

# Marketplace Frontend Components Documentation

## Overview

This document outlines the dynamic frontend components created for the marketplace application. These components are designed to provide a seamless user experience and include product listings, search functionality, shopping cart, checkout flow, reviews, ratings, and consistent header/footer sections.

## 1. Product Listing Component

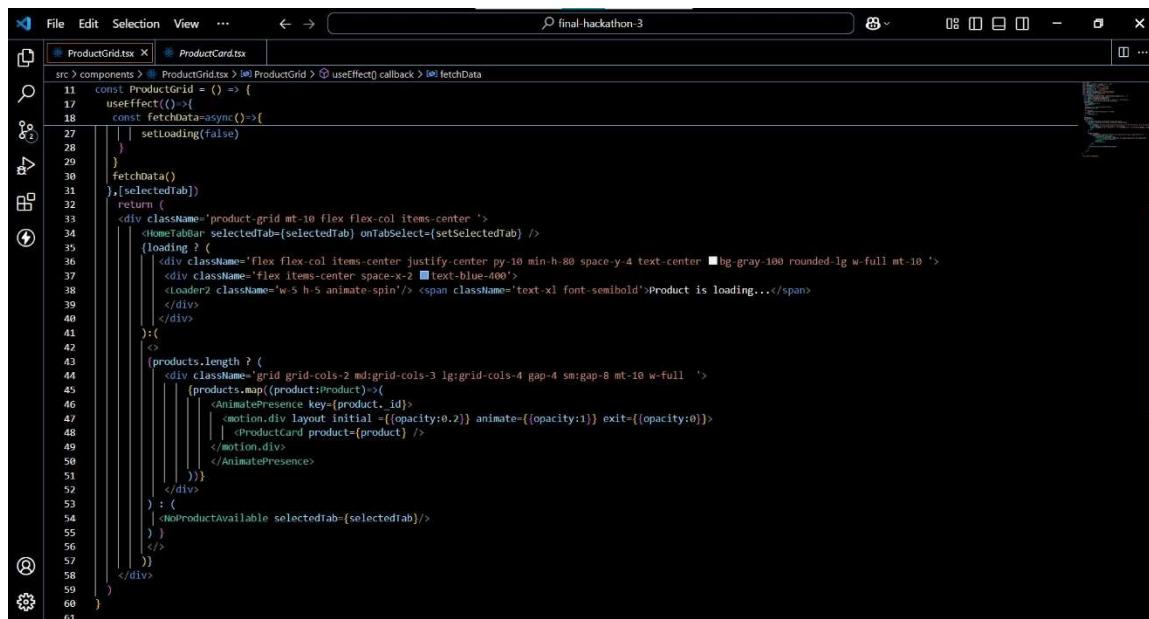
### Description:

The Product Listing Component displays a grid of products fetched dynamically from the API or Sanity CMS. Each product card includes details like the product name, price, image, and stock status.

### Features:

- Grid layout for products.
- Data fetched dynamically via API.
- Displays product name, price, image, and stock status.

