DYNAMIC FRONTEND COMPONENTS-URBANSTITCH

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

Table of Contents

Overview	2
Features and Technical Workflow	2
Product Listing Code Screenshot	3
Dynamic Routing Code Screenshot	4
Redux and Hooks	5
Cart Slice	6
Provider	7
Pagination	8
Toastify	9
Product Listing and Product Detail Front end Page	10
Toastify notification	10
Cart Page	11

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

- o 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

```
pp > product > [id] > 🏶 page.tsx > 📵 star
 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";
  <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[];
 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 disispatch = useDispatch()
 useEffect(() => {
   const fetchProduct = async () => {
     try {
       setLoading(true);
       const products: Iproducts[] = await client.fetch(
          `*[ type == 'products']{
           "image": image.asset->url,
           category,
           colors,
       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",
```

Redux Store Code

```
import { combineReducers, configureStore } from '@reduxjs/toolkit';
import cartSlice from './features/cart';
 import { persistReducer, persistStore } from 'redux-persist';
import createWebStorage from 'redux-persist/lib/storage/createWebStorage';
 const createNoopStorage = () => ({
  getItem(_key: string) {
  setItem(_key: string, _value: any) {
   return Promise.resolve();
    return Promise.resolve();
  typeof window !== 'undefined' ? createWebStorage('local') : createNoopStorage();
// Redux persist configuration
const persistConfig = {
  version: 1,
  storage, // Use the appropriate storage
const rootReducer = combineReducers({
  cart: cartSlice, // Add additional reducers here
const persistedReducer = persistReducer(persistConfig, rootReducer);
export const store = configureStore({
  reducer: persistedReducer
  middleware: (getDefaultMiddleware) =>
    getDefaultMiddleware({
       serializableCheck: false, // Disable serializable checks for redux-persist
// Create the persistor
export const persistor = persistStore(store);
export type RootState = ReturnType<typeof store.getState>;
export type AppDispatch = typeof store.dispatch;
```

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
      import { createSlice, PayloadAction } from '@reduxjs/toolkit'
      export const cartSlice = createSlice({
        name: 'products',
        // `createSlice` will infer the state type from the `initialState` argument
        initialState:[],
        reducers: {
          add(state:any,action){
            let uuid = Math.floor(1000+Math.random()*9000)
            let newobj = {...action.payload,uuid}
            state.push(newobj)
           // delete from cart
16
           remove(state:any,{payload}){
             return state.filter((val:any)=> val.uuid !== payload)
          addition(state:any,action){
            let obj = state.find(
              (val:any)=>
                val.id == action.payload.id &&
                val.color == action.payload.color &&
                val.size == action.payload.size
            if(obj){
              ++obj.qty;
              let newState = state.filter((val:any)=> val.id !== obj.id);
              state = [...newState,obj];
              return
                subraction of item
           subraction(state:any,action){
            let obj = state.find(
```

Provider Code

Pagination Code

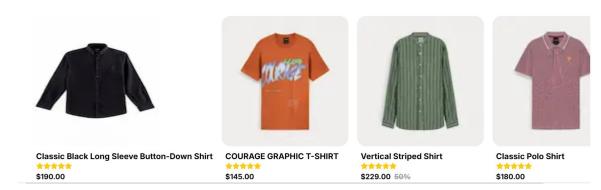
```
rc > components > 🏶 pagination.tsx > ...
      import {
          Pagination,
          PaginationContent,
          PaginationEllipsis,
          PaginationItem,
         PaginationLink,
          PaginationNext,
          PaginationPrevious,
        } from "@/components/ui/pagination"
      import React from 'react'
      function Paginationpage() {
       return (
          <div className="w-full mt-4">
       <Pagination>
       <PaginationContent className="w-full space-x-4 flex justify-center lg:ml-7">
          <PaginationItem>
            <PaginationPrevious href="/" />
          </PaginationItem>
          <PaginationItem className="hidden lg:block">
            <PaginationLink href="/">Home</PaginationLink>
          </PaginationItem>
          <PaginationItem className="hidden lg:block">
            <PaginationEllipsis />
          </PaginationItem>
          <PaginationItem className="hidden lg:block">
            <PaginationLink href="/sell">Top Sell</PaginationLink>
          </PaginationItem>
          <PaginationItem className="hidden lg:block">
            <PaginationEllipsis />
          </PaginationItem>
          <PaginationItem>
            <PaginationLink href="/brand">Brands</PaginationLink>
          </PaginationItem>
```

Toastify Code

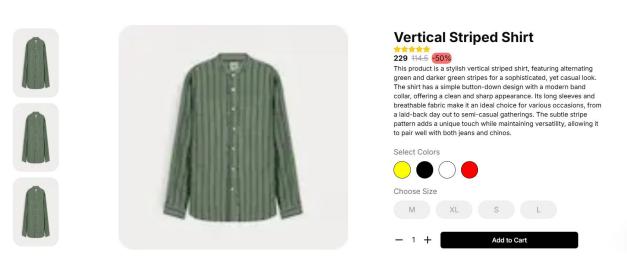
```
src > app > cart > ∰ toastify.tsx > ۞ Toastify > [∅] dispatch
      "use client"
      import React from 'react';
      import { useDispatch } from 'react-redux';
      import { Bounce, ToastContainer, toast } from 'react-toastify';
      import "react-toastify/ReactToastify.css";
      import { add } from '../Redux/features/cart';
      import { Button } from '@/components/ui/button';
       function Toastify({cartItem}:any) {
      const dispatch = useDispatch()
       const handleadd = (cartItem:any)=>{
          dispatch(add(cartItem))
           const notify = () =>
           toast.success('Product added Successfully!', {
             position: "bottom-right",
             autoClose: 5000,
             hideProgressBar: false,
             closeOnClick: false,
             pauseOnHover: true,
             draggable: true,
             progress: undefined,
             theme: "light",
             transition: Bounce,
           <div onClick={()=>handleadd(cartItem)}>
                <Button onClick={notify} className="bg-black text-white w-[300px]"</pre>
                  >Add to Cart</Button>
```

Product Listing

TOP SELLING

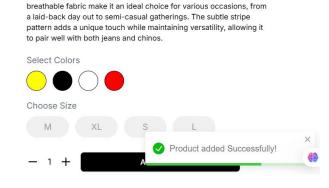


Product Detail Page



Toastify notification



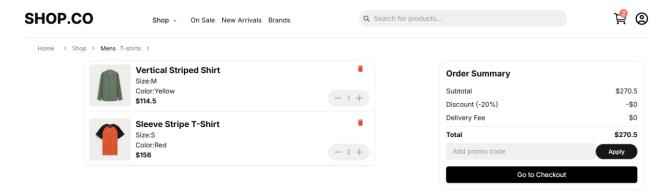


Vertical Striped Shirt

This product is a stylish vertical striped shirt, featuring alternating green and darker green stripes for a sophisticated, yet casual look. The shirt has a simple button-down design with a modern band collar, offering a clean and sharp appearance. Its long sleeves and

229 114.5 -50%

Cart Page



Payment Gateway Integration

Will use **Stripe** for Payment Gateway Integration