-400 font-bold line-through"> {data.old_price}</span>
                    </div>
                  </div>
                )
              })
            }
          </div>
        </div>
      )
    }
  }

```

## Redux Store Code

```
c> app > Redux > TS store.ts > ...
1 import { combineReducers, configureStore } from '@reduxjs/toolkit';
2 import cartSlice from '../features/cart';
3 import { persistReducer, persistStore } from 'redux-persist';
4 import createWebStorage from 'redux-persist/lib/storage/createWebStorage';
5
6 // Create a fallback for SSR
7 const createNoopStorage = () => ({
8   getItem(_key: string) {
9     return Promise.resolve(null);
10   },
11   setItem(_key: string, _value: any) {
12     return Promise.resolve();
13   },
14   removeItem(_key: string) {
15     return Promise.resolve();
16   },
17 });
18
19 // Use the browser storage in the client and noop storage in SSR
20 const storage =
21   typeof window !== 'undefined' ? createWebStorage('local') : createNoopStorage();
22
23 // Redux persist configuration
24 const persistConfig = {
25   key: 'root',
26   version: 1,
27   storage, // Use the appropriate storage
28 };
29
30 // Combine reducers
31 const rootReducer = combineReducers({
32   cart: cartSlice, // Add additional reducers here
33 });
34
35 // Persisted reducer
36 const persistedReducer = persistReducer(persistConfig, rootReducer);
37
38 // Configure the Redux store
39 export const store = configureStore({
40   reducer: persistedReducer,
41   middleware: (getDefaultMiddleware) =>
42     getDefaultMiddleware({
43       serializableCheck: false, // Disable serializable checks for redux-persist
44     }),
45 });
46
47 // Create the persistor
48 export const persistor = persistStore(store);
49
50 // Infer the `RootState` and `AppDispatch` types
51 export type RootState = ReturnType<typeof store.getState>;
52 export type AppDispatch = typeof store.dispatch;
53
```

## Hooks Code

```
src > app > Redux > TS hooks.ts > ...
1 import { useDispatch, useSelector } from 'react-redux'
2 import type { AppDispatch, RootState } from './store'
3
4 // Use throughout your app instead of plain `useDispatch` and `useSelector`
5 export const useAppDispatch = useDispatch.withTypes<AppDispatch>()
6 export const useAppSelector = useSelector.withTypes<RootState>()
```

## Cart Slice Code

```
rc > app > Redux > features > TS cart.ts > cartSlice > reducers
1 import { createSlice, PayloadAction } from '@reduxjs/toolkit'
2
3 // Define the initial state using that type
4
5 export const cartSlice = createSlice({
6   name: 'products',
7   // `createSlice` will infer the state type from the `initialState` argument
8   initialState: [],
9   reducers: [
10    // add to cart functionality
11    add(state:any,action){
12      let uuid = Math.floor(1000+Math.random()*9000)
13      let newobj = {...action.payload,uuid}
14      state.push(newobj)
15    },
16    // delete from cart
17    remove(state:any,{payload}){
18      return state.filter((val:any)=> val.uuid !== payload)
19    },
20    // addition of item
21    addition(state:any,action){
22      let obj = state.find(
23        (val:any)=>
24        val.id == action.payload.id &&
25        val.color == action.payload.color &&
26        val.size == action.payload.size
27      );
28      if(obj){
29        ++obj.qty;
30        let newState = state.filter((val:any)=> val.id !== obj.id);
31        state = [...newState,obj];
32        return
33      }
34    },
35    // subtraction of item
36    subtraction(state:any,action){
37      let obj = state.find(
```

## Provider Code

```
providers.tsx U X
src > app > providers.tsx > ...
1  "use client"
2  import { Provider } from "react-redux";
3  import { store } from "../app/Redux/store";
4  import { PersistGate } from "redux-persist/integration/react";
5  import { persistStore } from "redux-persist";
6
7
8  function Providers({ children, }: Readonly<{children: React.ReactNode;}>) {
9    let persistore = persistStore(store)
10   return (
11     <Provider store={store}>
12       <PersistGate persistor={persistore}>
13         {children}
14       </PersistGate>
15     </Provider>
16   )
17 }
18
19 export default Providers
```



## Pagination Code

```
rc > components > pagination.tsx > ...