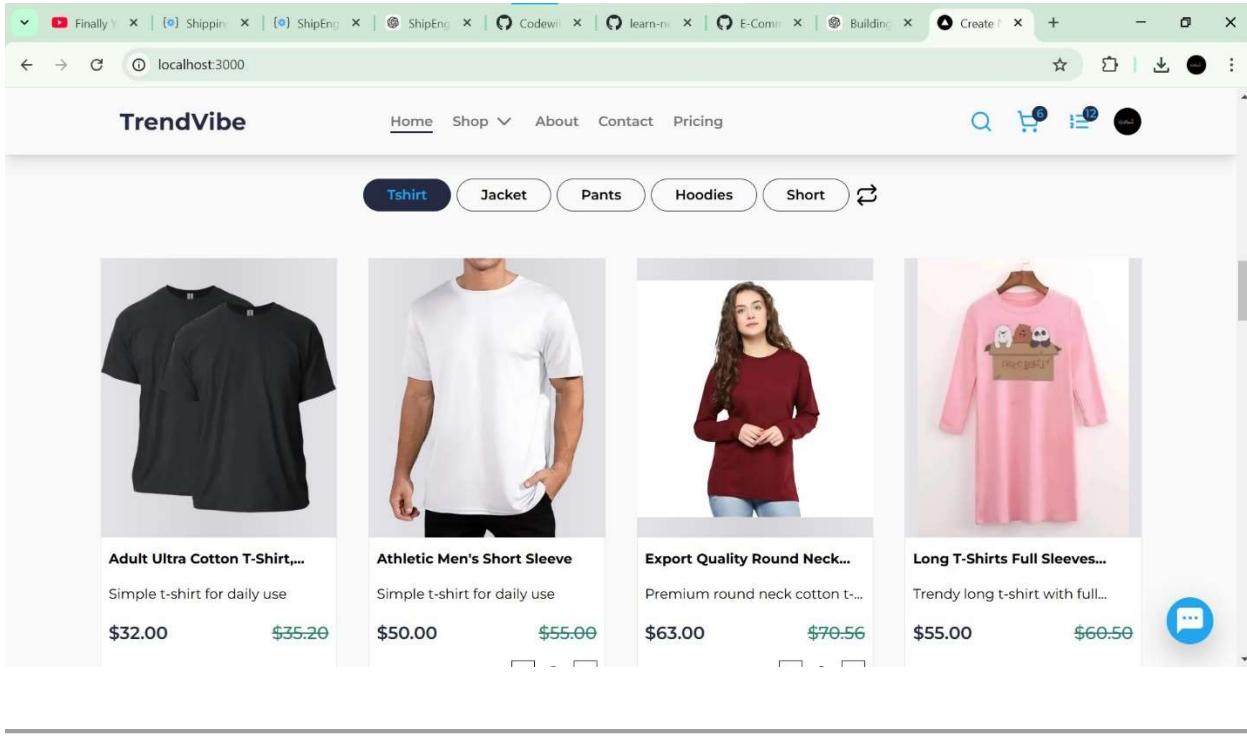
### Code Screenshot:



A screenshot of a code editor window titled "final-hackathon-3". The main pane shows the code for the "ProductGrid.tsx" component. The code uses functional programming and hooks like useEffect and useState. It handles loading states and maps over an array of products to render cards. A sidebar on the right shows a tree view of the project structure, including "ProductCard.tsx" and other components.

```
File Edit Selection View ... ← → ⌂ final-hackathon-3
ProductGrid.tsx ProductCard.tsx
src > components > ProductGrid.tsx > ProductGrid > useEffect() callback > fetchData
11 const ProductGrid = () => {
12   useEffect(() => {
13     const fetchData=async()=>{
27       setLoading(false)
28     }
29   }
30   fetchData()
31 },[selectedTab])
32   return (
33     <div className='product-grid mt-10 flex flex-col items-center'>
34       <HomeTabBar selectedTab={selectedTab} onTabSelect={setSelectedTab} />
35       {loading ? (
36         <div className='flex flex-col items-center justify-center py-10 min-h-80 space-y-4 text-center bg-gray-100 rounded-lg w-full mt-10'>
37           <Loader> <span>Product is loading...</span>
38         </div>
39       ):(
40         <>
41           {products.length ? (
42             <div className='grid grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-4 sm:gap-8 mt-10 w-full'>
43               {products.map((product)=>(
44                 <AnimatePresence key={product.id}>
45                   <motion.div layoutInitial={{opacity:0.2}} animate={{(opacity:1)}} exit={{(opacity:0)}}>
46                     <ProductCard product={product} />
47                   </motion.div>
48                 </AnimatePresence>
49               ))
50             </div>
51           ) : (
52             <NoProductAvailable selectedTab={selectedTab}/>
53           )
54         </>
55       )
56     )
57   )
58 }
59
60
61 }
```

## UI Screenshot:



## 2. Product Detail Component

### Description:

The Product Detail Component displays detailed information about a selected product. This includes a description, and other relevant details.

