Day 4

IMPLEMENTATION REPORT {TRENDIFY}

INTRODUCTION:

Welcome to Trendify – your ultimate destination for the latest fashion trends! At Trendify, we bring you a wide range of stylish and trendy clothing for men, women, and kids. Whether you're looking for casual wear, formal outfits, or statement pieces, Trendify has something for everyone. Our curated collection features high-quality fabrics, unique designs, and the perfect fit to keep you looking and feeling your best. Shop with us to stay ahead in the fashion game and elevate your wardrobe with the latest styles!

- **Planning and Designing**:
  - First, I understood the requirements of the website and during the design phase, I created wireframes and mockups. This helped identify key components like product cards, filters, and the checkout system.
- **Building Reusable Components**:
  - I made the components modular so that they could be reused across multiple pages. For instance, the `ProductCard` was created to display details of each product, and the `CategoryFilter` was implemented for filtering different categories.
  - Data was passed through props to make the components flexible and maintainable.
- **State Management**:
  - I used React's `useState` and `useContext` hooks to manage data between components. `useState` was used for local state management, while `useContext` was implemented for global state, like cart items or user authentication state.
- **Styling the Components**:
  - Tailwind CSS was used to style the components and make them responsive. By utilizing Tailwind's utility classes, I quickly styled the components and ensured the website was mobile-friendly and desktop compatible.

- **Integrating with Backend**:
  - I integrated backend services for handling product data, customer details, and order management systems. This integration was done through REST APIs or GraphQL endpoints to load data dynamically.
- **Performance Optimization**:
  - I implemented lazy loading for images, which helped improve the website's performance by loading images only when needed. To efficiently load large datasets, I used pagination and infinite scrolling.
- **Testing and Debugging**:
  - I thoroughly tested the components to ensure everything was working as expected. Bugs and errors were resolved, and the final version was deployed.

Challenges Faced and Solutions Implemented:

1. **Data Integration and Management**:
   - **Challenge**: Integrating dynamic data from the backend and ensuring smooth communication between the frontend and backend was challenging, especially when dealing with large datasets.
   - **Solution**: I implemented REST APIs and GraphQL to fetch data efficiently. For large datasets, I used pagination and lazy loading techniques to improve performance and prevent page loading delays.
2. **State Management**:
   - **Challenge**: Managing global and local state across multiple components was complex, especially when the application grew and the data flow became more complicated.
   - **Solution**: I used React's `useState` for local state management and `useContext` for global state management. This allowed me to maintain a clean and manageable flow of data, while ensuring components remained decoupled.
3. **Responsive Design Issues**:
   - **Challenge**: Ensuring the website looked good on all screen sizes, especially with different device orientations and screen resolutions, was tricky.
   - **Solution**: I used Tailwind CSS to create responsive layouts with ease. Its utility classes helped in adjusting the layout at different breakpoints, ensuring the website was mobile-friendly and desktop compatible.
4. **Performance Optimization**:
   - **Challenge**: Optimizing the website for fast load times, especially when handling a large number of images and products, posed performance challenges.
   - **Solution**: I implemented lazy loading for images, ensuring they only load when required. Additionally, I used infinite scrolling for product lists, which allowed me to load content as the user scrolls, improving both performance and user experience.

5. **Cross-Browser Compatibility**:
   - **Challenge**: Ensuring the website performed consistently across different browsers was another hurdle, as some styles and functionalities didn’t work as expected on all platforms.
   - **Solution**: I used browser testing tools and ensured CSS compatibility with vendor prefixes. Tailwind CSS helped to mitigate many cross-browser issues by providing a consistent design system.
6. **Error Handling and Debugging**:
   - **Challenge**: Debugging complex issues in state management and API integrations took considerable time, especially when errors weren’t easy to track.
   - **Solution**: I used debugging tools like React Developer Tools and console logging to trace the issue. Additionally, I incorporated error boundaries to catch errors gracefully in production.

Best Practices Followed During Development:

1. **Modular and Reusable Code**:
   - I followed the best practice of writing modular and reusable components. This helps reduce code duplication and makes it easier to maintain and update the project over time. Components like `ProductCard` and `CategoryFilter` were created to be reusable across different pages.
2. **State Management with React Hooks**:
   - I used React’s `useState` and `useContext` hooks for managing state efficiently. Local state was managed within components using `useState`, while global state, such as user authentication and cart data, was handled using `useContext`. This kept the application’s data flow clean and manageable.
3. **Responsive and Mobile-First Design**:
   - I adopted a mobile-first approach while designing the website. Using Tailwind CSS, I made sure that the layout is responsive and works seamlessly across all screen sizes. This ensured a smooth user experience on both mobile devices and desktop.
4. **Performance Optimization**:
   - To enhance the performance of the website, I implemented techniques like lazy loading for images and pagination/infinite scrolling for large datasets. This helped the site load faster and provided a better user experience.
5. **Error Handling and Debugging**:
   - I followed proper error handling practices by using error boundaries and adding necessary checks for potential errors. For debugging, I used tools like React Developer Tools and console logging, ensuring the application runs smoothly without crashing.
6. **Version Control with Git**:
   - I used Git for version control, which helped track changes, collaborate with team members, and easily revert back to previous versions if needed. This ensured that the development process remained organized and transparent.
7. **Code Consistency and Formatting**:

   - I followed a consistent code style and used linters like ESLint to maintain clean, readable, and error-free code. Prettier was also used for automatic code formatting to ensure uniformity.

8. **Security Best Practices**:
   - For user authentication and data security, I made sure to hash passwords before storing them in the database. Additionally, I ensured that all sensitive information, like API keys, was stored securely using environment variables.
9. **Testing and Quality Assurance**:
   - I performed thorough testing to ensure the application worked as expected. Unit tests and integration tests were used to verify the functionality of components and API calls, ensuring the app remained stable as new features were added.

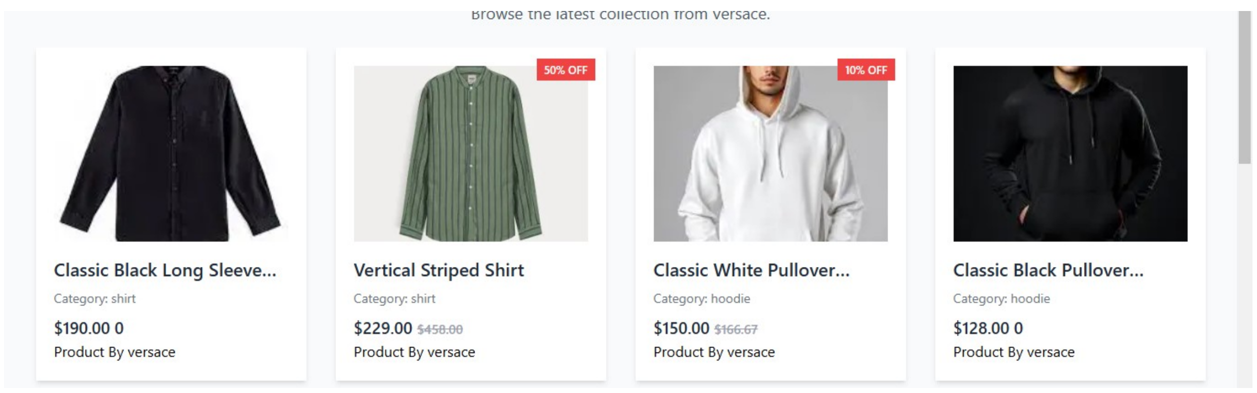
User Authentication:

We will learn it after ka hackathon.