1  import {
2      Pagination,
3      PaginationContent,
4      PaginationEllipsis,
5      PaginationItem,
6      PaginationLink,
7      PaginationNext,
8      PaginationPrevious,
9  } from "@components/ui/pagination"
10 import React from 'react'
11
12 function Paginationpage() {
13     return (
14         <div className="w-full mt-4">
15             <Pagination>
16                 <PaginationContent className="w-full space-x-4 flex justify-center lg:ml-7">
17                     <PaginationItem>
18                         <PaginationPrevious href="/" />
19                     </PaginationItem>
20
21                     <PaginationItem className="hidden lg:block">
22                         <PaginationLink href="/">Home</PaginationLink>
23                     </PaginationItem>
24                     <PaginationItem className="hidden lg:block">
25                         <PaginationEllipsis />
26                     </PaginationItem>
27
28                     <PaginationItem className="hidden lg:block">
29                         <PaginationLink href="/sell">Top Sell</PaginationLink>
30                     </PaginationItem>
31                     <PaginationItem className="hidden lg:block">
32                         <PaginationEllipsis />
33                     </PaginationItem>
34
35                     <PaginationItem>
36                         <PaginationLink href="/brand">Brands</PaginationLink>
37                     </PaginationItem>

```

## Toastify Code

```
toastify.tsx U X
src > app > cart > toastify.tsx > Toastify > dispatch
1  "use client"
2  import React from 'react';
3  import { useDispatch } from 'react-redux';
4  import { Bounce, ToastContainer, toast } from 'react-toastify';
5  import "react-toastify/ReactToastify.css";
6  import { add } from '../Redux/features/cart';
7  import { Button } from '@components/ui/button';
8
9
10 function Toastify({cartItem}:any) {
11   ⚠
12   const dispatch = useDispatch()
13
14   const handleadd = (cartItem:any)=>{
15     dispatch(add(cartItem))
16   }
17
18
19   const notify = () =>
20   toast.success('Product added Successfully!', {
21     position: "bottom-right",
22     autoClose: 5000,
23     hideProgressBar: false,
24     closeOnClick: false,
25     pauseOnHover: true,
26     draggable: true,
27     progress: undefined,
28     theme: "light",
29     transition: Bounce,
30   });
31   return (
32     <>
33     <div onClick={()=>handleadd(cartItem)}>
34       <Button onClick={notify} className="bg-black text-white w-[300px]"
35         >Add to Cart</Button>
36     </div>
37     <ToastContainer
```

# Product Listing


Sign up and get 20% off to your first order. Sign Up Now Sign Up Now

URBAN STITCH


Shop On Sale New Arrivals Brands

Search for products...


NEW ARRIVALS




Gradient Graphic T-shirt  
★★★★★  
145



Sleeve Stripe T-Shirt  
★★★★★  
130 40



Classic Polo Shirt  
★★★★★  
180



COURAGE GRAPHIC T-SHIRT  
★★★★★  
145

View all

# Product Detail Page




Sign up and get 20% off to your first order. Sign Up Now Sign Up Now


URBAN STITCH

Shop On Sale New Arrivals Brands

Search for products...

Home > Shop > Mens T-shirts >





Sleeve Stripe T-Shirt

★★★★★

130 78 40%

This product is a vibrant orange and black striped t-shirt with a sporty design. The shirt features vertical pinstripes in black on an orange base, complemented by contrasting black raglan sleeves for a casual, athletic-inspired look. Ideal for adding a bold touch to your casual wardrobe, this tee combines comfort and style, perfect for daily wear or sporting events. The soft fabric ensures a comfortable fit, while the unique design makes it a standout piece for any outfit.