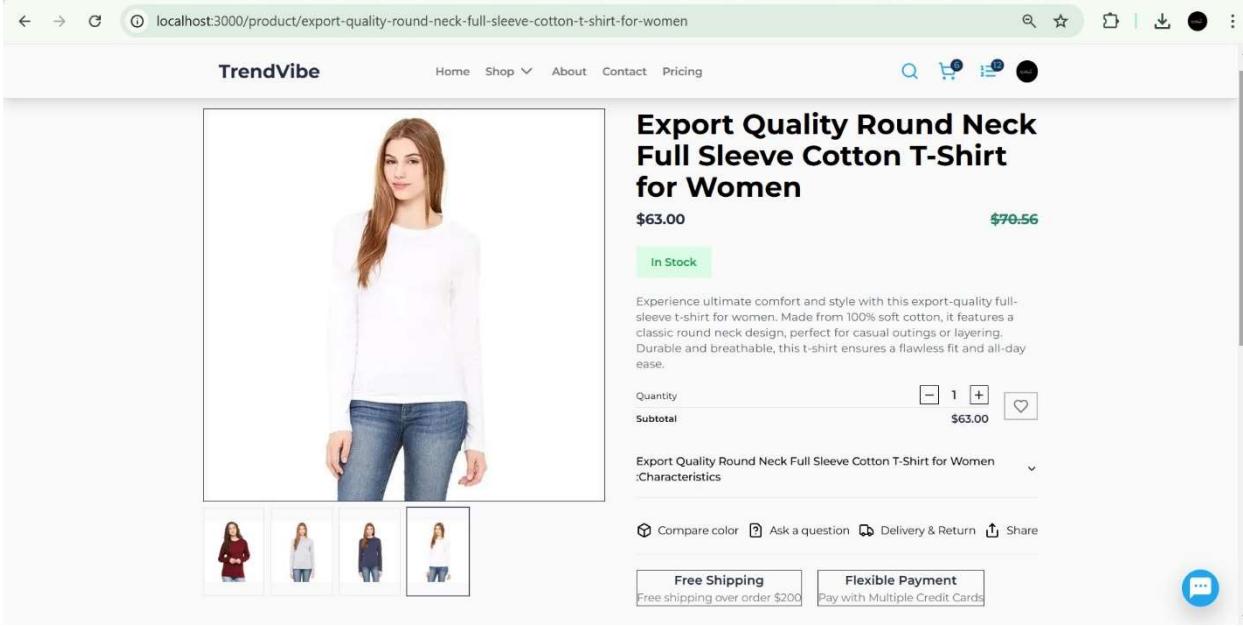
### Features:

- Dynamic routing to individual product pages.
- Displays detailed information such as price, description and other relevant details.

### Code Screenshot:

A screenshot of a code editor (VS Code) showing the file 'page.tsx'. The code is a React component named 'SingleProductPage'. It starts with a function 'const SingleProductPage = async (params: {params: Promise<{slug: string}>}) => {'. The component structure includes a container with images, a title, price, stock status, a description, and several buttons ('Add to Cart', 'Compare', 'Ask a question', 'Delivery & Returns', 'Share'). The code uses Tailwind CSS classes for styling. The code editor interface shows the file path 'src > app > (client) > product > [slug] > page.tsx' and the file name 'final-hackathon-3' in the title bar.

## UI Screenshot:



## 3. Category Component

### Description:

The Category Component displays a list of categories dynamically fetched from the CMS. This allows users to filter products based on category.

