

HACKATHON - 3

Governor Initiative Artificial Intelligence Course

Day 4

Name: Muhammad Farooq Rehmani
Roll Number - 00142495

Contents

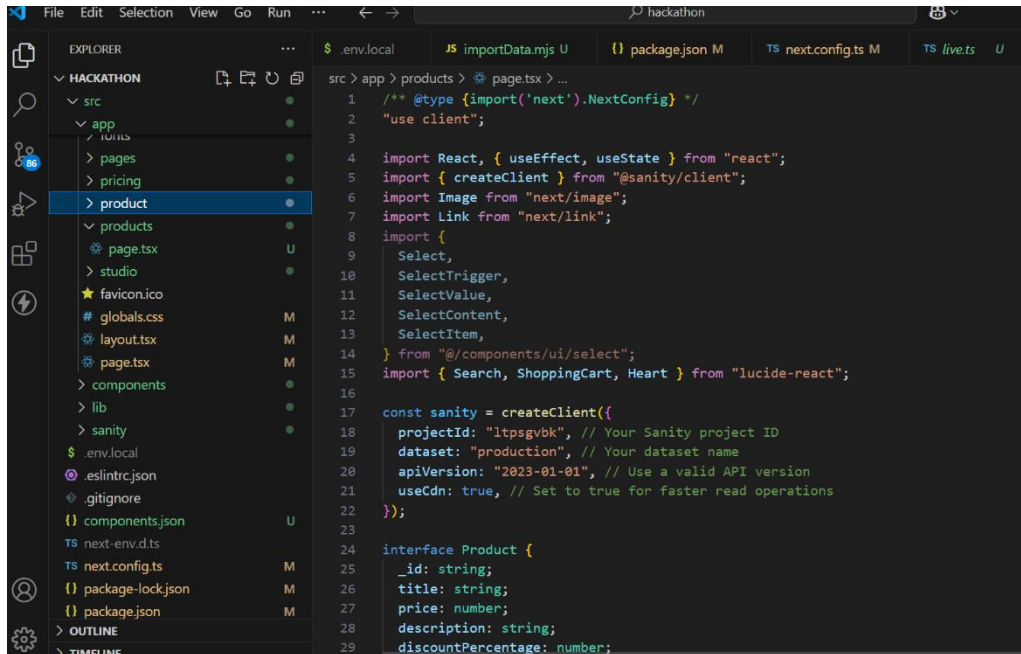
CODING OF API PRODUCT:.....	3
RESULT:.....	4
TAGS:	5
CATAGORIES.....	6
LOGIN CODE	7
NOTIFICATION	8
USER AUTHENTICATION.....	8
PAYENT GATEWAY	8
METHODOLOGY OF WEB FLOW.....	8
WHY THIS WEBSITE IS USER FRIENDLY?	9