Payment Gateway:

- **Stripe**
- **PayPal**
- **JazzCash**
- **Online Bank Transfer**

Conclusion: Trendify is more than just an eCommerce platform; it is a one-stop destination for fashion enthusiasts. With a diverse range of high-quality clothing for men, women, and children, Trendify aims to bring the latest trends directly to your doorstep. The seamless shopping experience, user-friendly interface, and secure payment methods ensure that customers enjoy a hassle-free and satisfying journey. By focusing on performance optimization, responsive design, and modern technologies, Trendify promises to remain at the forefront of fashion and technology, offering stylish clothing that meets the needs of today's fashion-forward individuals. Join us at Trendify, and elevate your wardrobe with the latest and best in fashion!



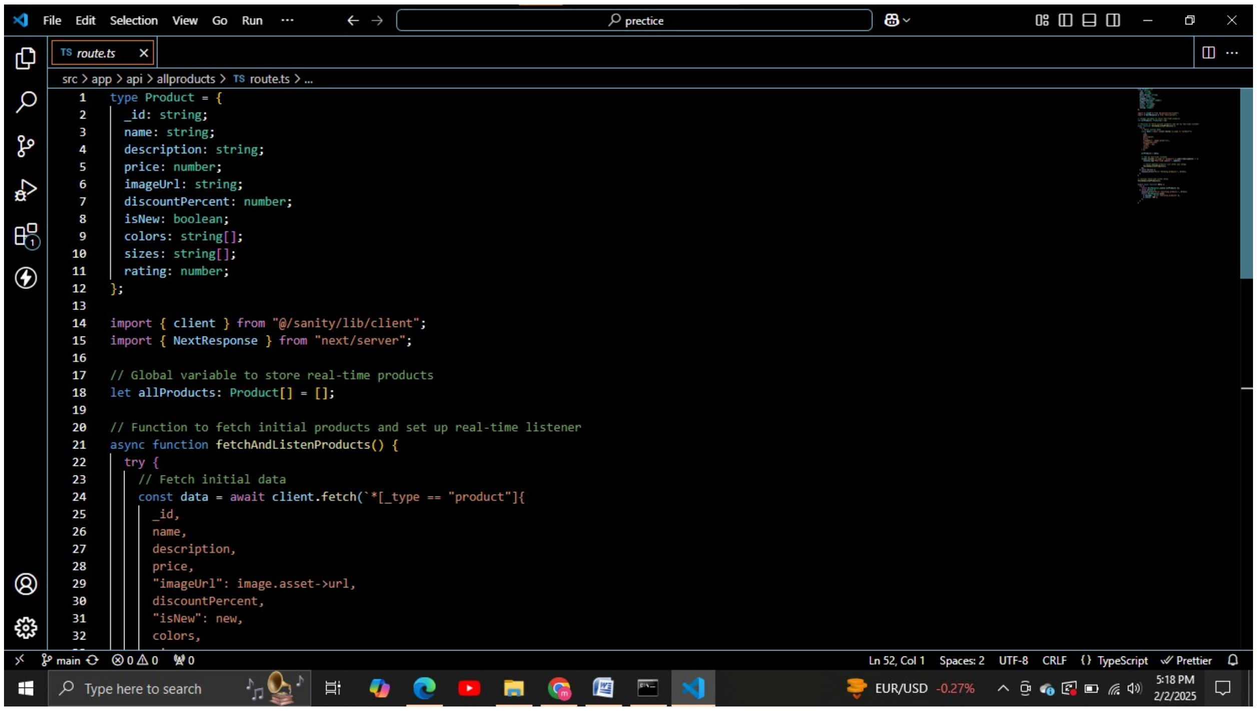
Browse the latest collection from versace.

Classic Black Long Sleeve...
Category: shirt
$190.00 0
Product By versace

50% OFF
Vertical Striped Shirt
Category: shirt
$229.00 ~~$458.00~~
Product By versace

10% OFF
Classic White Pullover...
Category: hoodie
$150.00 ~~$166.67~~
Product By versace

Classic Black Pullover...
Category: hoodie
$128.00 0
Product By versace



```
type Product = {
  _id: string;
  name: string;
  description: string;
  price: number;
  imageUrl: string;
  discountPercent: number;
  isNew: boolean;
  colors: string[];
  sizes: string[];
  rating: number;
};

import { client } from "@/sanity/lib/client";
import { NextResponse } from "next/server";

// Global variable to store real-time products
let allProducts: Product[] = [];

// Function to fetch initial products and set up real-time listener
async function fetchAndListenProducts() {
  try {
    // Fetch initial data
    const data = await client.fetch(`*[_type == "product"]{
      _id,
      name,
      description,
      price,
      "imageUrl": image.asset->url,
      discountPercent,
      "isNew": new,
      colors,
```

Individual Product Details

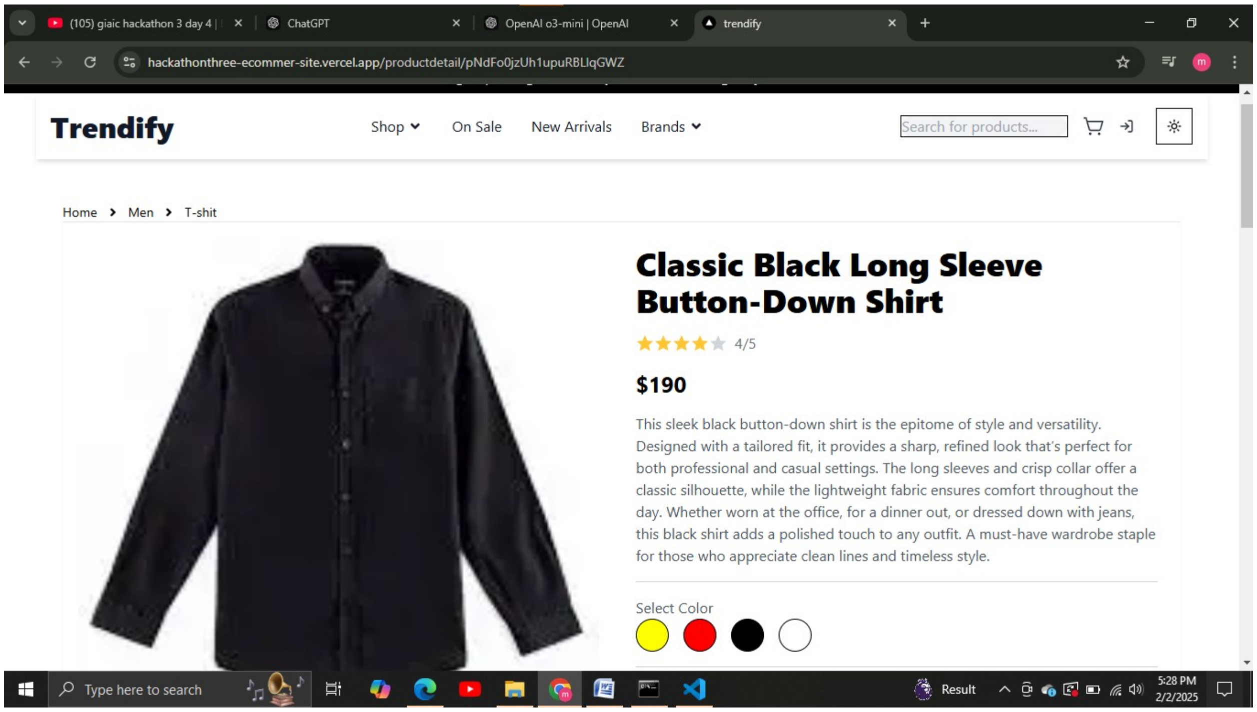
CODE:

```
"use client";
import React, { useState, useEffect } from "react";
import { Card, CardContent } from "@/components/ui/card";
import { LiaStarSolid } from "react-icons/lia";
import Image from "next/image";
import { notFound } from "next/navigation";
import { ButtonDemo } from "../button";
import { Counter } from "../counter";
import Reviews from "../reviwes";
import YouMightAlsoLike from "../youmightlikealso";
import { BreadcrumbWithCustomSeparator } from "../breadcrumbs";
import { useRouter } from "next/navigation";

interface Product {
  _id: number;
  name: string;
  description: string;
  price: number;
  imageUrl: string;
  category: string;
  discountPercent: number;
  isNew: boolean;
  colors: string[];
  sizes: string[];
  rating: number;
  original_price?: number;
}

interface CartItem {
  id: number;
  name: string;
  price: number;
  discountPercent: number;
  quantity: number;
  image: string;
  color: string;
  size: string;
}

const getProductData = async (id: string): Promise<Product | null> => {
  const res = await fetch(
    `${process.env.NEXT_PUBLIC_SITE_URL}api/allproducts`,
    {
      cache: "no-cache",
    },
  );
  if (!res.ok) {
    return null;
  }
  const data = await res.json();
  return (
    data.allProducts.find(
      (product: Product) => product._id.toString() === id,
    ) || null
  );
};

const ProductDetailPage = ({ params }: { params: { id: string } }) => {
  const [product, setProduct] = useState<Product | null>(null);
  const [count, setCount] = useState<number>(1);
  const [cartItems, setCartItems] = useState<CartItem[]>([]);
  const [selectedColor, setSelectedColor] = useState<string | null>(null);
  const [selectedSize, setSelectedSize] = useState<string | null>(null);
  const router = useRouter();

  useEffect(() => {
    const fetchData = async () => {
      const productData = await getProductData(params.id);
      if (!productData) {
        notFound();
      } else {
        setProduct({
          ...productData,
          // Assigning a random rating (between 1 and 5) to the product
          rating: Math.floor(Math.random() * 5) + 1,
        });
      }
    };

    fetchData();
  }, [params.id]);

  useEffect(() => {
    if (typeof window !== "undefined") {
      const storedCart = localStorage.getItem("cart");
      if (storedCart) {
        setCartItems(JSON.parse(storedCart));
      }
    }
  }, []);

  if (!product) {
    return <div>Loading...</div>;
  }

  const handleAddToCart = async () => {
    if (!selectedColor || !selectedSize) {
      alert("Please select both color and size.");
      return;
    }

    const updatedCart = [...cartItems];
    const existingProductIndex = updatedCart.findIndex(
      (item) =>
        item.id === product._id &&
        item.color === selectedColor &&
        item.size === selectedSize,
    );

    if (existingProductIndex !== -1) {
      updatedCart[existingProductIndex].quantity += count;
    } else {
      const newItem: CartItem = {
        id: product._id,
        name: product.name,
        price: product.price,
        discountPercent: product.discountPercent,
        quantity: count,
        image: product.imageUrl,
        color: selectedColor,
        size: selectedSize,
      };
      updatedCart.push(newItem);
    }

    // Saving updated cart to localStorage
    localStorage.setItem("cart", JSON.stringify(updatedCart));
    setCartItems(updatedCart);

    alert(`${count} item${count > 1 ? "s" : ""} added to your cart`);

    try {
      console.log("Sending to backend:", updatedCart);
      const response = await fetch(
        `${process.env.NEXT_PUBLIC_SITE_URL}api/cart`,
        {
          method: "POST",
          headers: {
            "Content-Type": "application/json",
          },
          body: JSON.stringify({
            id: product._id,
            quantity: count,
            name: product.name,
            price: product.price,
            discountPercent: product.discountPercent,
            image: product.imageUrl,
            color: selectedColor,
            size: selectedSize,
          }),
        },
      );
      if (!response.ok) {
        throw new Error("Failed to update cart in API");
      }

      const data = await response.json();
      console.log("API Response:", data);
      router.push(`${process.env.NEXT_PUBLIC_SITE_URL}/cart`);
    } catch (error) {
      console.error("Error adding to cart:", error);
      alert("Failed to add to cart. Please try again.");
    }
  };

  return (
    <div>
      <div className="overflow-x-hidden">
        <div className="mt-12 px-4 sm:px-8 lg:px-16 h-auto">
          <BreadcrumbWithCustomSeparator />

          <Card className="bg-white rounded-lg p-4 border-l-white border-r-
white">
            <div className="flex flex-col sm:flex-row gap-6 sm:gap-8">
              <div className="flex flex-col sm:flex-row-reverse gap-4 w-full
sm:w-1/2">
                <div className="relative w-full h-[300px] sm:h-[400px] lg:h-
[515px]">
                  <Image
                    src={product.imageUrl}
                    alt={product.name}
                    className="rounded-lg"
                    layout="fill"
                    objectFit="cover"
                  />
                </div>
              </div>

              <div className="w-full sm:w-1/2">
                <CardContent className="p-2">
                  <div className="text-start">
                    <h1 className="text-2xl sm:text-4xl font-extrabold text-black
mb-4">
                      {product.name}
                    </h1>
                    <div className="flex justify-start items-center mb-4">
```

```
                      {Array.from({ length: 5 }).map((_, index) => (
                        <LiaStarSolid
                          key={index}
                          className={`w-4 h-4 sm:w-5 sm:h-5 ${index <
product.rating ? "text-[#FFC633]" : "text-gray-300"}`}
                        />
                      ))}
                      <span className="ml-2 text-sm sm:text-base text-gray-500">
                        {product.rating}/5
                      </span>
                    </div>
                    <div className="flex items-center gap-4 mb-4">
                      <span className="text-xl sm:text-2xl font-bold text-black">
                        ${product.price}
                      </span>
                      {product.original_price && product.original_price > 0 && (
                        <span className="text-xl sm:text-2xl font-bold text-gray-
400 line-through">
                          ${product.original_price}
                        </span>
                      )}
                      {product.discountPercent > 0 && (
                        <span className="text-[#FF3333] font-medium text-xs py-
[2px] px-[8px] bg-[#FF33331A] rounded-full">
                          {product.discountPercent}%
                        </span>
                      )}
                    </div>
                    <p className="text-sm sm:text-base font-normal text-gray-600
mb-4">
                      {product.description ||
                        "This is a placeholder description."}
                    </p>
                    <div className="border-t border-gray-300 my-4"></div>
                    <div className="mb-4">
                      <span className="text-sm sm:text-base font-normal text-
gray-600 mb-2">
                        Select Color
                      </span>
                      <div className="flex sm:gap-4">
                        {product.colors.map((color, index) => (
                          <div
                            key={index}
                            className={`w-8 h-8 sm:w-9 sm:h-9 rounded-full
cursor-pointer border border-black border-1 ${selectedColor === color ? "ring-2
ring-blue-500" : ""}`}
                            style={{ backgroundColor: color }}
                            onClick={() => setSelectedColor(color)}
                          />
                        ))}
                      </div>
                    </div>
                    <div className="border-t border-gray-300 my-4"></div>
                    <div className="flex flex-col gap-4 mb-4">
                      <span className="text-sm sm:text-base font-normal text-
gray-600">
                        Choose Size
                      </span>
                      <div className="flex gap-2 sm:gap-4">
                        {product.sizes.map((size, index) => (
                          <button
                            key={index}
                            className={`px-3 py-2 sm:px-4 sm:py-2 rounded-full
text-sm sm:text-base font-medium ${selectedSize === size ? "bg-black text-white"
: "bg-gray-200 text-gray-600"}`}
                            onClick={() => setSelectedSize(size)}>
                            {size}
                          </button>
                        ))}
                      </div>
                    </div>
                    <div className="border-t border-gray-300 my-4"></div>
                    <div className="flex justify-between items-center mt-4">
                      <Counter onChange={(value) => setCount(value)} />
                      <ButtonDemo
                        label="Add to Cart"
                        onClick={handleAddToCart}
                      />
                    </div>
                    <div className="border-t border-gray-200 my-4"></div>
                  </div>
                </CardContent>
              </div>
            </div>
          </Card>
        </div>
        <Reviews />
        <YouMightAlsoLike />
```

```
      </div>
    </div>
  );
};

export default ProductDetailPage;
```

PRODUCT:1



Home > Men > T-shit

Classic Black Long Sleeve Button-Down Shirt

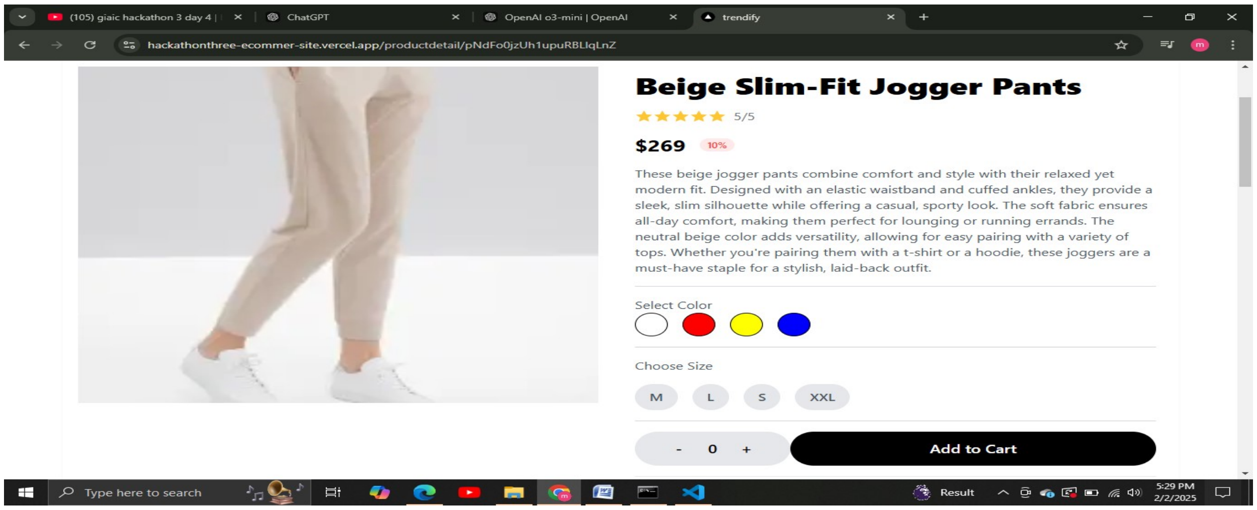
4/5

$190

This sleek black button-down shirt is the epitome of style and versatility. Designed with a tailored fit, it provides a sharp, refined look that's perfect for both professional and casual settings. The long sleeves and crisp collar offer a classic silhouette, while the lightweight fabric ensures comfort throughout the day. Whether worn at the office, for a dinner out, or dressed down with jeans, this black shirt adds a polished touch to any outfit. A must-have wardrobe staple for those who appreciate clean lines and timeless style.

Select Color

PRODUCT:2


(105) giaic hackathon 3 day 4 | ChatGPT | OpenAI o3-mini | OpenAI | trendify

hackathonthree-ecommer-site.vercel.app/productdetail/pNdFo0jzUh1upuRBLlqLnZ

Beige Slim-Fit Jogger Pants

★★★★★ 5/5

$269 10%

These beige jogger pants combine comfort and style with their relaxed yet modern fit. Designed with an elastic waistband and cuffed ankles, they provide a sleek, slim silhouette while offering a casual, sporty look. The soft fabric ensures all-day comfort, making them perfect for lounging or running errands. The neutral beige color adds versatility, allowing for easy pairing with a variety of tops. Whether you're pairing them with a t-shirt or a hoodie, these joggers are a must-have staple for a stylish, laid-back outfit.

Select Color

Choose Size

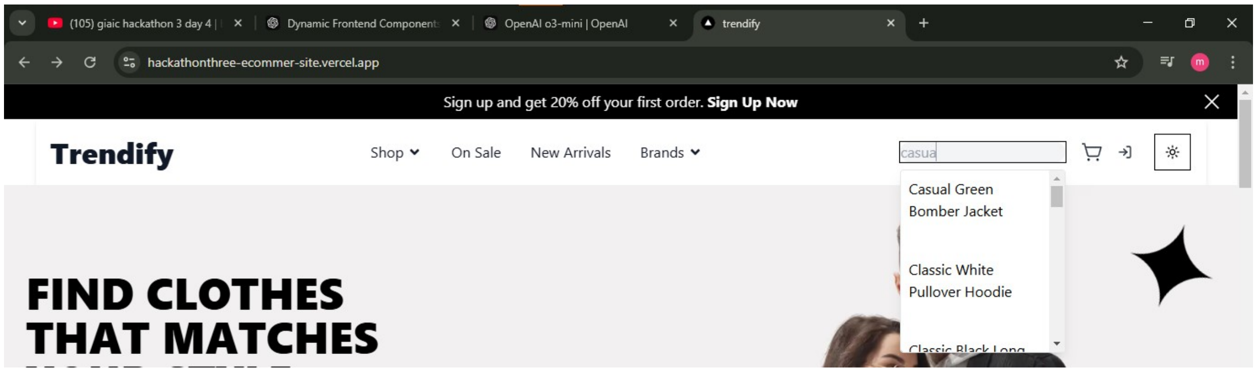
M L S XXL

- 0 + Add to Cart

Type here to search Result 5:29 PM 2/2/2025

SearchBar:

I have created a search bar using a **ShadCN UI** component, which allows users to search for products available on my website by their name. When a user clicks on a product name, they are redirected to the product's details page.


(105) giaic hackathon 3 day 4 | Dynamic Frontend Components | OpenAI o3-mini | OpenAI | trendify

hackathonthree-ecommer-site.vercel.app

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

Trendify Shop On Sale New Arrivals Brands

casua

Casual Green Bomber Jacket

Classic White Pullover Hoodie

Classic Black Long

FIND CLOTHES THAT MATCHES

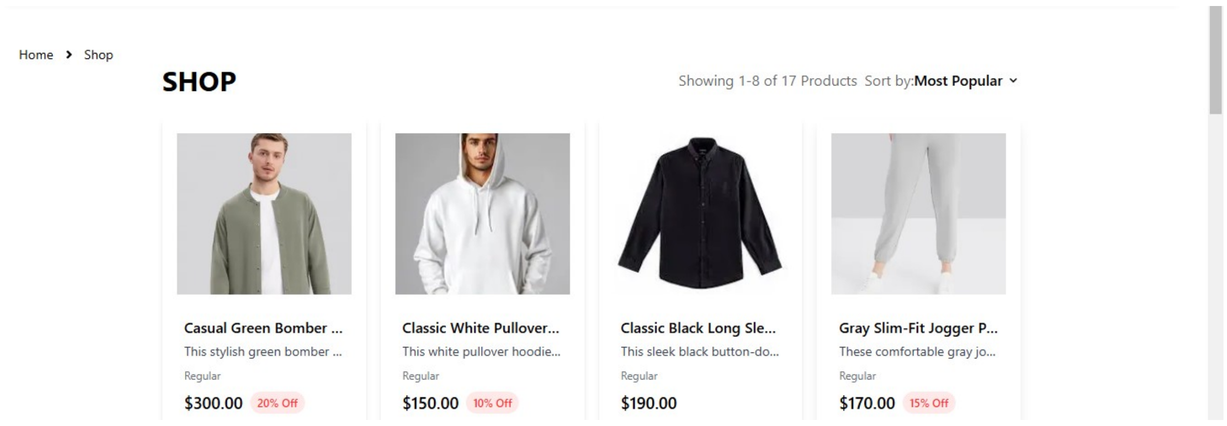
```
"use client";

import * as React from "react";
import { Check } from "lucide-react";
import { useRouter } from "next/navigation";

// Define the type for search result
```

```
type SearchResult = {
  _id: string;
  name: string;
};

export function GlobalSearchBar() {
  const [value, setValue] = React.useState("");
  const [searchResults, setSearchResults] = React.useState<SearchResult[]>([]);
// Use the defined type
  const router = useRouter();

  // Fetch results when value changes
  React.useEffect(() => {
    if (value.trim()) {
      const fetchResults = async () => {
        const response = await fetch(
          `/api/search?q=${value}`,
        );
        const data = await response.json();
        setSearchResults(data);
      };
      fetchResults();
    } else {
      setSearchResults([]);
    }
  }, [value]);

  return (
    <div className="relative flex items-center w-full gap-3 bg-gray-100 rounded-
full">
      <input
        type="text"
        placeholder="Search for products..."
        value={value}
        onChange={(e: React.ChangeEvent<HTMLInputElement>) =>
          setValue(e.target.value)
        }
        className="border-none bg-transparent text-gray-400 font-satoshi text-
[16px] leading-[22px] outline-none w-full"
      />
      {value && searchResults.length > 0 && (
        <div className="absolute top-full mt-2 w-full bg-white shadow-lg border
rounded z-10 max-h-[200px] overflow-y-auto">
          {searchResults.map((result, index) => (
            <div
```

```
              key={result._id}
              onClick={() => {
                router.push(`/productdetail/${result._id}`); // Navigate to
product page
              }}
              className={`p-2 hover:bg-gray-100 cursor-pointer ${index < 3 ? "" :
"opacity-60"}`}>
              {result.name}
              <Check className="ml-auto opacity-0" />
            </div>
          ))}
        </div>
      )}
      {value && searchResults.length === 0 && (
        <div className="absolute top-full mt-2 w-full bg-white shadow-lg border
rounded z-10">
          <div className="p-2">No results found.</div>
        </div>
      )}
    </div>
  );
}
```

Pagination:

My website has a total of **17 products**, which will be displayed when clicking on the **"Shop"** option in the navbar. However, only **8 products** will be shown per page. Using **pagination**, the next **8 products** will appear on the second page, and the last **1 product** will be on the third page. The pagination on my website is working correctly wherever it has been implemented.


Home > Shop

SHOP

Showing 1-8 of 17 Products Sort by:**Most Popular**

Casual Green Bomber ...
This stylish green bomber ...
Regular
$300.00 20% Off

Classic White Pullover...
This white pullover hoodie...
Regular
$150.00 10% Off

Classic Black Long Sle...
This sleek black button-do...
Regular
$190.00

Gray Slim-Fit Jogger P...
These comfortable gray jo...
Regular
$170.00 15% Off

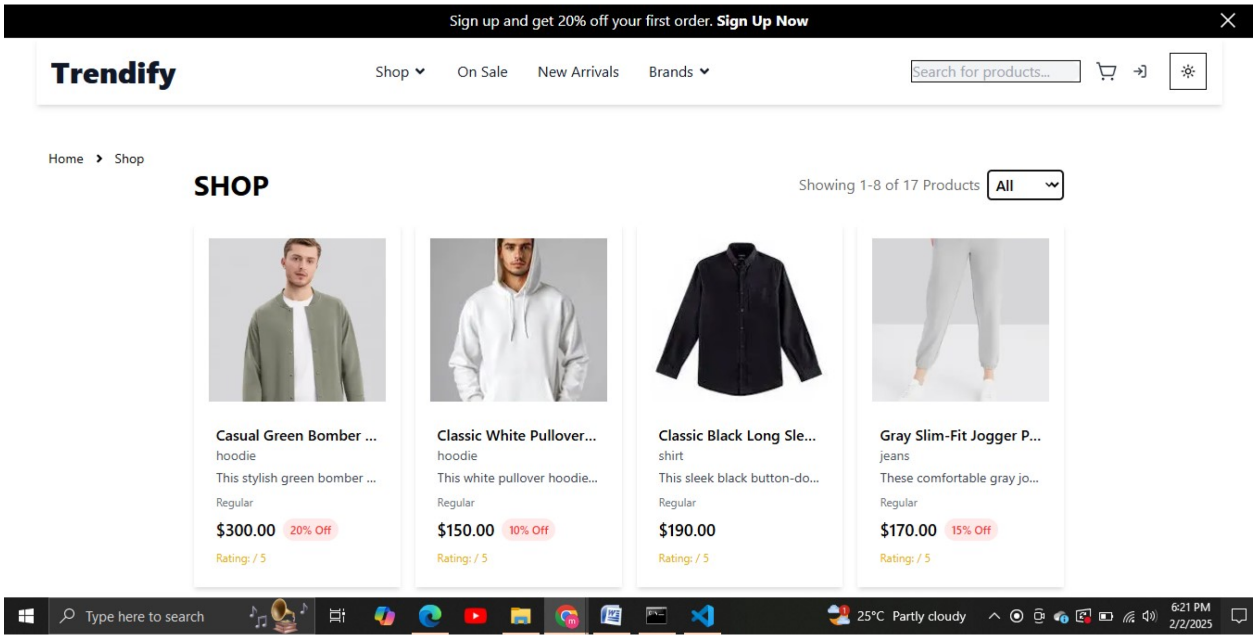
```
import {
  Pagination,
  PaginationContent,
  PaginationEllipsis,
  PaginationItem,
  PaginationLink,
  PaginationNext,
  PaginationPrevious,
} from "@/components/ui/pagination";

const PaginationDemo = ({
  productsPerPage,
  totalProducts,
  onPageChange,
  isNextButtonDisabled,
}: {
  productsPerPage: number;
  totalProducts: number;
  onPageChange: (pageNumber: number) => void;
  isNextButtonDisabled: boolean;
}) => {
  const totalPages = Math.ceil(totalProducts / productsPerPage);
  const pages = Array.from({ length: totalPages }, (_, i) => i + 1);

  return (
    <Pagination>
      <PaginationContent>
        <PaginationItem>
          <PaginationPrevious href="#" onClick={() => onPageChange(1)} />
        </PaginationItem>

        {pages.map((page) => (
          <PaginationItem key={page}>
```

```
            <PaginationLink
              href="#"
              onClick={() => onPageChange(page)}
              isActive={page === 1} // Assume page 1 is active initially
            >
              {page}
            </PaginationLink>
          </PaginationItem>
        ))}

        {totalPages > 5 && (
          <PaginationItem>
            <PaginationEllipsis />
          </PaginationItem>
        )}

        <PaginationItem>
          <PaginationNext
            href="#"
            onClick={() => onPageChange(totalPages)}
            className={
              isNextButtonDisabled ? "opacity-50 cursor-not-allowed" : ""
            }
          />
        </PaginationItem>
      </PaginationContent>
    </Pagination>
  );
};

export default PaginationDemo;
```

Category Filter:

I have rendered all my products on the shop page, which appears when clicking the "Shop" link in the navbar. I have also added a category-wise filter option on the same page, which is working successfully.

```
"use client";
import React, { useState, useEffect } from "react";
import { MdKeyboardArrowDown } from "react-icons/md";
import { BreadcrumbWithCustomSeparator } from "./breadcrumbs";
import Link from "next/link";
import PaginationDemo from "./pignation";
import Image from "next/image";

// Define the Product type
type Product = {
  _id: string;
  name: string;
  description: string;
  price: number;
  imageUrl: string;
  category: string;
  discountPercent: number;
  isNew: boolean;
  colors: string[];
  sizes: string[];
  rating: number;
};

const AllProductsSection: React.FC = () => {
  const [products, setProducts] = useState<Product[]>([]);
  const [loading, setLoading] = useState<boolean>(true);
  const [currentPage, setCurrentPage] = useState<number>(1);
  const [productsPerPage] = useState<number>(8);
  const [selectedCategory, setSelectedCategory] = useState<string>("All");

  useEffect(() => {
    const fetchProducts = async () => {
      try {
        const response = await fetch(
          `${process.env.NEXT_PUBLIC_SITE_URL}/api/allproducts`,
        );
        const data = await response.json();
        setProducts(data.allProducts);
      } catch (error) {
        console.error("Error fetching products:", error);
      } finally {
        setLoading(false);
      }
    };

    fetchProducts();
  }, []);

  const categories = [
    "All",
    ...new Set(products.map((product) => product.category)),
  ];

  const filteredProducts =
    selectedCategory === "All"
      ? products
      : products.filter((product) => product.category === selectedCategory);

  const indexOfLastProduct = currentPage * productsPerPage;
  const indexOfFirstProduct = indexOfLastProduct - productsPerPage;
  const currentProducts = filteredProducts.slice(
    indexOfFirstProduct,
    indexOfLastProduct,
  );

  const handlePageChange = (pageNumber: number) => setCurrentPage(pageNumber);

  const isNextButtonDisabled = indexOfLastProduct >= filteredProducts.length;

  return (
    <section className="container py-12 px-4 md:px-12 max-w-full overflow-x-
hidden">
      <BreadcrumbWithCustomSeparator />
      <div className="flex flex-col items-center">
        <div className="w-full md:w-3/4">
          <div className="flex justify-between items-center mb-6">
            <h2 className="text-2xl md:text-3xl font-bold">SHOP</h2>
            <div className="flex items-center space-x-2 text-sm sm:text-base">
              <span className="text-black/60">
                Showing {indexOfFirstProduct + 1}-
                {Math.min(indexOfLastProduct, filteredProducts.length)} of{" "}
                {filteredProducts.length} Products
              </span>
              <div className="relative">
                <select
                  className="text-black font-semibold bg-white border p-1 rounded
cursor-pointer"
                  value={selectedCategory}
                  onChange={(e) => {
                    setSelectedCategory(e.target.value);
                    setCurrentPage(1);
                  }}>
                  {categories.map((category) => (
                    <option key={category} value={category}>
                      {category}
                    </option>
                  ))}
                </select>
                <MdKeyboardArrowDown className="absolute right-2 top-1/2
transform -translate-y-1/2 text-black" />
              </div>
            </div>
          </div>

          <div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-
cols-4 gap-4 w-full">
            {loading ? (
              <p className="text-center text-lg md:text-xl">Loading...</p>
            ) : (
              currentProducts.map((product) => (
                <Link key={product._id} href={`/productdetail/${product._id}`}>
                  <div className="p-2 sm:p-4 bg-white rounded-lg shadow-md
hover:shadow-lg transition-shadow duration-300">
                    <div className="w-full h-40 sm:h-44 mb-4 relative">
                      <Image
                        src={product.imageUrl}
                        alt={product.name}
                        width={400}
                        height={400}
                        className="rounded-lg w-full h-full object-cover max-w-
full"

                      />
                    </div>
                    <div className="p-2">
                      <h3 className="font-semibold text-sm md:text-base
truncate">

                        {product.name}
                      </h3>
                      <p className="text-xs md:text-sm text-gray-600 truncate">
                        {product.category}
                      </p>
                      <p className="text-sm text-gray-700 mt-1 truncate">
                        {product.description}
                      </p>
                      <div className="flex justify-start items-center mt-2">
                        <span
                          className={`text-xs ${product.isNew ? "text-green-500"
: "text-gray-500"}`}>

                          {product.isNew ? "New Arrival" : "Regular"}
                        </span>
                      </div>
                      <div className="flex items-center justify-start mt-2 space-
x-2">
                        <p className="text-black font-semibold text-sm md:text-
lg">
                          ${product.price.toFixed(2)}
                        </p>
                        {product.discountPercent > 0 && (
                          <span className="text-[#FF3333] font-medium text-xs py-
1 px-2 bg-[#FF33331A] rounded-full">
                            {product.discountPercent}% Off
                          </span>
                        )}
                      </div>
                      <div className="flex justify-start items-center mt-2">
                        <span className="text-xs text-yellow-500">
                          Rating: {product.rating} / 5
                        </span>
                      </div>
                    </div>
                  </div>
                </Link>
              ))
            )}
          </div>

          <div className="border-t my-4 sm:my-6 lg:my-8 w-full"></div>

          <PaginationDemo
            productsPerPage={productsPerPage}
            totalProducts={filteredProducts.length}
            onPageChange={handlePageChange}
            isNextButtonDisabled={isNextButtonDisabled}
          />
        </div>
      </div>
    </section>
  );
};

export default AllProductsSection;
```


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

Trendify

Shop On Sale New Arrivals Brands

Search for products...

Home > Shop

SHOP

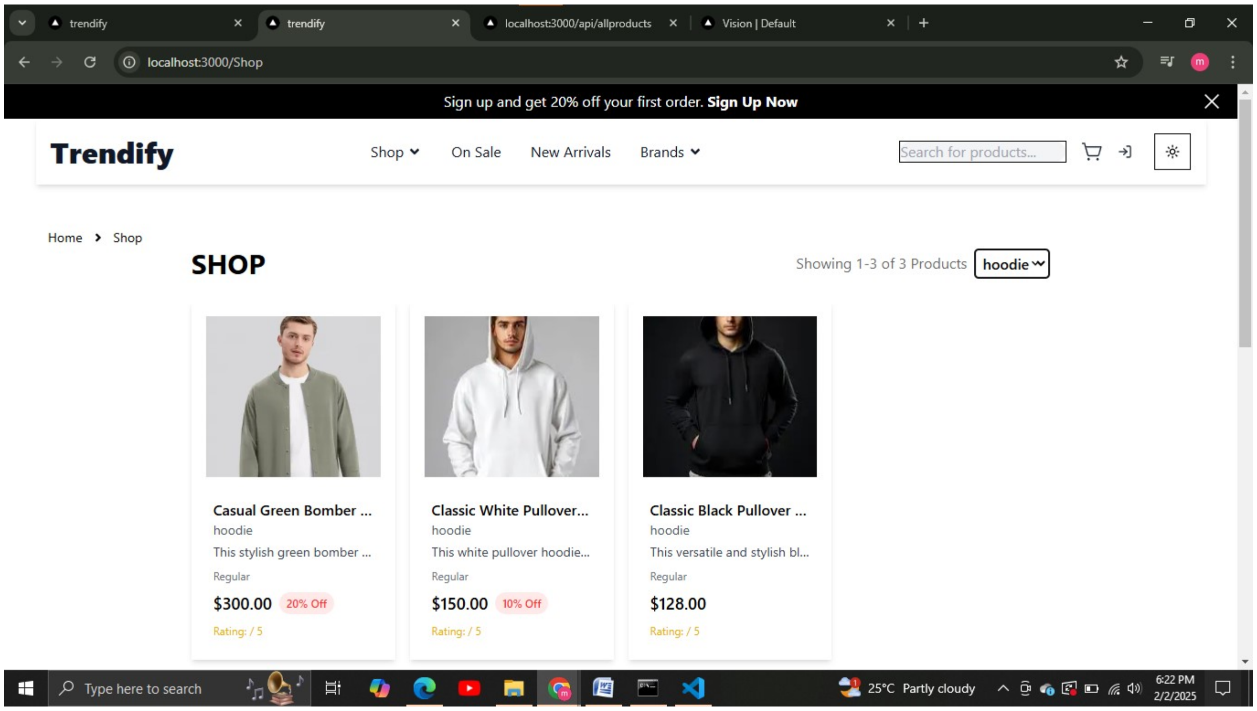
Showing 1-8 of 17 Products All

Casual Green Bomber ...
hoodie
This stylish green bomber ...
Regular
$300.00 20% Off
Rating: / 5

Classic White Pullover...
hoodie
This white pullover hoodie...
Regular
$150.00 10% Off
Rating: / 5

Classic Black Long Sle...
shirt
This sleek black button-do...
Regular
$190.00
Rating: / 5

Gray Slim-Fit Jogger P...
jeans
These comfortable gray jo...
Regular
$170.00 15% Off
Rating: / 5


localhost:3000/Shop

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

Trendify

Shop On Sale New Arrivals Brands

Search for products...

Home > Shop

SHOP

Showing 1-3 of 3 Products hoodie

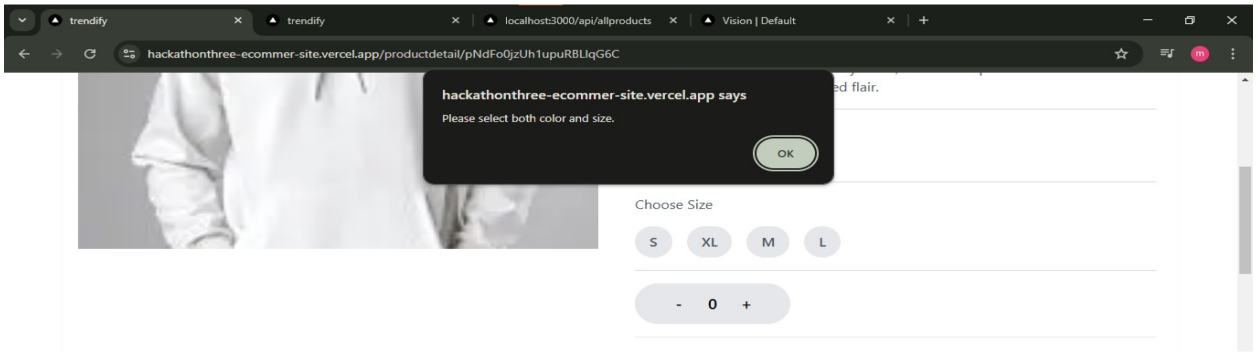
Casual Green Bomber ...
hoodie
This stylish green bomber ...
Regular
$300.00 20% Off
Rating: / 5

Classic White Pullover...
hoodie
This white pullover hoodie...
Regular
$150.00 10% Off
Rating: / 5

Classic Black Pullover ...
hoodie
This versatile and stylish bl...
Regular
$128.00
Rating: / 5

Alart:

If "Add to Cart" is clicked without selecting colors or size, an alert will appear.



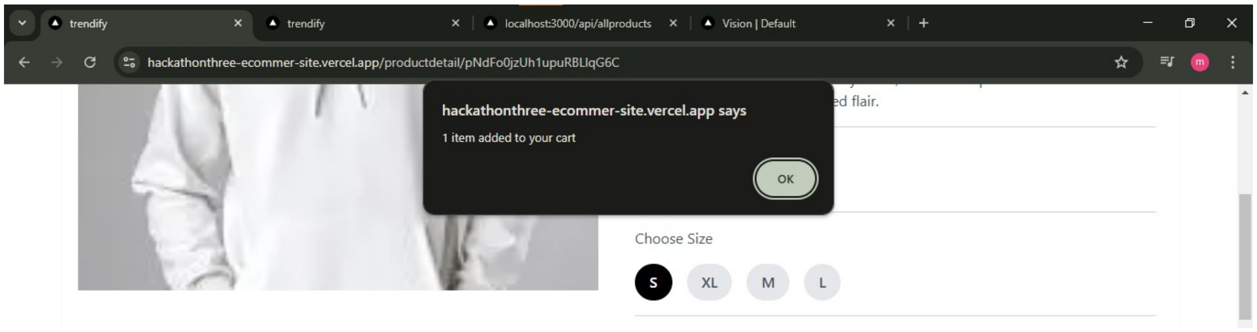
trendify
trendify
localhost:3000/api/allproducts
Vision | Default
hackathonthree-ecommer-site.vercel.app/productdetail/pNdFo0jzUh1upuRBLlqG6C

hackathonthree-ecommer-site.vercel.app says

Please select both color and size.

OK
ed flair.
Choose Size
S
XL
M
L
-
0
+

When color and size are selected and "Add to Cart" is clicked, another alert will appear, informing that the item has been added to the cart.



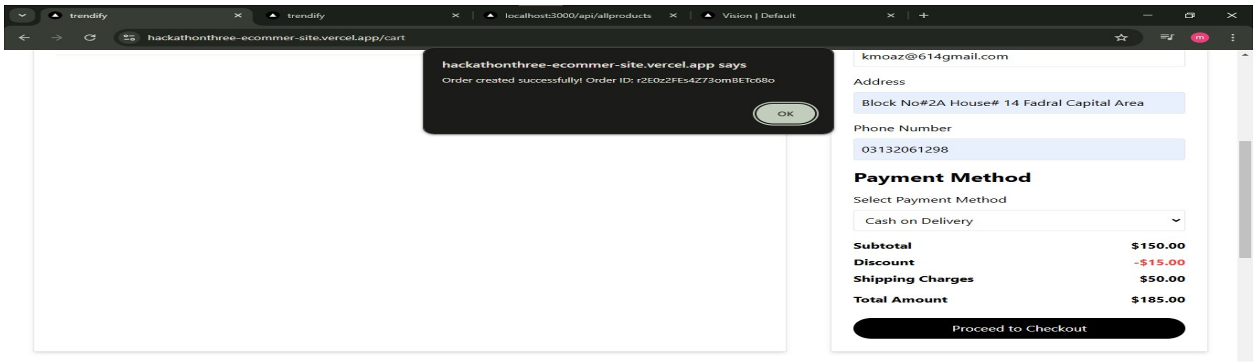
trendify
trendify
localhost:3000/api/allproducts
Vision | Default
hackathonthree-ecommer-site.vercel.app/productdetail/pNdFo0jzUh1upuRBLlqG6C

hackathonthree-ecommer-site.vercel.app says

1 item added to your cart

OK
ed flair.
Choose Size
S
XL
M
L

After filling in your details, when you proceed to checkout, the website will give you another alert confirming that your order has been successfully created.



trendify
trendify
localhost:3000/api/allproducts
Vision | Default
hackathonthree-ecommer-site.vercel.app/cart

hackathonthree-ecommer-site.vercel.app says

Order created successfully! Order ID: r2E0z2FEs4Z73omBETc68o

kmoaz@614gmail.com

Address: Block No#2A House# 14 Fadral Capital Area

Phone Number: 03132061298

Payment Method

Select Payment Method: Cash on Delivery

Subtotal	**$150.00**
Discount	**-$15.00**
Shipping Charges	**$50.00**
Total Amount	**$185.00**

Task 2:

Reusable Components:

In my project, I created modular components like `ProductCard` and `CategoryFilter` that can be reused across multiple pages. These components receive data through props, which maintains flexibility and reduces code duplication.

State Management:

I used React's `useState` and `useContext` hooks for state management.

- **Local State** is handled using `useState`, which manages component-specific data.
- **Global State** is managed using `useContext`, allowing data to be shared across multiple components, like user login state or cart data.

Styling:

I used modern styling libraries like **Tailwind CSS** and **styled-components**.

- **Tailwind CSS** helps in creating responsive designs quickly, ensuring that the layout looks great on both mobile and desktop.
- This styling method keeps the code maintainable and scalable.

Performance Optimization:

I implemented a few performance optimization techniques such as:

- **Lazy loading**: This loads images only when they are needed, improving the site's performance.
- **Pagination / Infinite Scrolling**: For large datasets, these techniques efficiently load data and improve the site's loading speed.