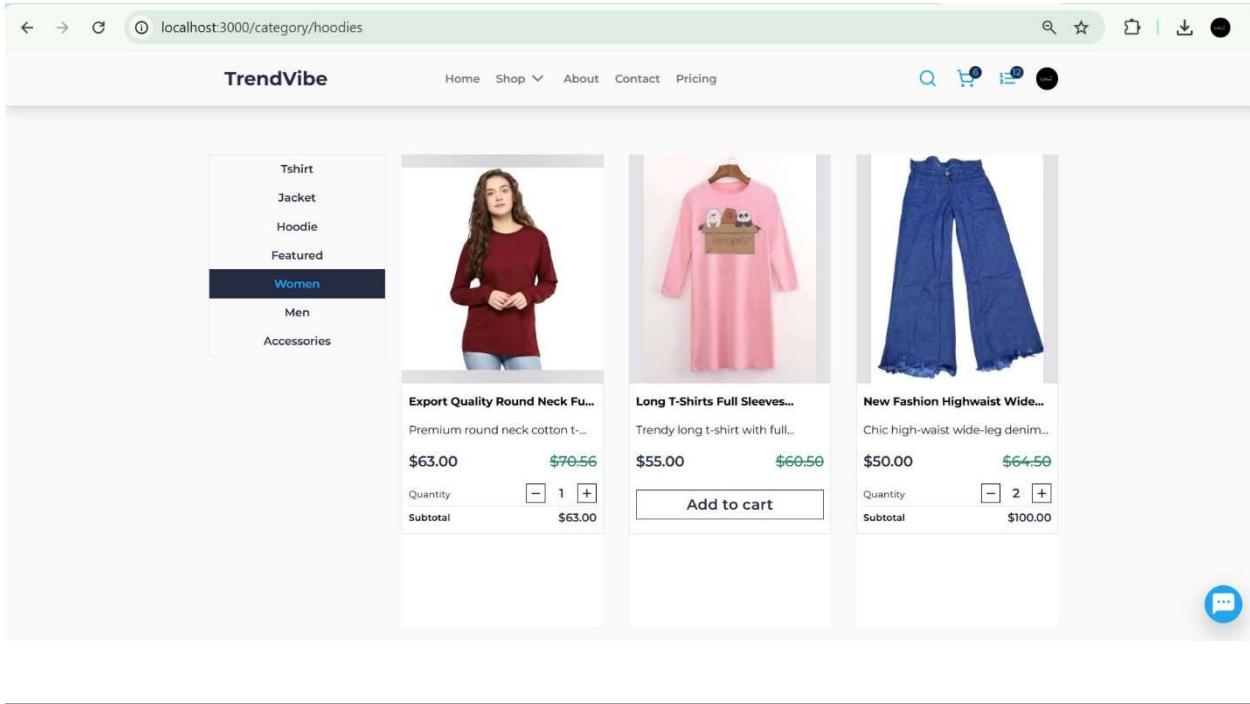
### Features:

- Displays categories.
- Allows filtering of products by category.

### Code Screenshot:

A screenshot of a code editor showing the implementation of the Category Component. The file is named "CategoryProducts.tsx". The code uses functional programming and hooks like useState and useEffect. It fetches products based on the current slug and maps them to a list. It includes loading and error handling logic. The code is well-structured with comments explaining the logic. A status bar at the bottom right says "Don't give up... Keep trying!"

## UI Screenshot:



## 4. Search Bar

### Description:

The Search Bar allows users to search for products by name or tags. It filters the products dynamically as the user types.