DAY 4

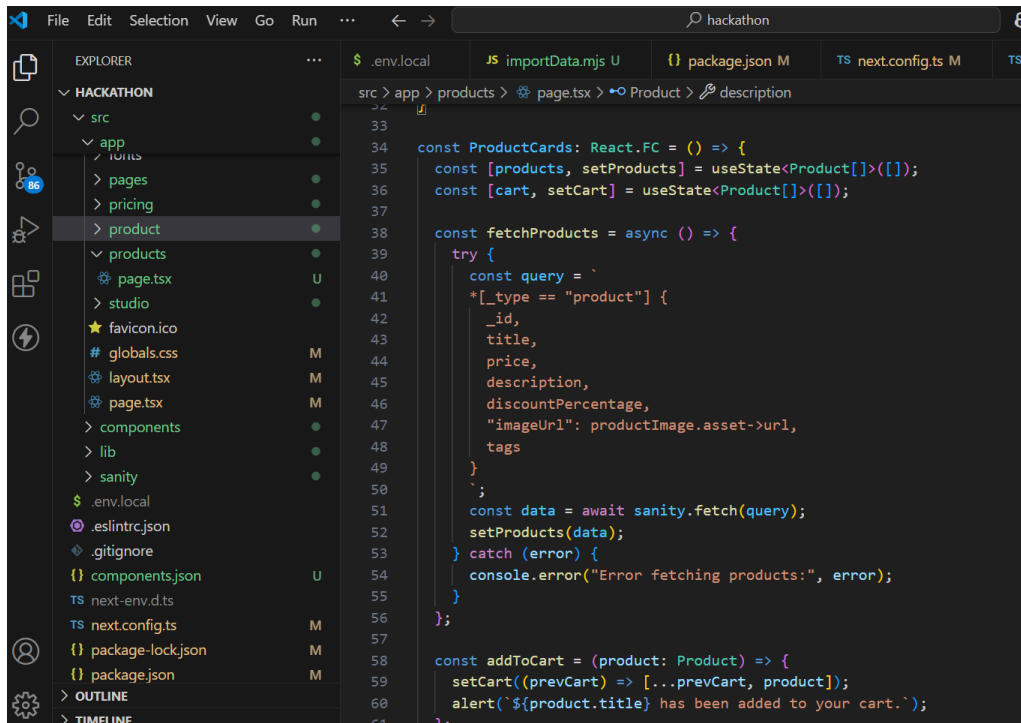
I have developed an **E-Commerce website** using **Template 5**, which represents the "Bandage" brand, catering to **Men, Women, Kids, and Accessories**. The website is designed to provide customers with a seamless and engaging shopping experience. It enables users to easily browse through various categories of stylish outfits and accessories, view detailed product descriptions, and add items to their cart for a smooth checkout process. The website leverages **API fetching through Sanity**, allowing for dynamic updates to product information and images. Although the documentation by **Sir Bilal** required using **Template 6 API**, leading to furniture images being fetched on the product page, this can be adjusted. All pages, including Home, Shop, Product Listings, Blog, and Contact, are fully functional. The accompanying **Word document** showcases images of the website and its features.

This website offers numerous benefits to customers. The intuitive navigation and well-structured pages ensure users can effortlessly find their desired products. The responsive design ensures a consistent experience across devices, while the integration of dynamic APIs keeps the content updated. The use of **Tailwind CSS** enhances the visual appeal and functionality of the website. Tailwind CSS's utility-first approach was applied to achieve a **clean layout, consistent spacing, responsive grids, and customized typography**. Tailwind's **hover effects** add interactivity to buttons and links, while the **predefined classes** streamline the styling of components like cards, navbars, and footers. Overall, the website is both user-friendly and visually engaging, creating a reliable platform for online shopping.

CODING OF API PRODUCT:

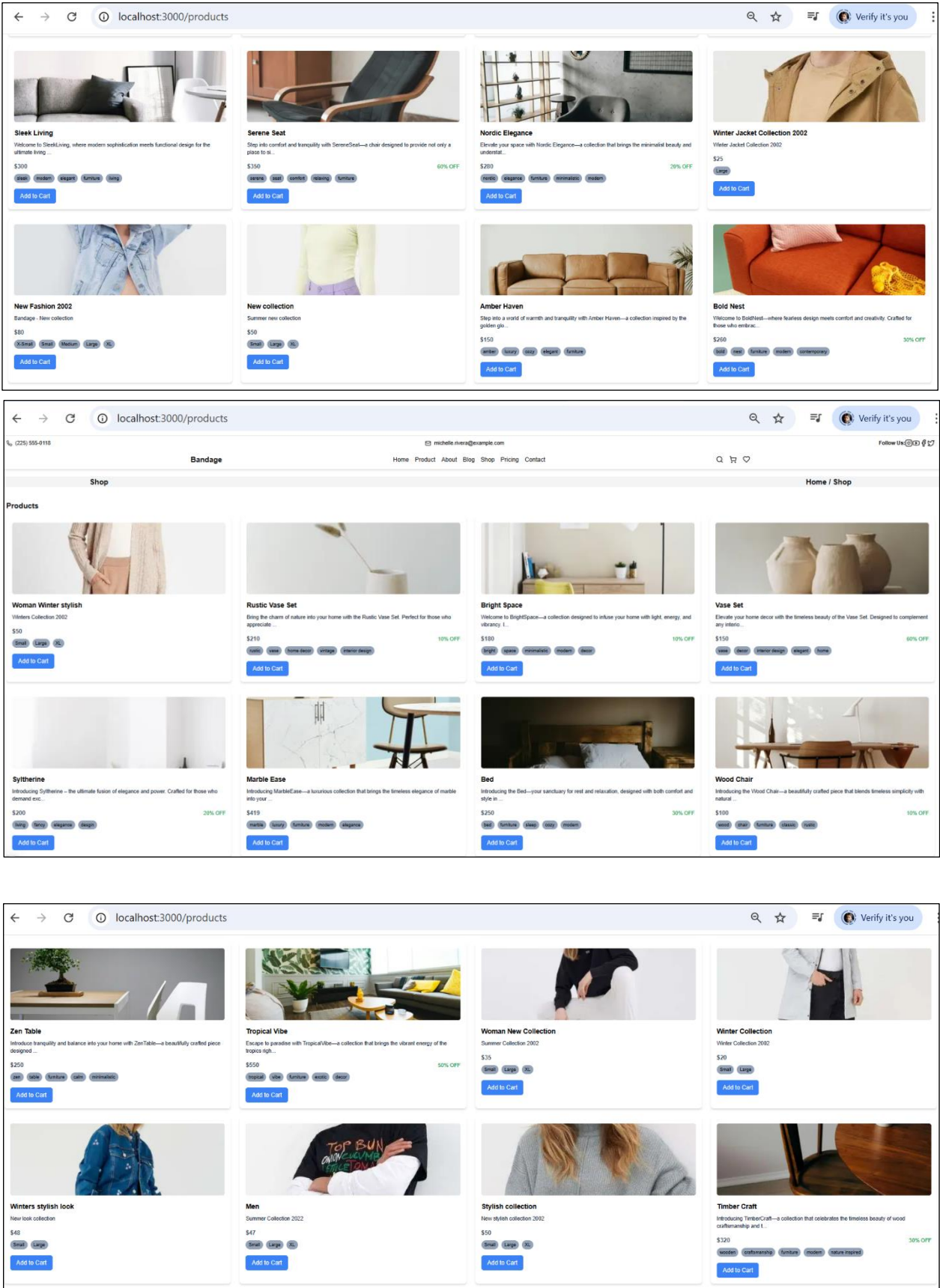


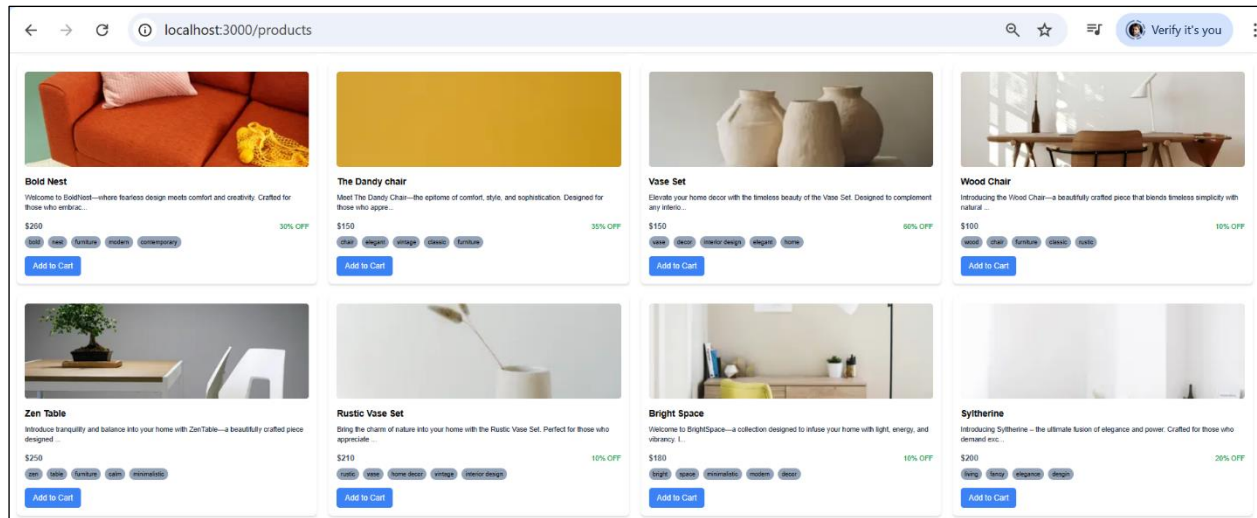
```
1  /** @type {import('next').NextConfig} */
2  "use client";
3
4  import React, { useEffect, useState } from "react";
5  import { createClient } from "@sanity/client";
6  import Image from "next/image";
7  import Link from "next/link";
8  import {
9    Select,
10   SelectTrigger,
11   SelectValue,
12   SelectContent,
13   SelectItem,
14 } from "@components/ui/select";
15 import { Search, ShoppingCart, Heart } from "lucide-react";
16
17 const sanity = createClient({
18   projectId: "ltpsgvbvk", // Your Sanity project ID
19   dataset: "production", // Your dataset name
20   apiVersion: "2023-01-01", // Use a valid API version
21   useCdn: true, // Set to true for faster read operations
22 });
23
24 interface Product {
25   _id: string;
26   title: string;
27   price: number;
28   description: string;
29   discountPercentage: number;
```



```
33
34 const ProductCards: React.FC = () => {
35   const [products, setProducts] = useState<Product[]>([]);
36   const [cart, setCart] = useState<Product[]>([]);
37
38   const fetchProducts = async () => {
39     try {
40       const query = `
41         *[_type == "product"] {
42           _id,
43           title,
44           price,
45           description,
46           discountPercentage,
47           "imageUrl": productImage.asset->url,