Select Colors

Choose Size

S

L

XL

M

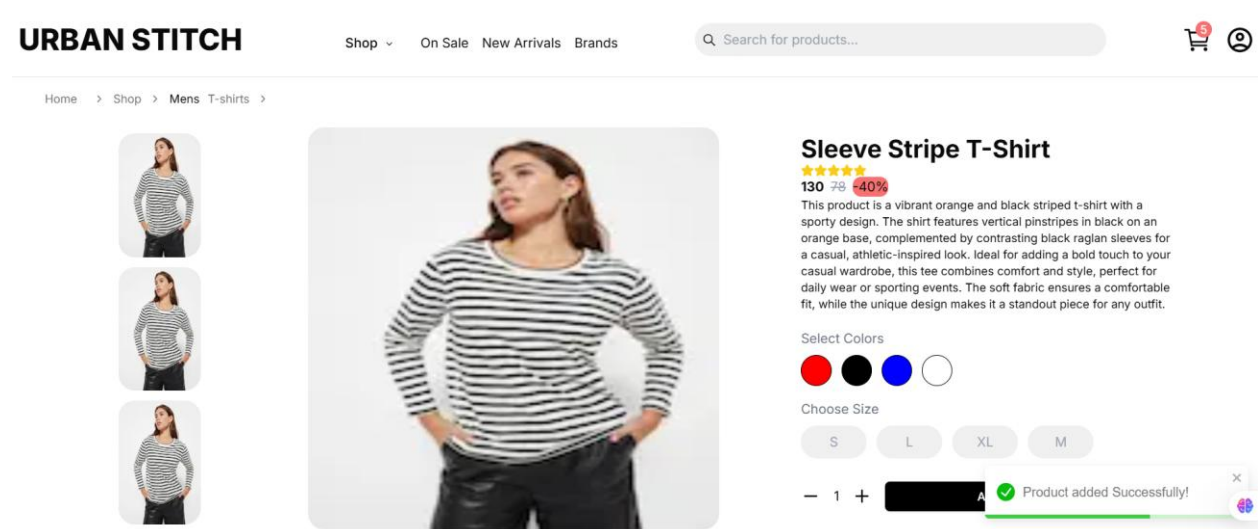
1

+

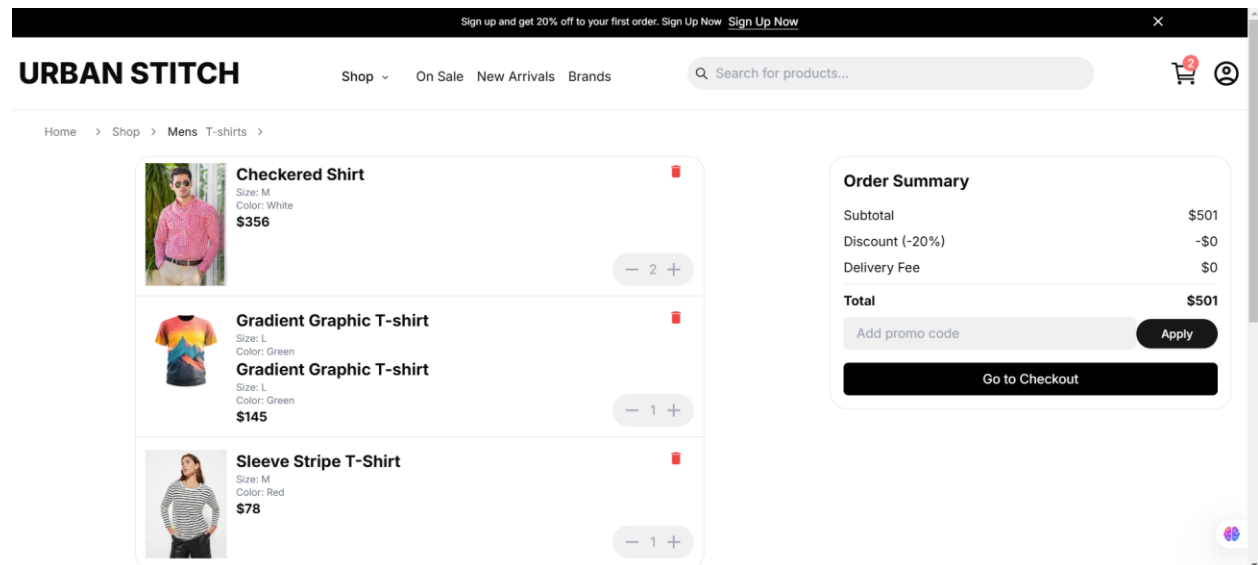
Add to Cart

Page 10 of 17

## Toastify notification



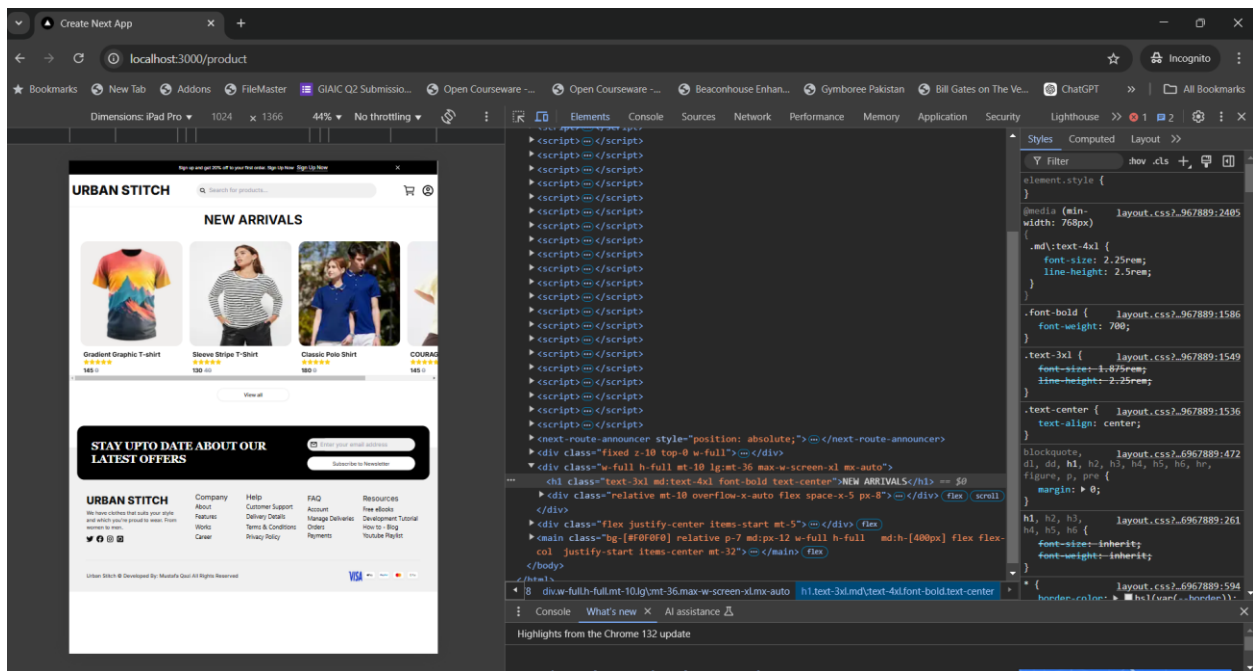
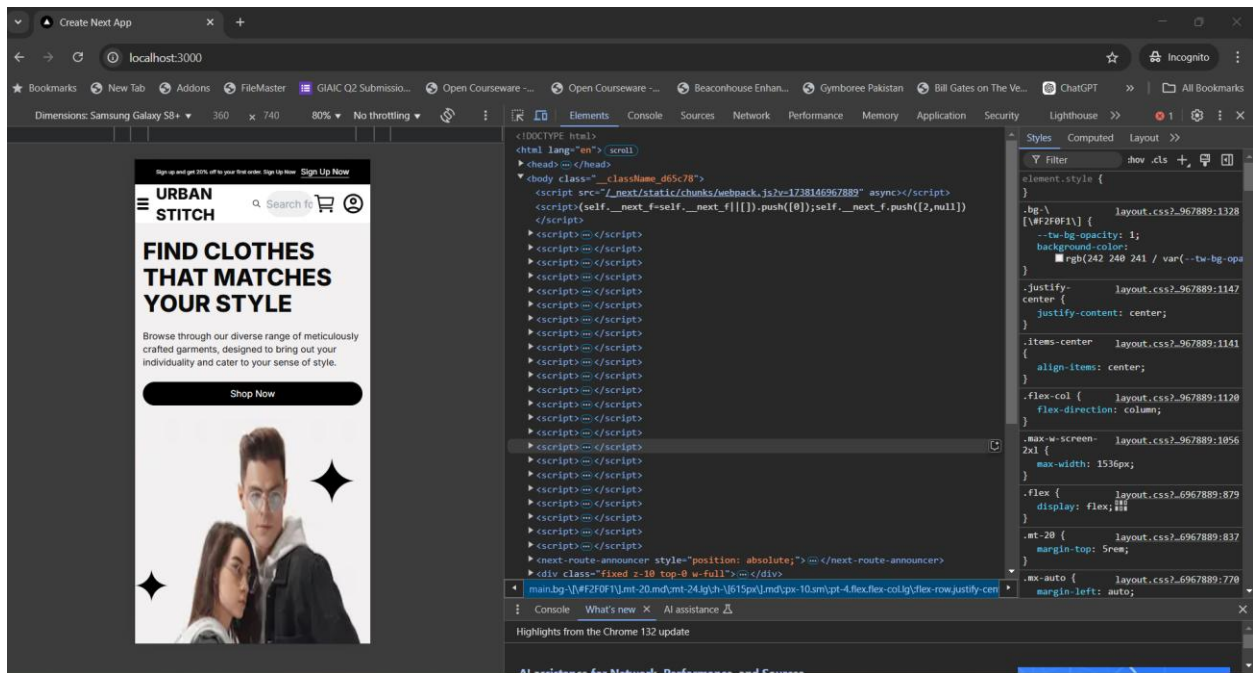
## Cart Page



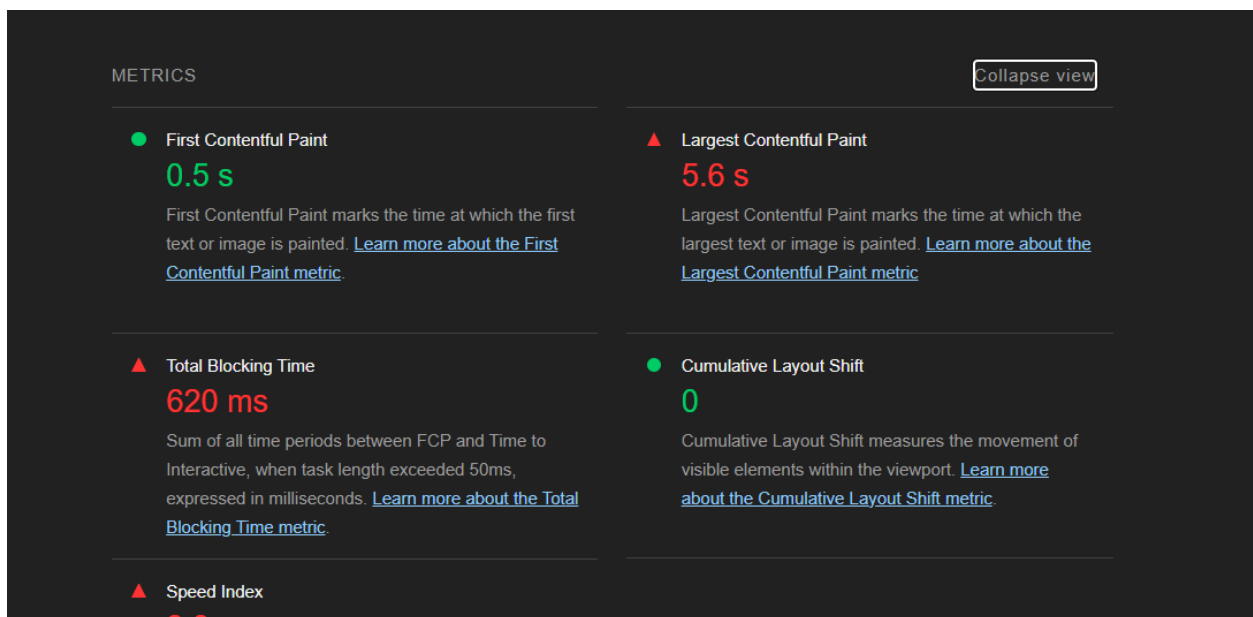
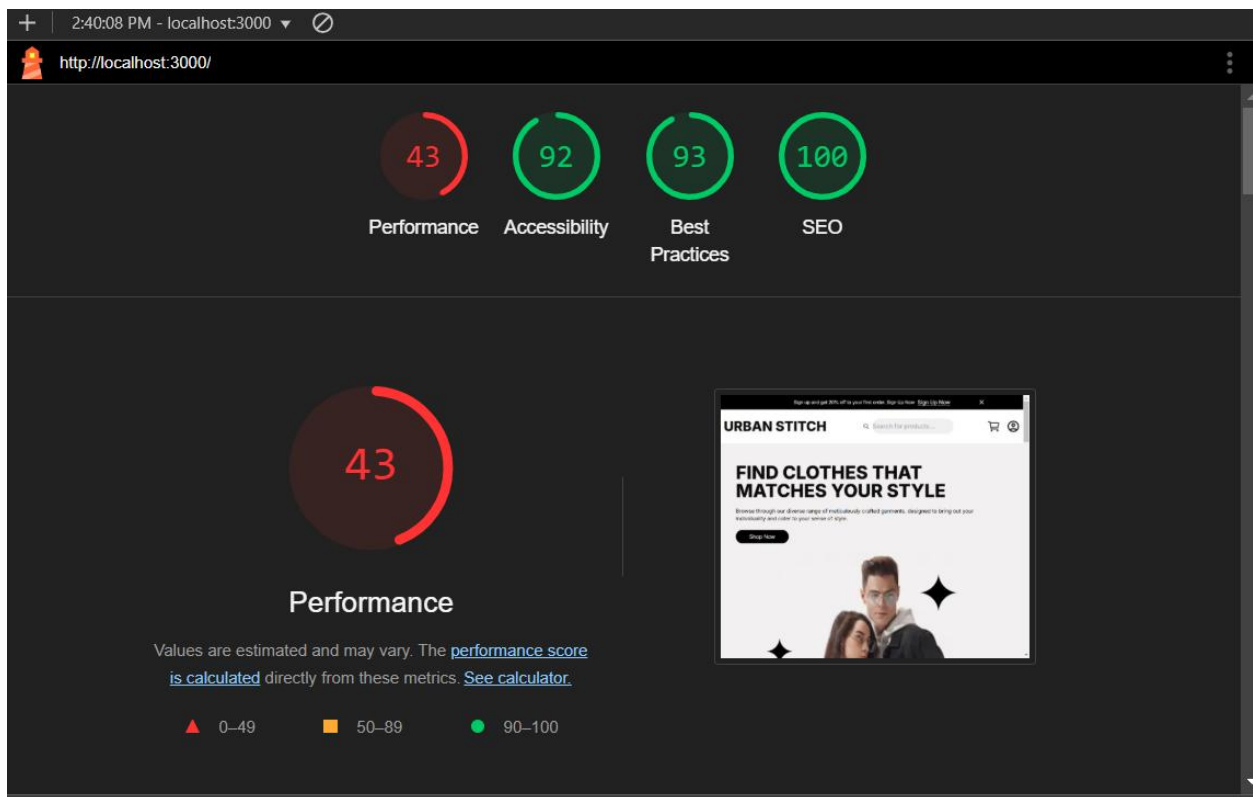
## Payment Gateway Integration

Will use **Stripe** for Payment Gateway Integration

## Responsive



## Lighthouse testing tool – Screenshots



DIAGNOSTICS		
▲	Largest Contentful Paint element — 5,620 ms	▼
▲	Reduce unused JavaScript — Potential savings of 154 KiB	▼
▲	Page prevented back/forward cache restoration — 3 failure reasons	▼
■	Minify CSS — Potential savings of 2 KiB	▼
■	Minify JavaScript — Potential savings of 5 KiB	▼
■	Avoid serving legacy JavaScript to modern browsers — Potential savings of 0 KiB	▼
■	Avoid enormous network payloads — Total size was 6,215 KiB	▼
○	JavaScript execution time — 1.0 s	▼

PASSED AUDITS (23)		Hide
●	Eliminate render-blocking resources	▼
●	Properly size images	▼
●	Defer offscreen images	▼
●	Reduce unused CSS	▼
●	Efficiently encode images	▼
●	Serve images in next-gen formats	▼
●	Enable text compression	▼
●	Preconnect to required origins	▼



## Accessibility

These checks highlight opportunities to [improve the accessibility of your web app](#). Automatic detection can only detect a subset of issues and does not guarantee the accessibility of your web app, so [manual testing](#) is also encouraged.

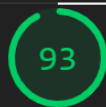
### CONTRAST

- ▲ Background and foreground colors do not have a sufficient contrast ratio.

These are opportunities to improve the legibility of your content.

### NAMES AND LABELS

- ▲ Links do not have a discernible name



## Best Practices

### USER EXPERIENCE

- ▲ Displays images with incorrect aspect ratio

### GENERAL

- ▲ Browser errors were logged to the console
- ▲ Missing source maps for large first-party JavaScript

### TRUST AND SAFETY



100

## SEO

These checks ensure that your page is following basic search engine optimization advice. There are many additional factors Lighthouse does not score here that may affect your search ranking, including performance on [Core Web Vitals](#). [Learn more about Google Search Essentials](#).

ADDITIONAL ITEMS TO MANUALLY CHECK (1)

Hide

Structured data is valid

Run these additional validators on your site to check additional SEO best practices.

PASSED AUDITS (8)

Show

## Postman Testing tool Results

My Workspace

NewImport

OverviewGETUntitled RequestPOSThttp://localhost:3000:GETGet data

REST API basics: CRUD, test & variable / Get data

GEThttp://localhost:3000Send

ParamsAuthorizationHeaders (6)BodyScriptsSettingsCookies

Query Params

Key	Value	Description
Key	Value	Description

BodyCookiesHeaders (9)Test Results (1/1)

200 OK96 ms16.34 KB

HTMLPreviewVisualize

Collections

REST API basics: CRUD, test & variable

GETGet data

POSTPost data

PUTUpdate data

DELETEDelete data

Environments

Flows

History

Save

Share

Info

Code

Help

My Workspace

NewImport

OverviewGET Untitled RequestPOST http://localhost:3REST API basics: CGET http://localhost:3GET https://template1-

No environment

Collections

Environments

Flows

History

My first collection

First folder inside collection

Second folder inside collection

Create a collection for your requests

A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.

Create Collection

https://template1-neon-nu.vercel.app/api/products

SaveShare

GEThttps://template1-neon-nu.vercel.app/api/products

Send

ParamsAuthorizationHeaders (6)BodyScriptsSettings

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

BodyCookiesHeaders (12)Test Results

200 OK1.54 s5.76 KB

{ } JSONPreviewVisualize

_id	discountPercent	name	description	price	imageUrl
0dc7c847-8599-45d0-b02c-34429f7a639e	20	Casual Green Bomber Jacket	This stylish green bomber jacket offers a sleek and modern twist on a classic design. Made from soft and comfortable fabric, it features snap buttons and ribbed cuffs, giving it a	300	https://cdn.sanity.io/images/7xt4qcah/production/4e2ed6a9eaa6e1413843e53f3113ccfd2104c301-278x296.png