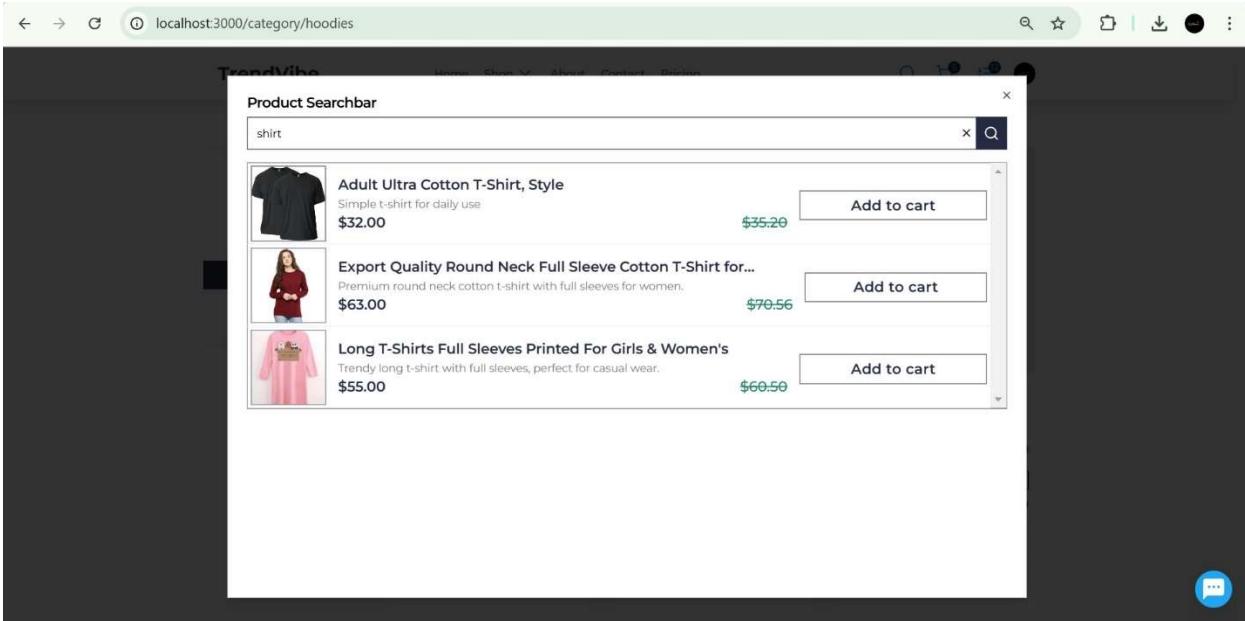
### Features:

- Search functionality for products.
- Filters products by name or tags.

### Code Screenshot:

A screenshot of a code editor showing the implementation of the SearchBar component. The code is written in React using TypeScript. It includes a DialogTrigger component, a DialogContent component with a form containing a search input, and a DialogHeader component with a loading indicator and a search result summary. The code uses various CSS-in-JS classes to style the components.

## UI Screenshot:



## 5. Cart Component

## Description:

The Cart Component displays the products added to the cart, showing details like quantity and total price. Users can adjust the quantity or remove items from the cart.

### **Features:**

- Displays cart items.
  - Allows users to adjust quantities and see the total price.

## Code Screenshot:

```
File Edit Selection View ... < > final-hackathon-3 08 -
```

src/app/client/cart/cart.page.tsx

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { CartService } from 'src/app/shared/cart.service';
import { Product } from 'src/app/shared/product.model';
import { CartPage } from './cart.page';

@Component({
  selector: 'app-cart',
  templateUrl: './cart.page.html',
  styleUrls: ['./cart.page.scss']
})
export class CartPage implements OnInit {
  products: Product[] = [];
  itemCount: number = 0;
  total: number = 0;