48           tags
49         }
50       `;
51       const data = await sanity.fetch(query);
52       setProducts(data);
53     } catch (error) {
54       console.error("Error fetching products:", error);
55     }
56   };
57
58   const addToCart = (product: Product) => {
59     setCart((prevCart) => [...prevCart, product]);
60     alert(`${product.title} has been added to your cart.`);
61   };
62 }
```

RESULT:





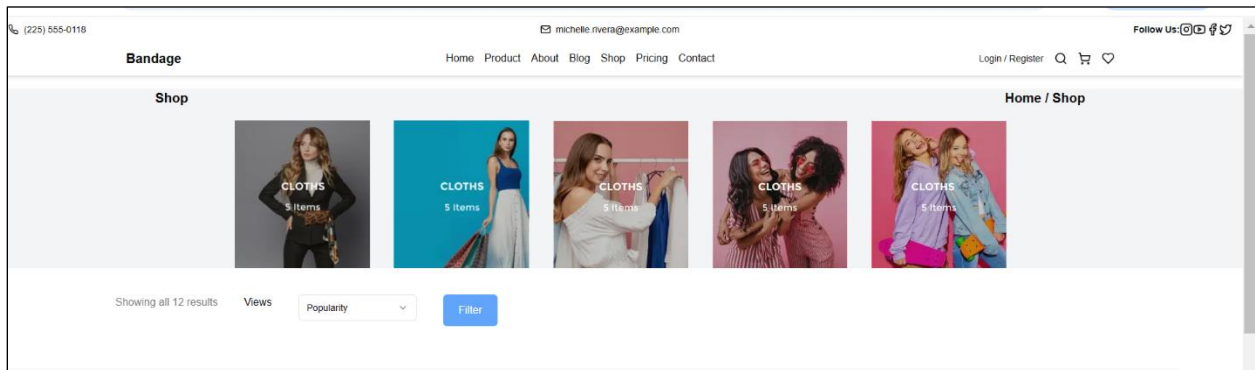
TAGS:

```
src > app > products > page.tsx > ProductCards > products.map() callback > product.tags.map() callback
34  const ProductCards: React.FC = () => {
112    {products.map((product) => (
131      {product.discountPercentage > 0 && (
132        <p className="text-sm text-green-600">{product.discountPercentage}% OFF</p>
133      )}
134    </div>
135    <div className="mt-2 flex flex-wrap gap-2">
136      {product.tags.map((tag, index) => (
137        <span
138          key={index}
139          className="text-xs bg-slate-400 text-black rounded-full px-2 py-1"
140        >
141          {tag}
142        </span>
143      )]}
144    </div>
145    <button
146      onClick={() => addToCart(product)}
147      className="mt-4 bg-blue-500 text-white px-4 py-2 rounded-md hover:bg-blue-600"
148    >
149      Add to Cart
150    </button>
151  </div>
152 </div>
153 ))}
154 </div>
155 </div>
```



CATAGORIES

```
...  page.tsx ...\product U  Footer.tsx U  desktop.tsx  page.tsx ...\desktop U  page.tsx ...\about U  page.tsx ...\
src > app > product > page.tsx > Header
14  export default function Header() {
15
16      /* Product Images */
17      <div className="flex items-center gap-9 justify-center">
18          {["c1", "c2", "c3", "c4", "c5"].map((img) => (
19              <Image key={img} src={`/${img}.png`} alt={img} height={223} width={206} />
20          ))}
21      </div>
22  </div>
23
24      /* Filter and Popularity */
25      <div className="container mx-auto py-10 flex justify-between items-center">
26          <div className="flex gap-10">
27              <h5 style={{ color: "gray", fontFamily: "Arial" }}>Showing all 12 results</h5>
28              <h5>Views</h5>
29              <div>
30                  <Select>
31                      <SelectTrigger className="w-[180px]">
32                          <SelectValue placeholder="Popularity" />
33                      </SelectTrigger>
34                      <SelectContent>
35                          <SelectItem value="popularity">Popularity</SelectItem>
36                          <SelectItem value="price_low">Price: Low to High</SelectItem>
37                          <SelectItem value="price_high">Price: High to Low</SelectItem>
38                      </SelectContent>
39                  </Select>
40              </div>
41              <button className="bg-blue-400 text-white px-6 py-3 rounded-md hover:bg-blue-600">
42                  Filter
43              </button>
44          </div>
45      </div>
46  </div>
```



LOGIN CODE

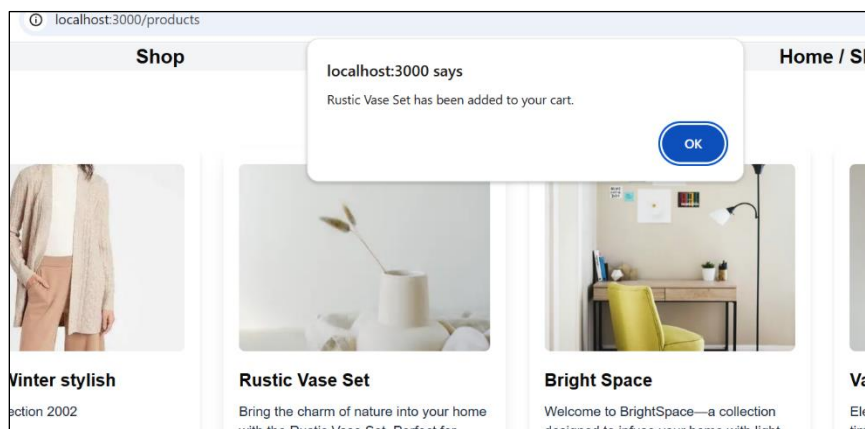
The image shows a code editor on the left and a web browser on the right. The code editor displays the `LoginPage` component in `page.tsx`. The code includes a `handleLogin` function that checks if the password length is less than 6 characters. If so, it sets an error message. Otherwise, it simulates a successful login by logging to the console and pushing the user to the homepage. The component returns a JSX element with a login form and an error message.

```
7  const LoginPage: React.FC = () => {
13    const handleLogin = (e: React.FormEvent) => {
14      // ...
15    }
16  }
17
21  if (password.length < 6) {
22    setError("Password must be at least 6 characters.");
23    return;
24  }
25
26  // Simulated successful login
27  console.log("Login successful:", { email, password });
28  router.push("/"); // Redirect to the homepage
29  };
30
31  return (
32    <div className="flex items-center justify-center min-h-screen bg-gray-100">
33      <div className="bg-white p-8 rounded-lg shadow-md w-full max-w-md">
34        <h2 className="text-2xl font-bold mb-6 text-center">Login</h2>
35        {error && (
36          <div className="bg-red-100 text-red-800 p-2 rounded mb-4 text-sm">
37            {error}
38          </div>
39        )}
40        <form onSubmit={handleLogin}>
41          <div className="mb-4">
42            <label
43              htmlFor="email"
44              className="block text-sm font-medium text-gray-700 mb-1">
```

The web browser shows the login page at `localhost:3000/login`. The page has a header with the site name "Bandage" and navigation links. The main content area features a login form with fields for "Email" and "Password", a "Login" button, and a link to "Sign Up" for users who don't have an account.

NOTIFICATION

```
10 ;
11 const data = await sanity.fetch(query);
12 setProducts(data);
13 } catch (error) {
14   console.error("Error fetching products:", error);
15 }
16 };
17
18 const addToCart = (product: Product) => {
19   setCart((prevCart) => [...prevCart, product]);
20   alert(`${product.title} has been added to your cart.`);
21 };
22
23 const truncateDescription = (description: string): string => {
24   return description.length > 100 ? `${description.slice(0, 100)}...` : description;
25 };
26
27 useEffect(() => {
28   fetchProducts();
29 }, []);
```



USER AUTHENTICATION

We will learn it after hackathon

PAYMENT GATEWAY

We will use our payment gateway to Bank transfer

METHODOLOGY OF WEB FLOW

The customer journey on the website begins with landing on the **homepage**, where they can explore various categories such as Men, Women, Kids, and Accessories. Upon finding their desired product, they can click on it to view detailed information, including price, description, and discounts. The customer can then add the product to their **cart**, where they can review their selections. Once satisfied, they proceed to the **checkout page**, which integrates a secure **payment gateway**. This gateway directs them to enter their **bank details** to complete the payment transaction. After successful payment, the customer receives an order confirmation, and the system initiates the **shipment process**, providing tracking details for their convenience. This seamless flow ensures a user-friendly shopping experience, from browsing to delivery, making it efficient and secure for customers to shop online.

WHY THIS WEBSITE IS USER FRIENDLY?

This website is user-friendly due to its intuitive design and streamlined navigation, ensuring that customers can easily find and purchase products. The homepage is well-organized, with clear categories such as Men, Women, Kids, and Accessories, allowing customers to quickly navigate to their desired section. The **search functionality** helps users find specific products, and the **filter options** enable them to refine their choices based on preferences like size, color, and price. The **product pages** provide detailed information, including images, descriptions, and prices, giving customers all the information they need to make informed purchasing decisions.

The checkout process is straightforward, with a simple **cart system** and **secure payment gateway** that guides users step-by-step from adding items to their cart, entering payment details, and confirming their order. Additionally, the use of **Tailwind CSS** ensures fast loading times, a mobile-friendly design, and a visually appealing, responsive interface that adapts to all screen sizes. All these features contribute to a smooth, hassle-free shopping experience for users.