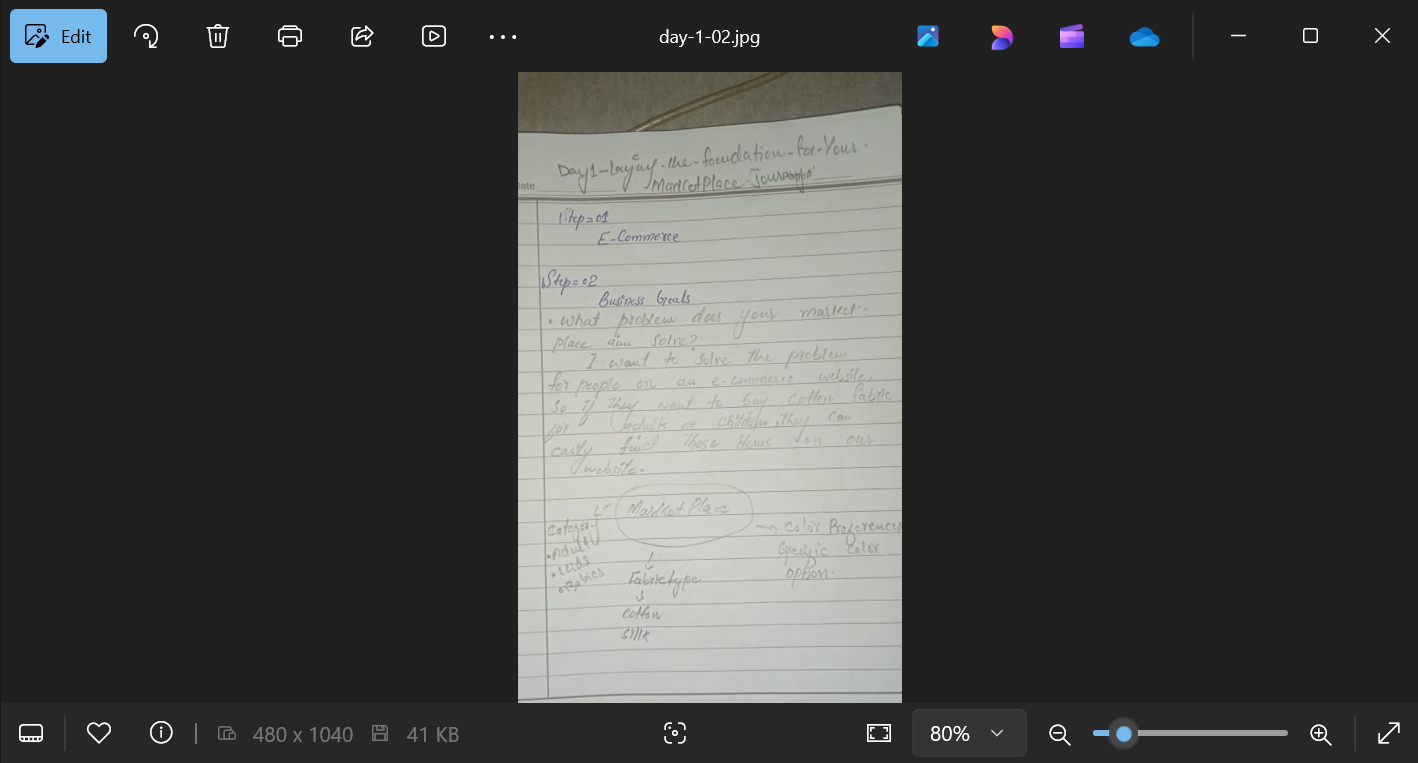
Day 1: Laying The Foundation For Your Marketplace Journey.

Instructor Shahzaib Sahito

Step 1: Choose Your Marketplace Type

* General E-Commerce
* Q-Commerce
* Rental E-Commerce
* Options:
* General E-Commerce: refers to the buying and selling of goods or services over the internet through online platforms or websites. It includes a wide range of transactions, such as retail shopping, wholesale trading, online services, and digital product purchases. Examples include platforms like Amazon, eBay, or Shopify stores.
* Q-Commerce: (Quick Commerce) is a type of e-commerce that focuses on delivering goods to customers very quickly, often within minutes or hours of ordering. It typically involves small quantities of everyday items like groceries, snacks, or essentials, delivered through local fulfillment centers or delivery partners. Examples include services like Gopuff, Getir, and DoorDash.
* Rental E-Commerce: refers to online platforms where customers can rent products or services for a specific period instead of buying them. It includes items like clothing, furniture, vehicles, or equipment. Examples of rental e-commerce platforms include Rent the Runway and Airbnb.

**Day-2**

**Objective**:

The purpose of this code is to define basic data structures for e-commerce system. These schemas model products, orders, and delivery zones

**Product:**

Details of products (like id, name, price, stock, category, etc.) can be stored and managed.

**Order:**

Customer orders and their details (such as customer information, order items, status, total amount) can be tracked.

**Delivery Zone:**

Delivery zones and their drives can be managed to ensure smooth delivery operations.

Key Points From the code (Schemas).

**Product Schema:**

**Id**: Unique Identifier for the product.

**Name**: Name of Product.

**Price**: Price of Product.

**Stock**: Number of items available in stock.

**Category**: Product Category (can ‘Women’, ‘Men’, or ‘Kids’).

**Size**: (optional) Available Size (‘S’, ‘M’, ‘L’, ‘XL’).

**Color**: (optional)Available color.

**Description**: (optional) Product Description.

**ImageURL**: (optional) URL for the product image.

**Product API Endpoints**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Endpoint** | **Description** | **Status Codes** |
| GET | /api/products | List all products | 200 (OK), 500 (Internal Error) |
| GET | /api/products/{id} | Get product by ID | 200 (OK), 404 (Not Found) |
| POST | /api/products | Create a new | 201 [Created],400 (Bad request |
| PUT | /api/products/{id} | Update product by ID | 200 (OK), 400 (Bad Request) 404 Not Found |
| DELETE | /api/products/{id} | Delete product by ID | 200 (OK), 404 Not Found |

**Order Schema:**

CustomerInfo: Includes Customer detail like name, email, phone, (optional), and address.

OrderItem: Represents a Single Product in the order, with fields like productID, quantity, and price.

**Order:**

orderId: Unique order identifier.

customerInfo: Customer’s details.

orderDetail: List of items orderd.

Status: Order Status (e.g, ‘Pending’, ‘Processing’, ‘Shipped’, ‘Delivered’,’Cancelled’.)

orderDate: Date when the order was placed.

Totalamount: Total amount of the order.

* *The shipment or delivery that has been defined will not actually show less because that functionality hasn't been provided.*
* **Orders API Endpoints**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Endpoint** | **Description** | **Status Codes** |
| GET | /api/orders | List all orders | 200 (OK), 500 (Internal Error) |
| GET | /api/ orders /{id} | Get orders by ID | 200 (OK), 404 (Not Found) |
| POST | /api/ orders | Create a new orders | 201 [Created],400 (Bad Request) |
| PUT | /api/ orders /{id} | Update orders by ID | 200 (OK), 400 (Bad Request) 404 Not Found |
| DELETE | /api/ orders /{id} | Delete orders by ID | 200 (OK), 404 Not Found |

**Delivery Zone Schema:**

CoverageArea: Specifies a city and list of postal codes covered by a delivery zone.

AssignedDriver: Information about a delivery driver, including driverid, name, and contact

**DeliveryZone:**

**zoneName**: Name of the delivery zone.

coverageArea: Area Covered by the delivery zone.

assignedDrivers: List of drivers assigned to this zone.

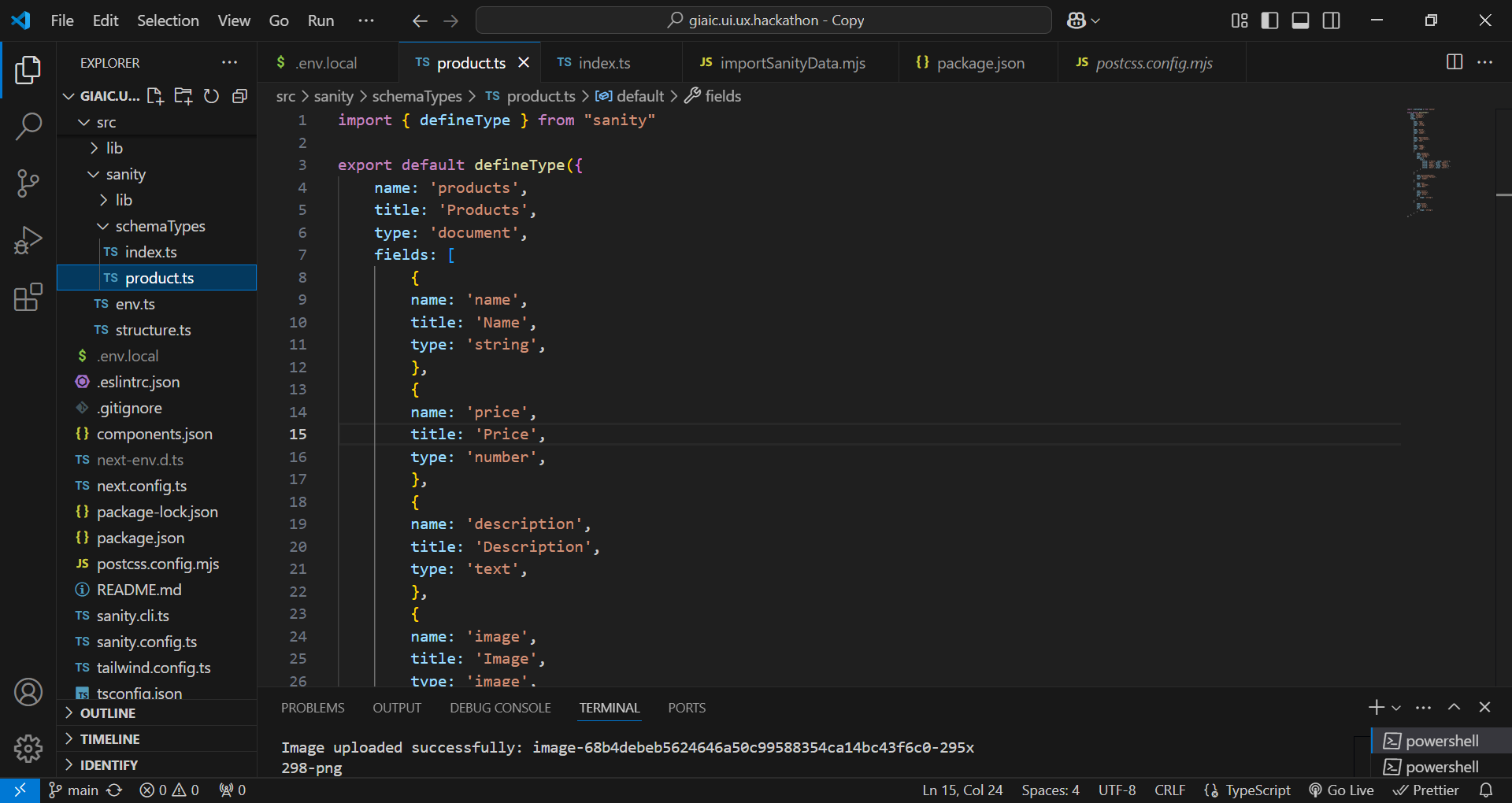
**Delivery API Endpoints**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Endpoint** | **Description** | **Status Codes** |
| GET | /api/deliveryZones | List all deliveryZones | 200 (OK), 500 (Internal Error) |
| GET | /api/ deliveryZones /{id} | Get deliveryZones by ID | 200 (OK), 404 (Not Found) |
| POST | /api/ deliveryZones | Create a new deliveryZones | 201 [Created],400 (Bad Request) |
| PUT | /api/ deliveryZones /{id} | Update deliveryZones by ID | 200 (OK), 400 (Bad Request) 404 Not Found |
| DELETE | /api/ deliveryZones /{id} | Delete deliveryZones by ID | 200 (OK), 404 Not Found |

*Day-3*

Sanity ko install krna ha

Schema list dani ha



**Integrating Sanity with Next.js**

In this blog post, we will explore how to integrate **Sanity** into an existing **Next.js** project. The focus will be on configuring environmental variables, setting up a Sanity schema, and importing data from an external API. This guide assumes you already have a **Next.js** project established and Sanity installed.

**Table of Contents**

1. [Setting Up Environment Variables](https://github.com/subhankaladi/Templates/blob/main/Template-1/scripts/readme.md#setting-up-environment-variables)
2. [Obtaining Sanity Project ID and API Token](https://github.com/subhankaladi/Templates/blob/main/Template-1/scripts/readme.md#obtaining-sanity-project-id-and-api-token)
3. [Creating the Sanity Schema](https://github.com/subhankaladi/Templates/blob/main/Template-1/scripts/readme.md#creating-the-sanity-schema)
4. [Setting Up the Data Import Script](https://github.com/subhankaladi/Templates/blob/main/Template-1/scripts/readme.md#setting-up-the-data-import-script)
5. [Running the Import Script](https://github.com/subhankaladi/Templates/blob/main/Template-1/scripts/readme.md#running-the-import-script)
6. [Conclusion](https://github.com/subhankaladi/Templates/blob/main/Template-1/scripts/readme.md#conclusion)

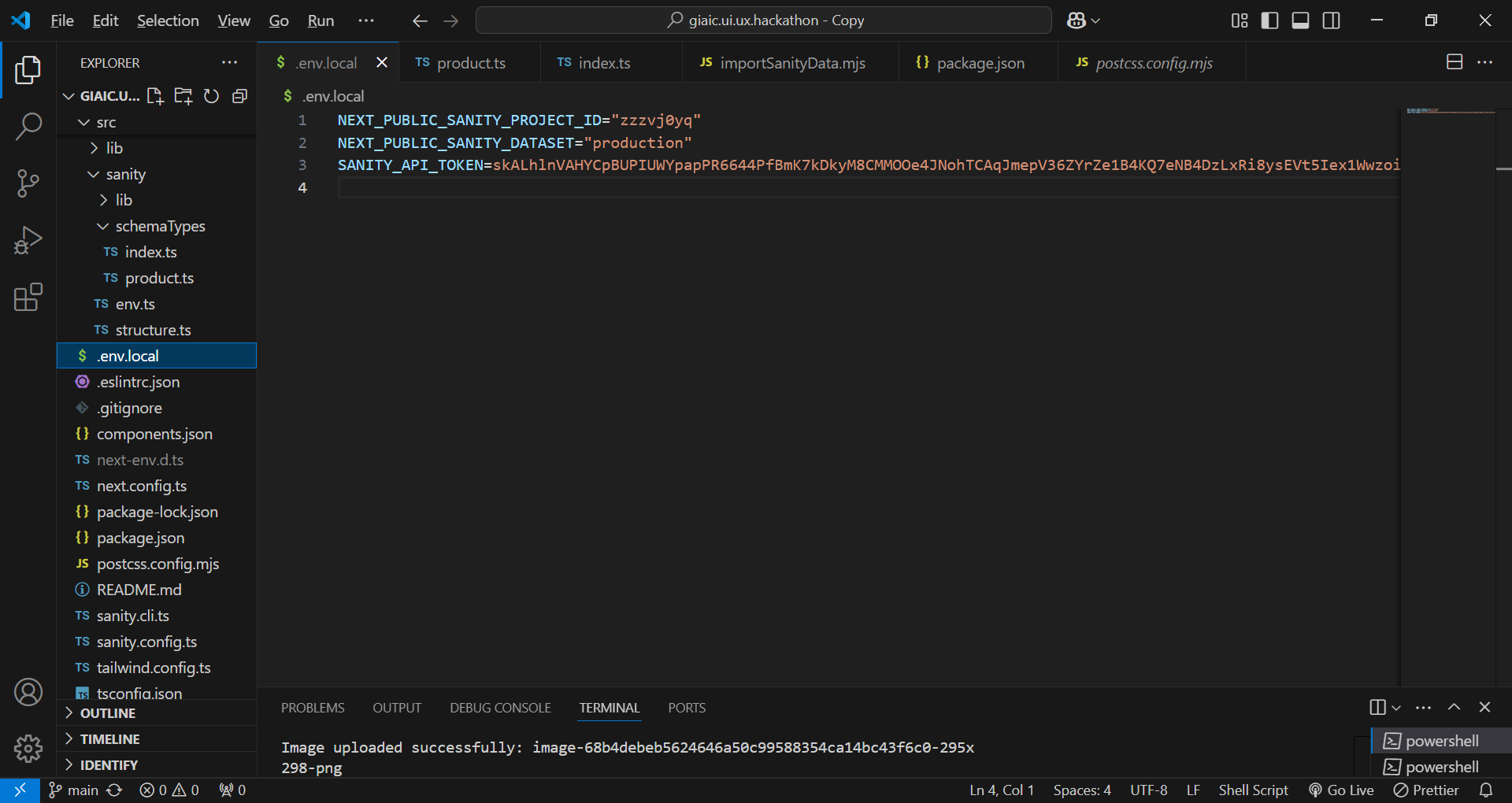
**Setting Up Environment Variables**

Begin by setting up your environment variables. If a .env.local file doesn't already exist in your project's root directory, create one. Then, add the following variables:

NEXT\_PUBLIC\_SANITY\_PROJECT\_ID=your\_project\_id

NEXT\_PUBLIC\_SANITY\_DATASET=production

SANITY\_API\_TOKEN=your\_sanity\_token



**Note:** Variables prefixed with NEXT\_PUBLIC\_ will be exposed to the browser, so be cautious about what you prefix.

**Installing Necessary Packages**

Now, let’s install the necessary packages. Run the following command in your terminal:

npm install @sanity/client axios dotenv

**Adding a Script to package.json**

To run the import script, we need to add a new script to our package.json file. Open your package.json and add the following to the "scripts" section:

"scripts": {

"dev": "next dev --turbopack",

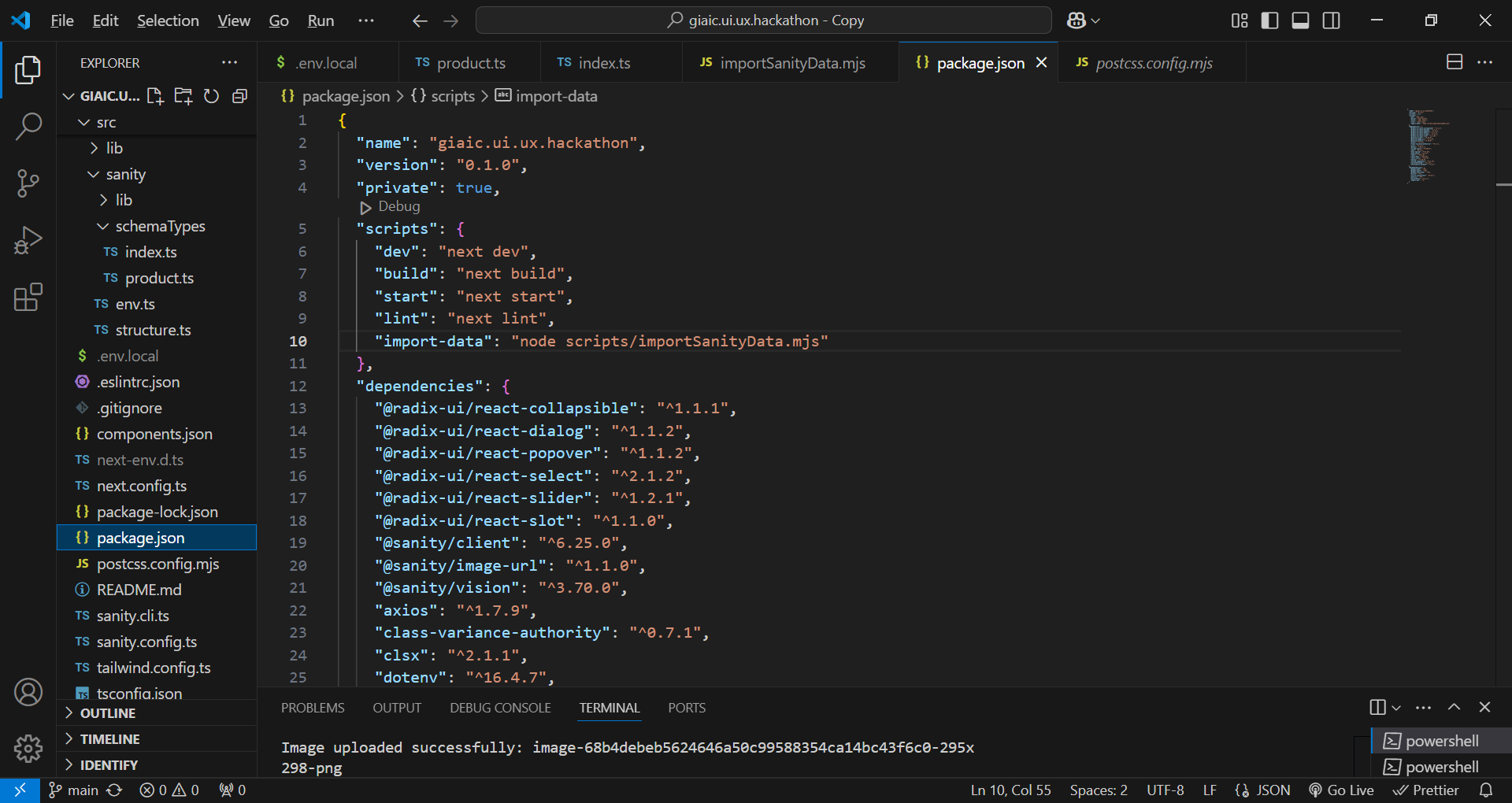
"build": "next build",

"start": "next start",

"lint": "next lint",

"import-data": "node scripts/importSanityData.mjs"

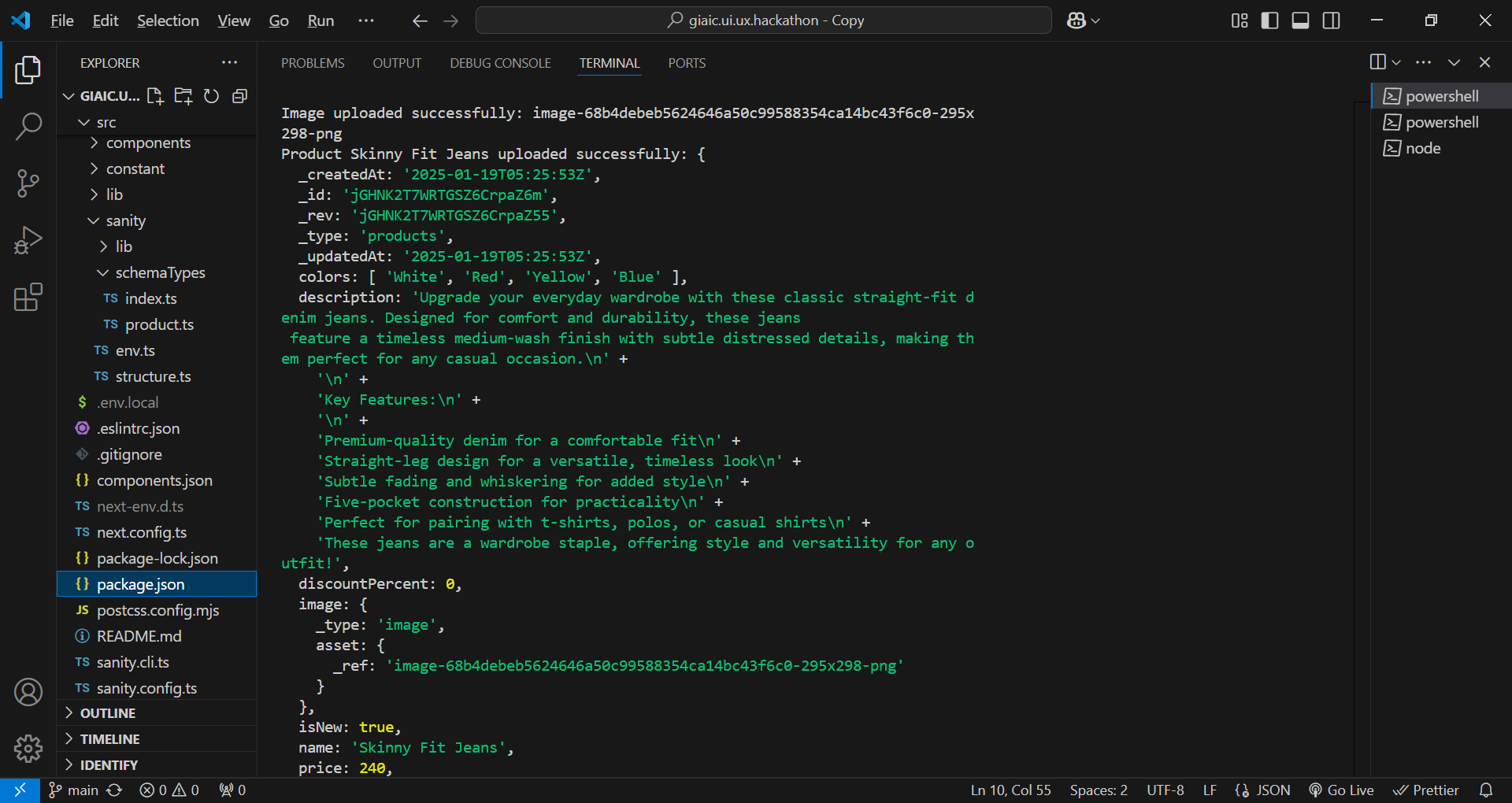
}



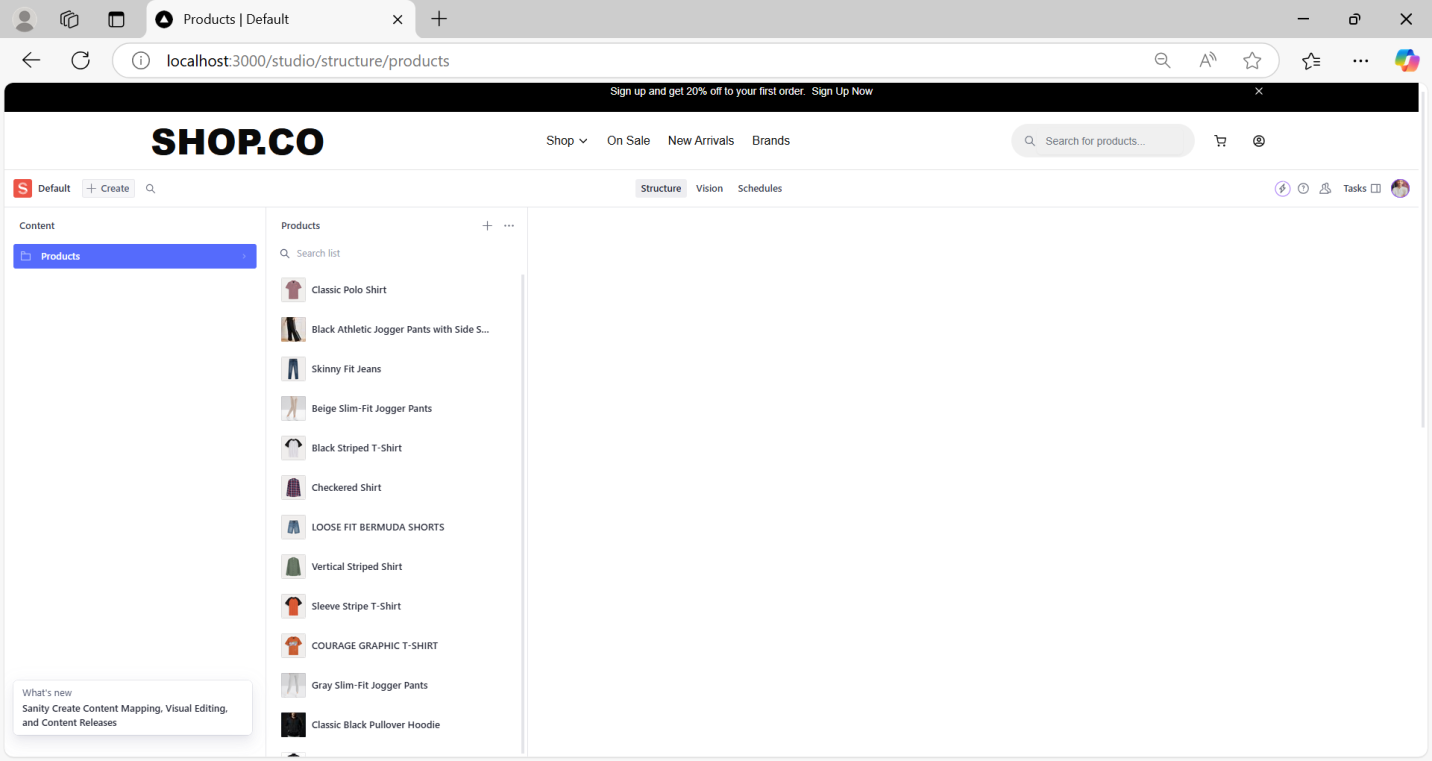
**Running the Import Script**

Now you can run the import script using:

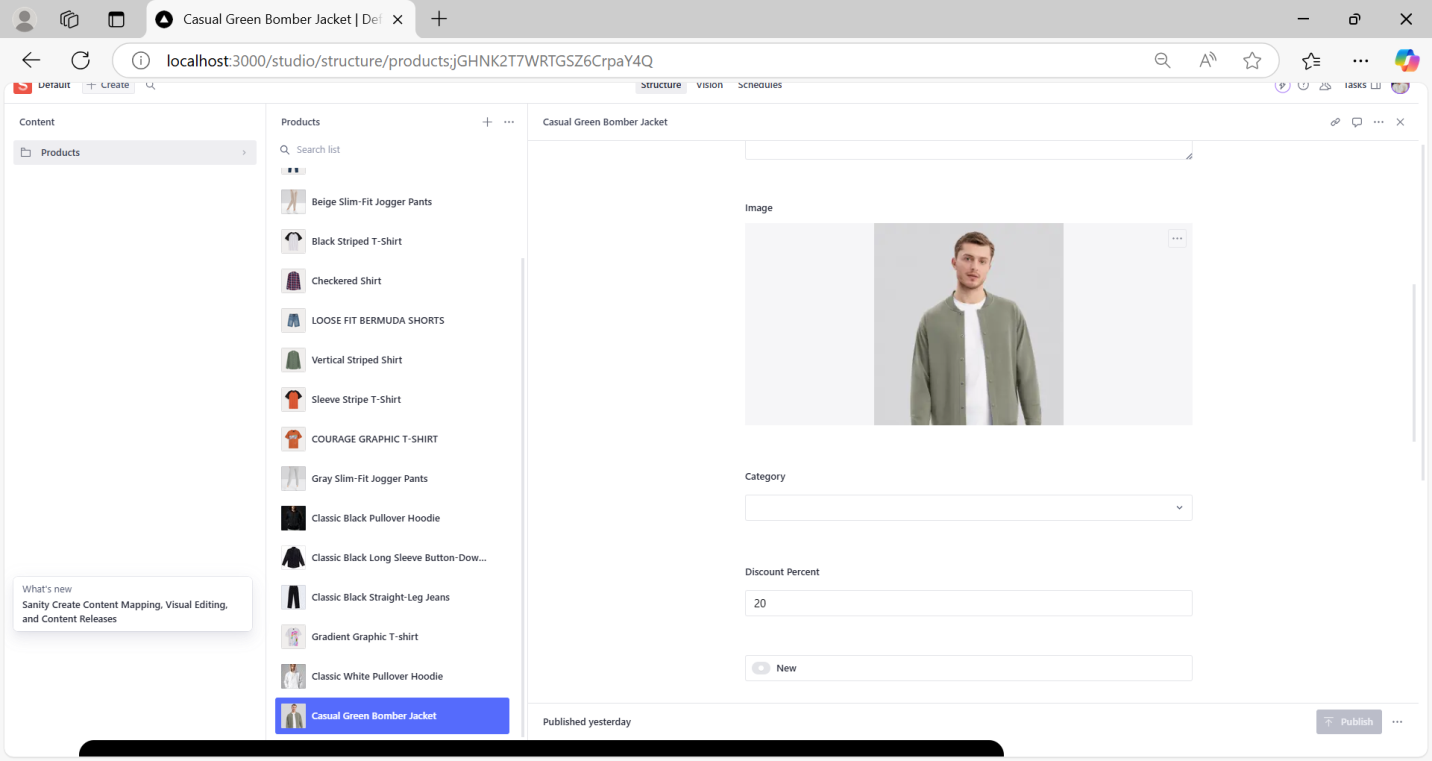
npm run import-data



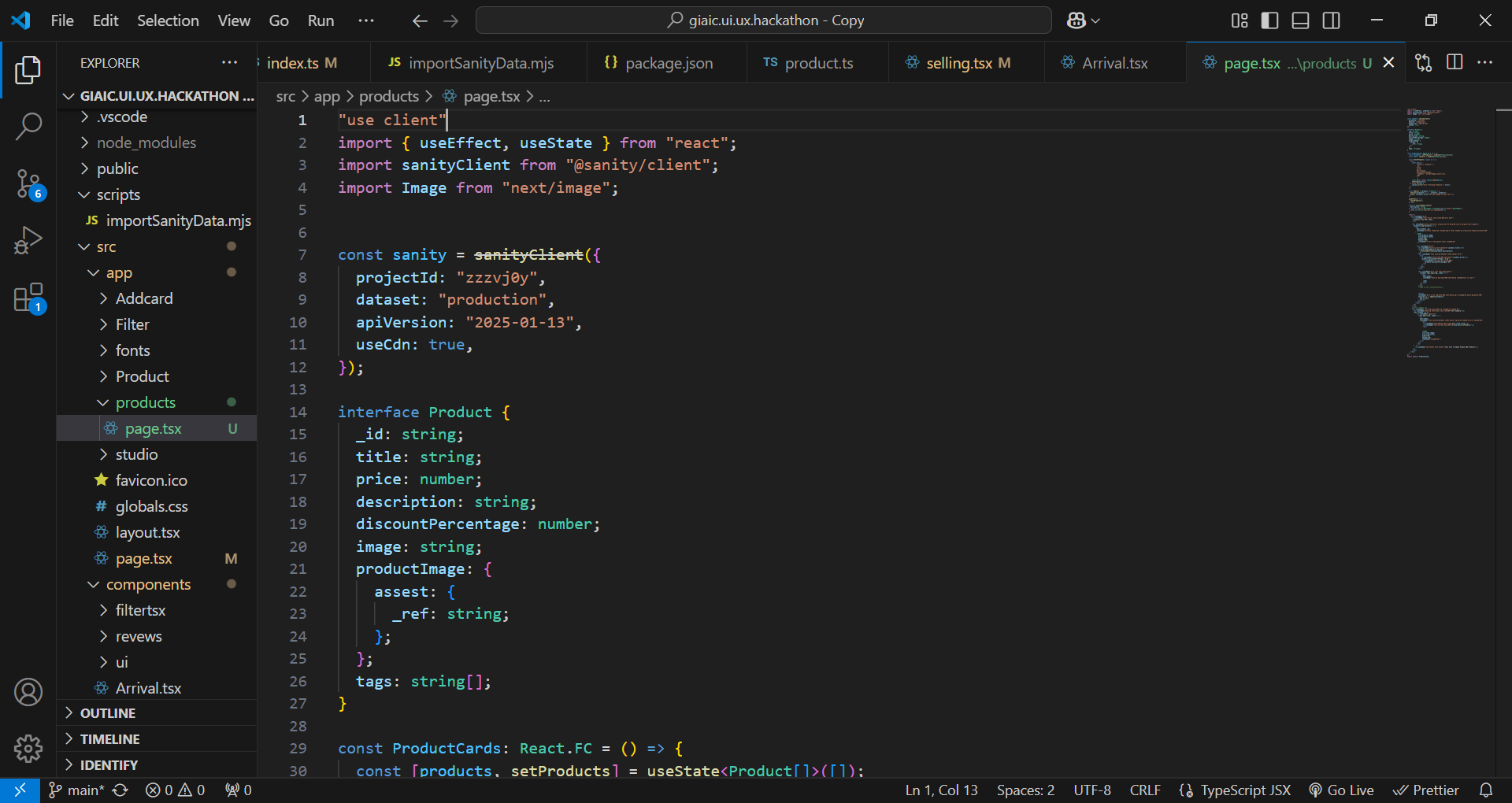
Studio Display



Second Picture Studio



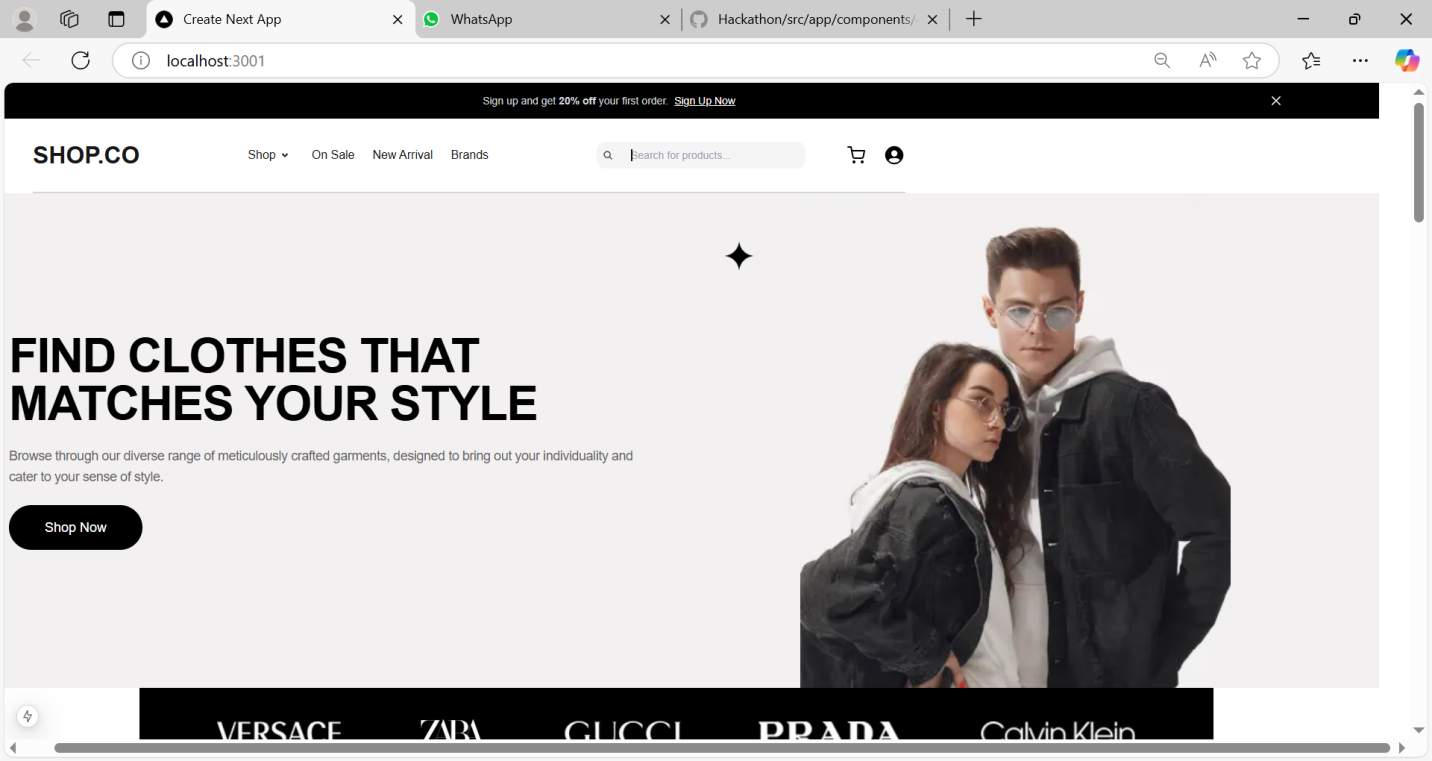
* This is the data being fetched, and this is its picture. I've tried a lot, but it’s showing some issues on the display. Inshallah, I will solve this very soon as well



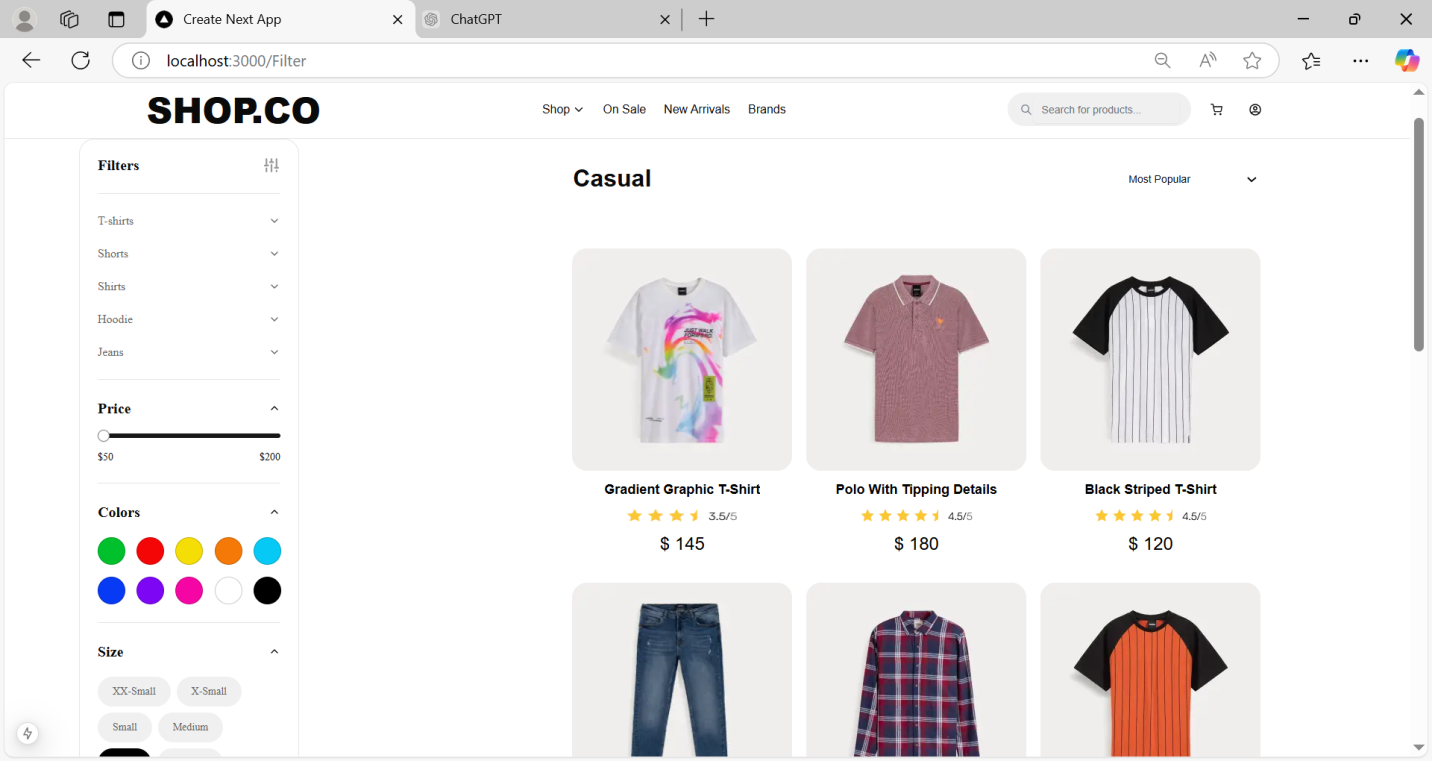
Day-5

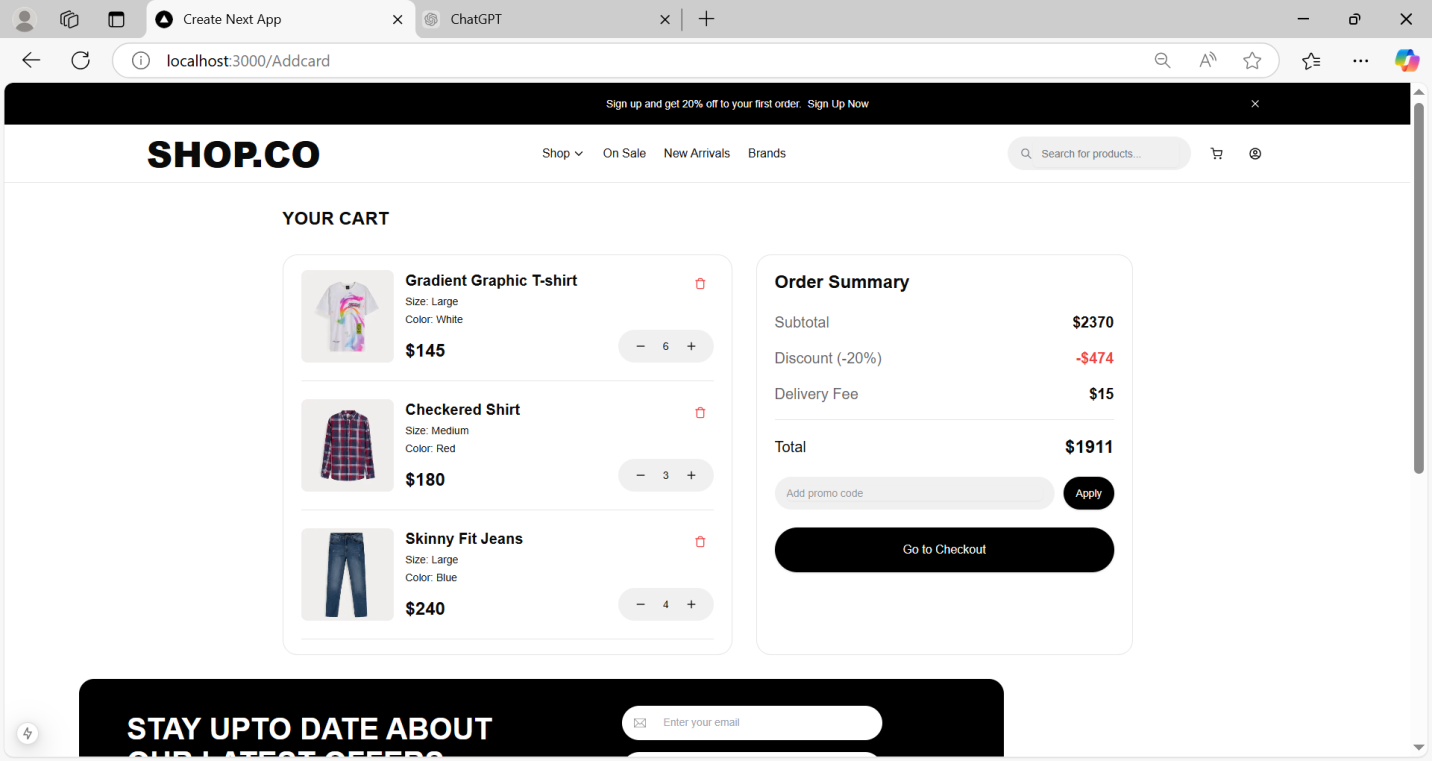
* *I will try to My level Best.*

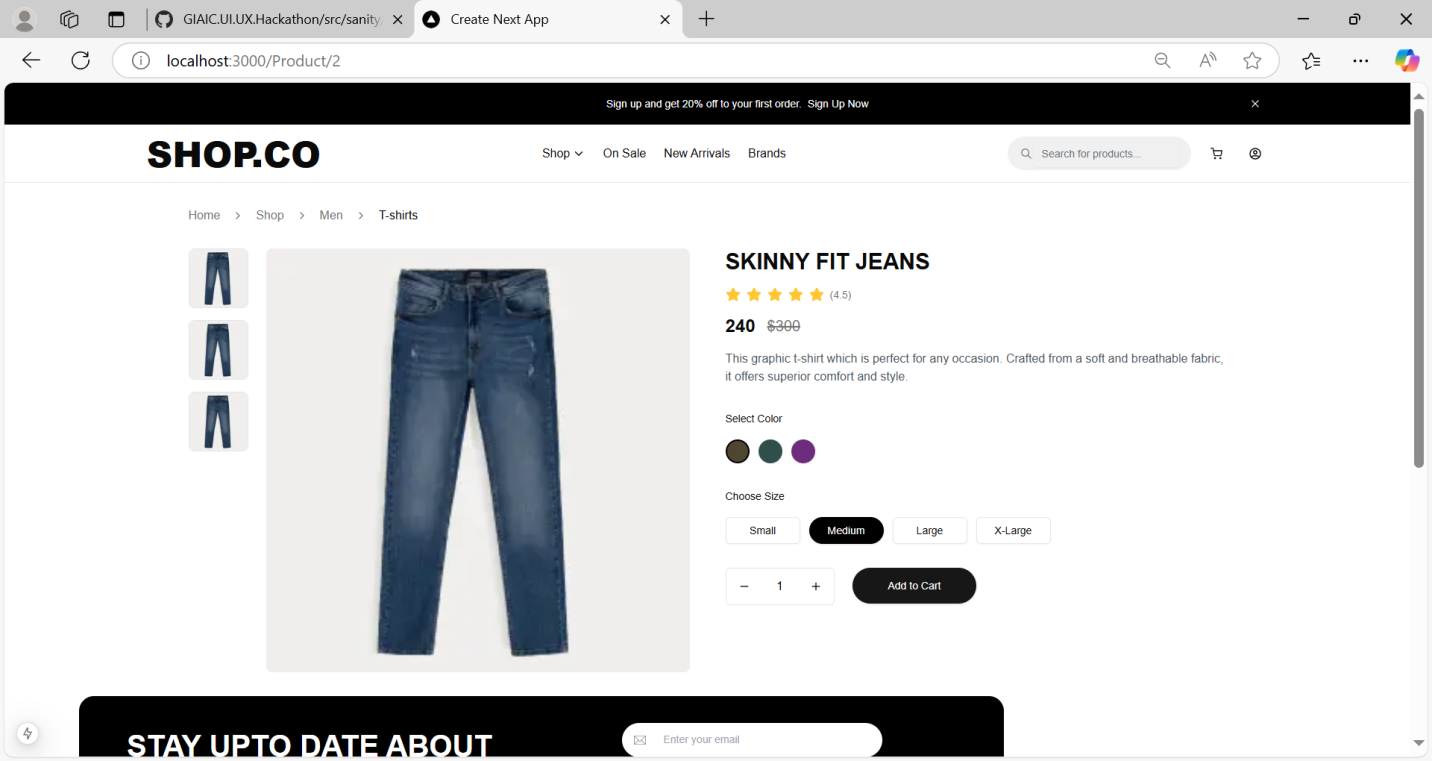
Ya front ki picture ha isma jo Navbar ma jo bi ha Shop Ho gaya onSale ha NewArrival ha Brands ha agar user isma kisi ko bi click kra ga example NewArrival pa click kra ga to us brand pa chala jai ga agar onSale pa click kra ga to onSale ki brands pa jai ga

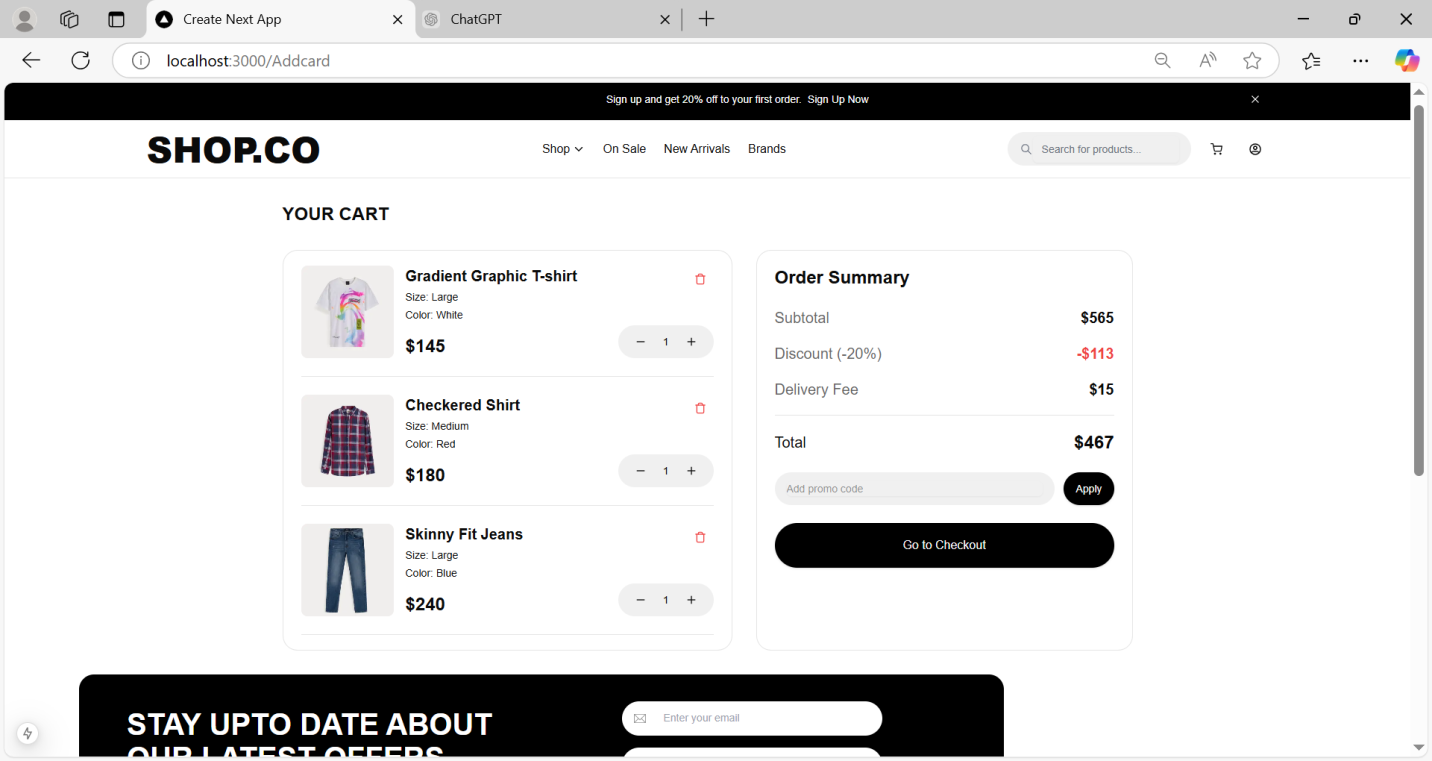


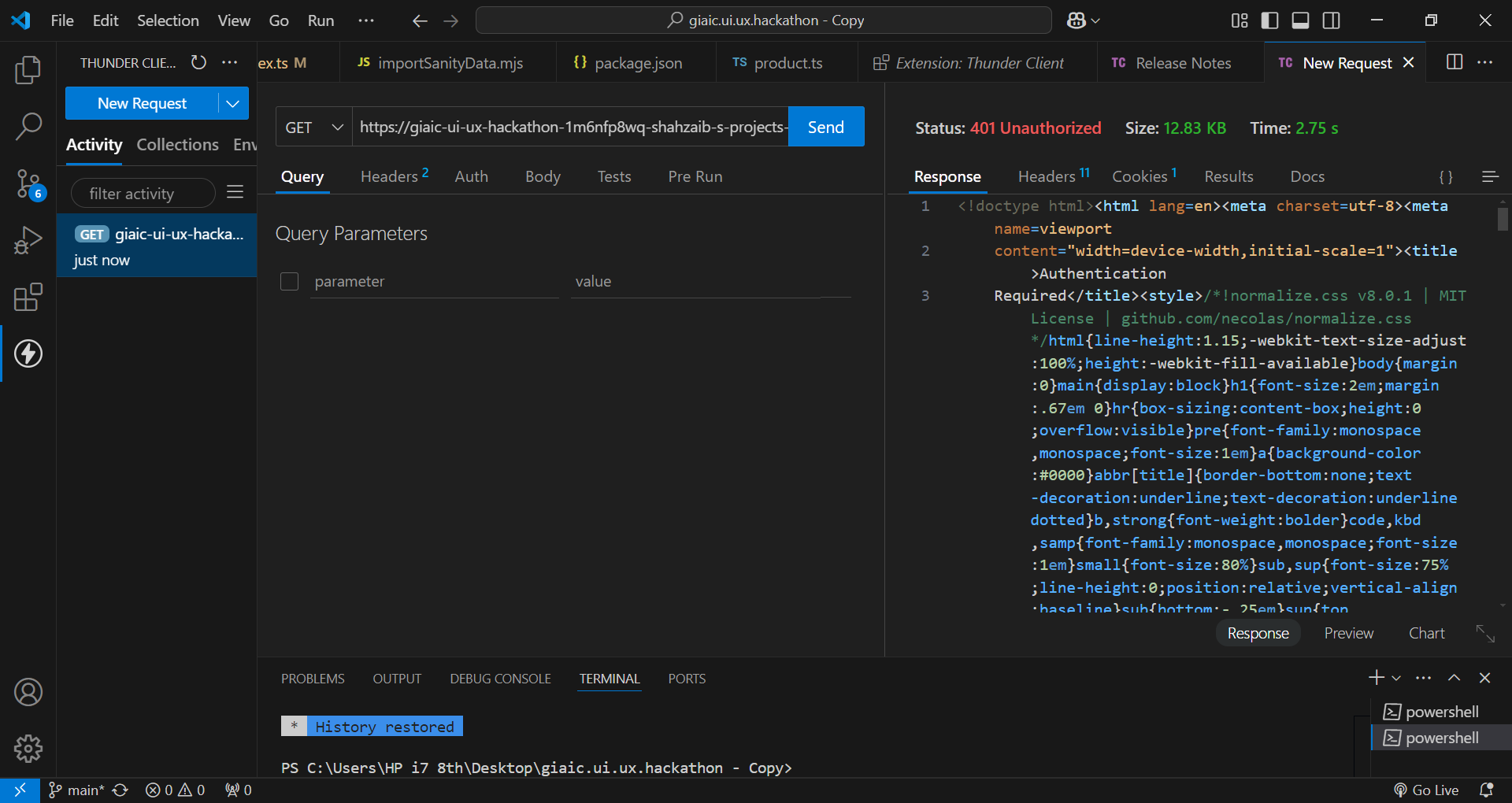
Ya sari functionalty dikhai dati ha niche picture ma











|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Case ID | Description | Steps | Expected Result | Actual Result | Status | Remarks |
| TC001 | Validate product listing | Open product page > Verify products | Products displayed | Products  displayed | Passed | No issues found |
| TC002 | Test API error handling | Disconnect API > Refresh page | Show fallback message | Fallback message shown | Passed | Handled gracefully |
| TC003 | Check cart functionality | Add item to cart > Verify cart | Cart updates correctly | Cart updates correctly | Passed | Works expected |
| TC004 | Test responsiveness layout | Resize browser window > Check layout | Layout adjusts properly | Layout adjusts properly | Passed | Responsive verified |

npm install –g lighthouse

lighthouse <http://localhost:3000> --view