  constructor(private cartService: CartService, private router: Router) {}

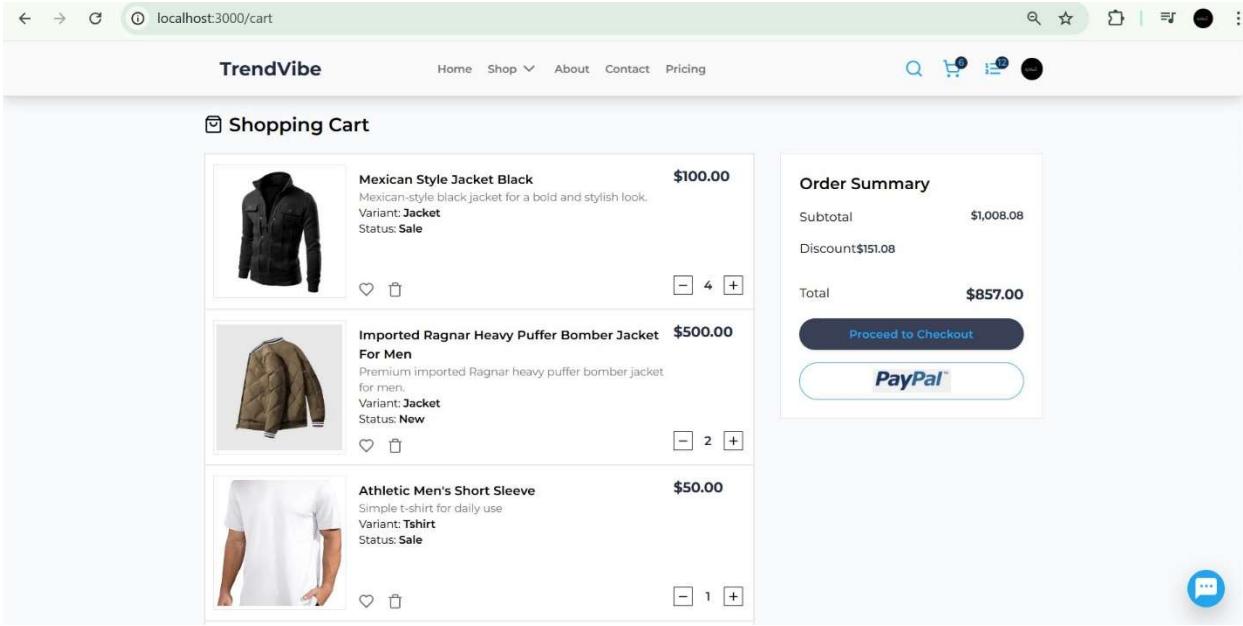
  ngOnInit(): void {
    this.cartService.getCart().subscribe((products) => {
      this.products = products;
      this.itemCount = products.length;
      this.total = products.reduce((acc, curr) => acc + curr.price);
    });
  }

  removeItem(productId: string) {
    this.cartService.removeItem(productId).subscribe();
    this.ngOnInit();
  }

  clearCart() {
    this.cartService.clearCart().subscribe();
    this.ngOnInit();
  }

  handleResetCart() {
    this.cartService.resetCart();
  }
}
```

## UI Screenshot:



## 6. Checkout Flow Component

### Description:

The Checkout Flow Component handles the multi-step checkout process, including fields for billing and payment details.

### Code Screenshot:

```
File Edit Selection View ... ↵ → final-hackathon-3
actions > TS createCheckoutSessions > TS createCheckoutSession > TS sessionPayload > JS line_items > TS items.map() callback > JS price_data > TS product_data
15 }
16 }
17 export async function createCheckoutSession(items:CartItem[],metadata:Metadata){
18   try {
19     const customers = await stripe.customers.list({
20       email:metadata.customerEmail,
21       limit:1
22     })
23     const customerId = customers.data.length > 0 ? customers.data[0].id : ""
24     const sessionPayload:stripe.Checkout.SessionCreateParams={
25       metadata:{
26         orderNumber:metadata.orderNumber,
27         customerName:metadata.customerName,
28         customerEmail:metadata.customerEmail,
29         clerkUserId:metadata.clerkUserId
30       },
31     },
32     mode:"payment",
33     allow_promotion_codes:true,
34     payment_method_types:["card"],
35     invoice_creation:{
36       enabled:true
37     },
38     success_url: "https://e-commerce-trend-vibe.vercel.app/success?session_id:[CHECKOUT_SESSION_ID]&orderNumber:${metadata.orderNumber}",
39     cancel_url: "https://e-commerce-trend-vibe.vercel.app/cancel",
40     line_items:items.map((item) =>{
41       price_data: {
42         currency:"USD",
43         unit_amount: Math.round(item.product.price! * 100),
44         product_data:[
45           {
46             name:item.product.name || "Unnamed Product",
47             description: item.product.description,
48             metadata:[{id:item.product._id},
49             images:item.product.images && item.product.images.length > 0 ? [urlFor(item.product.images[0]).url()] : undefined
50           ],
51           quantity:item.quantity,
52         })
53       }
54     })
55     if(customerId){
56       sessionPayload.customer = customerId
57     }else{
58       sessionPayload.customer_email = metadata.customerEmail
59     }
60     const session = await stripe.checkout.sessions.create(sessionPayload)
61     return session.url
62   }
```

## UI Screenshot:

The screenshot shows a payment interface for TrendVibe. On the left, a summary of items is displayed:

- Mexican Style Jacket Black: \$100.00
- Imported Ragnar Heavy Puffer Bomber Jacket For Men: \$500.00
- Athletic Men's Short Sleeve: \$50.00
- Brown dumbbell pullover warm fleece kangaroo hoodie for men and boys: \$44.00

The total amount is \$857.00. On the right, there is a green button labeled "Pay with link" and a "TEST MODE" indicator. Below it, there are fields for email (amnarafeeq68@gmail.com), card information (1234 1234 1234 1234, VISA logo), cardholder name (Full name on card), country or region (Pakistan), and a checkbox for "Securely save my information for 1-click checkout".

## 7. Header and Footer Components

### Description:

The Header and Footer Components provide consistent navigation and branding across the site.

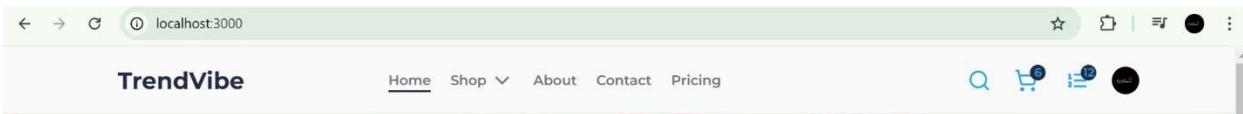
### Features:

- Navigation links to key pages like Home, About, and Contact.
- Branding and responsive design.

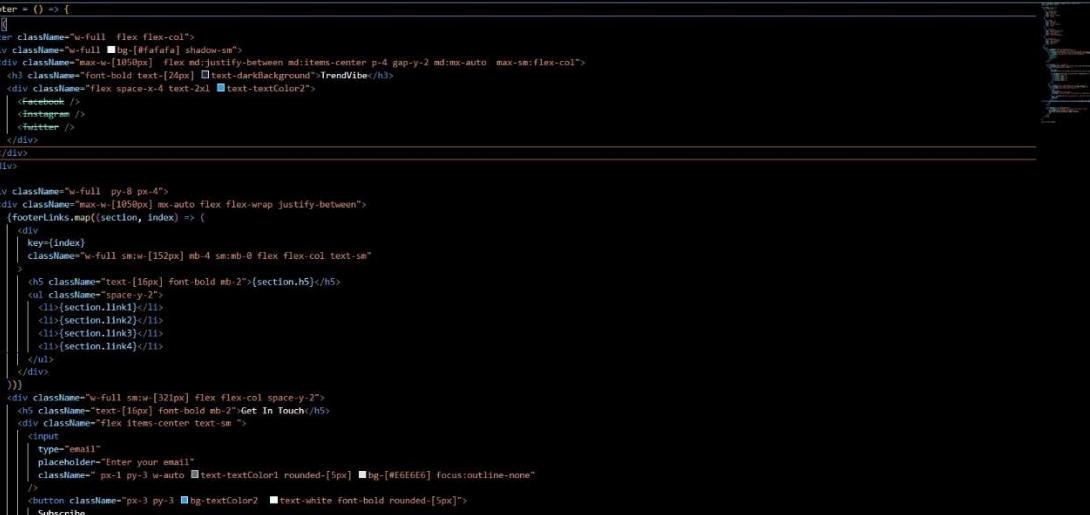
### Header Code Screenshot:

```
File Edit Selection View ... ⏪ ⏩ final-hackathon-3
Header.tsx
src > components > Header > Header.tsx
  import { useHeader } from 'react-use';
  import { getAllCategories, getMyOrders } from '@sanity/helpers/queries';
  const Header = async() => {
    const user = await currentUser();
    const { userId } = await auth();
    const categories = await getAllCategories();
    let orders = null;
    if(userId){
      orders = await getMyOrders(userId);
    }
    return (
      <header className='sticky top-0 z-50 bg-lightBackground shadow-lg border-b border-b-gray-300 py-5'>
        <Container className='flex justify-between items-center gap-7'>
          <div className='flex items-center ml-3 gap-2'>
            <MobileMenu/>
            <h2 className='text-xl sm:text-[24px] font-bold bg-darkBackground trendVibe'>trendVibe</h2>
          </div>
          <HeaderMenu categories={categories}></HeaderMenu>
        </Container>
        <div className='w-auto md:w-1/4 flex items-center justify-end gap-2 sm:gap-5'>
          <SearchBar/>
          <CartIcon/>
        </div>
        <div>
          <ClerkLoaded>
            <SignInLink>
              <Link href="/orders" className='group relative'>
                <ListOrdered className='w-6 sm:w-8 h-6 text-textColor2 group-hover:text-darkBackground hoverEffect' />
                <span className='absolute -top-1 -right-1 bg-darkBackground text-textColor2 h-4 w-4 rounded-full text-sm font-semibold flex items-center justify-center'>{orders?.length > orders.length ? 0}</span>
              </Link>
            </SignInLink>
          </ClerkLoaded>
        </div>
      </Container>
    );
  }
  export default useHeader(Header);
  </Header>
```

## Header UI Screenshot:



## Footer Code Screenshot:



```
Header.tsx Footer.tsx
File Edit Selection View ... ← → final-hackathon-3
src > components > Footer.tsx > Footer
4 const Footer = () => {
5   return [
6     <footer className="w-full flex flex-col">
7       <div className="max-w-[1050px] bg-[#f0f0f0] shadow-sm">
8         <div className="font-bold text-[24px] text-darkBackground">TrendVibe</h3>
9         <div className="flex space-x-4 text-2xl text-textColor2">
10           <Facebook />
11           <Instagram />
12           <Twitter />
13         </div>
14       </div>
15     </div>
16   </div>
17
18   <div className="w-full py-8 px-4">
19     <div className="max-w-[1050px] mx-auto flex flex-wrap justify-between">
20       <FooterLinks.map((section, index) => (
21         <div key={index} className="flex w-full sm:w-[1520px] mb-4 sm:mb-0 flex flex-col text-sm">
22           <h5 className="text-[16px] font-bold mb-2">{section.h5}</h5>
23           <ul className="space-y-2">
24             <li>{section.link1}</li>
25             <li>{section.link2}</li>
26             <li>{section.link3}</li>
27             <li>{section.link4}</li>
28           </ul>
29         </div>
30       )}</div>
31     </div>
32   </div>
33
34   <div className="w-full sm:w-[321px] flex flex-col space-y-2">
35     <h5 className="text-[16px] font-bold mb-2">Get In Touch</h5>
36     <div className="flex items-center text-sm">
37       <input type="email" placeholder="Enter your email" className="px-1 py-3 w-auto text-textColor1 rounded-[5px] bg-[#E6E6E6] focus:outline-none" />
38       <button className="px-3 py-3 bg-textColor2 text-white font-bold rounded-[5px]" type="button">
39         <Subscribe />
40       </button>
41     </div>
42     <p className="text-textColor1 text-[12px]">Lorem ipsum dolor Amet</p>
43   </div>
44 </div>
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
```

## Footer UI Screenshot:

