

Day 3 - API Integration and Data Migration Report

Objective: The goal for Day 3 was to integrate API data into Sanity CMS for the Bandage project, enabling dynamic content updates for the marketplace. Instead of manually entering data, the API integration provided a more efficient and scalable solution.

1. Step 01: (Sanity CMS Schema Design)

To ensure the seamless handling of product data, I designed a schema called product in Sanity CMS. The schema includes the following fields.

The Fields:

title: The product title (string type)

description: A detailed description of the product (text type).

productImage: The main product image (image type).

price: The price of the product (number type).

discountPercentage: The discount percentage (number type).

isNew: A boolean flag indicating if the product is new (boolean type)

Size : it is string for the t-shirt size.

Color : it is string for the clothes color.

Code snaps:

```

1  import { defineType } from "sanity"
2
3
4
5  export const product = defineType({
6    name: 'products',
7    title: 'Products',
8    type: 'document',
9    fields: [
10     {
11       name: 'name',
12       title: 'Name',
13       type: 'string',
14     },
15
16     {
17       name: 'price',
18       title: 'Price',
19       type: 'number',
20     },
21   ],
22   {
23     name: 'description',
24     title: 'Description',
25     type: 'text',
26   },
27   {
28     name: 'image',
29     title: 'Image',
30     type: 'image',
31   },
32   {
33     name: "category",
34     title: "Category",
35     type: 'string',
36     options: {
37       list: [
38         {title: 'T-Shirt', value: 'tshirt'},
39         {title: 'Short', value: 'short'},
40         {title: 'Jeans', value: 'jeans'},
41         {title: 'Hoddie', value: 'hoodie'},
42         {title: 'Shirt', value: 'shirt'},
43       ]
44     }
45   },
46   {

```

```

42         {title: 'SHIRT', value: 'SHIRT' },
43     ]
44 }
45 },
46 {
47     name:"discountPercent",
48     title:"Discount Percent",
49     type: 'number',
50 },
51 {
52     name:"new",
53     type: 'boolean',
54     title:"New",
55 },
56 {
57     name:"colors",
58     title:"Colors",
59     type: 'array',
60     of:[
61         {type: 'string'}
62     ]
63 },
64 {
65     name:"sizes",
66     title:"Sizes",
67     type: 'array',
68     of:[
69         {type: 'string'}
70     ]
71 }
72 ],
73 }

```

Step 02 : (API Integration and Data Migration)

API Data Fetching I retrieved product data from an external API, which contained information such as images, titles, descriptions, prices, and tags. This data was then mapped to the appropriate fields in the Sanity CMS schema.

```

type Product = {
  image(image:string): unknown;
  discountPercent: number;
  _id : string;
  name : string;
  price : number;
  image_url : string;
};

// check

const [CategoryProducts,setCategoryProducts] = useState<Product[]>([]);

useEffect(() => {
  const fetchCategoryProducts = async () => {
    const products = await client.fetch(`*[_type == "products"][1..2]
    + *[_type == "products"][4..9] + *[_type == "products"][11..11]`);

    setCategoryProducts(products);
  };

  fetchCategoryProducts();
}, []);

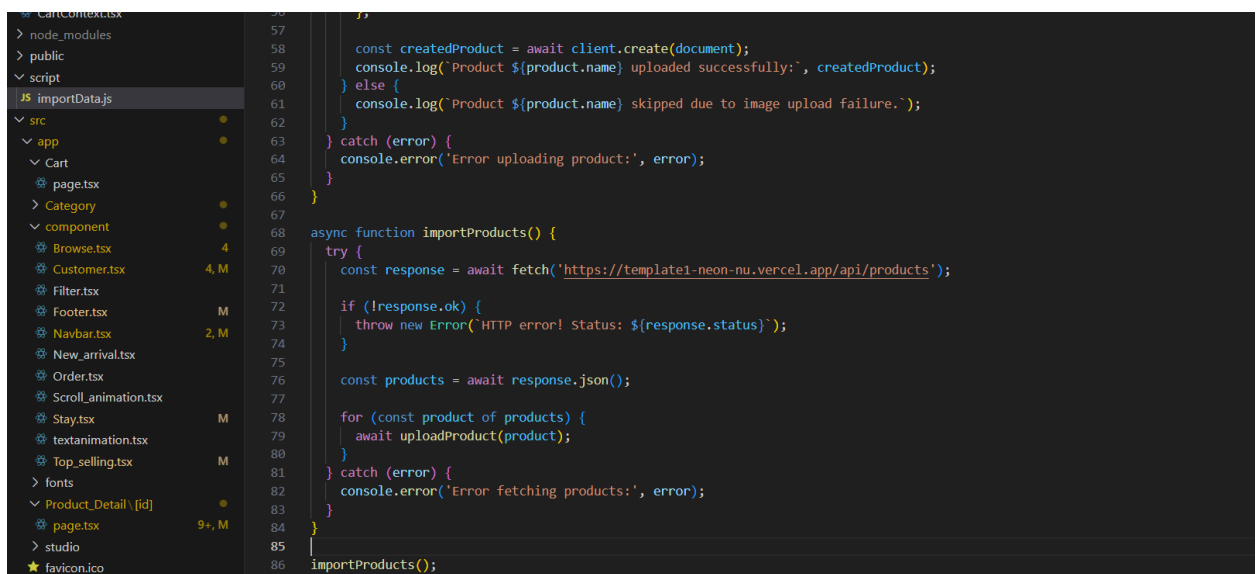
// Function to show the filter overlay
const handleShowFilter = () => {
  setShowFilter(true);
};

// Function to hide the filter overlay
const handleCloseFilter = () => {
  setShowFilter(false);
};

```

Data Population in Sanity CMS : After getting the API data, I filled in the product fields in Sanity CMS automatically. This made sure the product information was always correct and consistent across the platform.

Data Migration: Using the Sanity CLI, I backed up the dataset from Sanity CMS and later imported it again for testing. This process made sure all data was well-organized and displayed correctly on the frontend.



```
57,
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
importProducts();
```

3: Steps Taken for Data Migration:

Exporting Data: The first step was exporting the data from Sanity CMS using the Sanity CLI. This ensured that all product data was safely backed up before any further operations.

```

import { createClient } from '@sanity/client';

const client = createClient({
  projectId: "jxzt9vz4",
  dataset: "production",
  useCdn: true,
  apiVersion: '2025-01-13',
  token: "skSa3jbhTSjBsTuiIpNHcRGp3dpTNFaZx8RGK0yP3Hotn20HFMwxPggtKtndzdHg1Kp864KVd6vd6ssyuHfDcz5HYydGs0GP3pf5Wv0aw61YwQ3UAUtlLbblCcGQZKR");

async function uploadImageToSanity(imageUrl) {
  try {
    console.log(`Uploading image: ${imageUrl}`);

    const response = await fetch(imageUrl);
    if (!response.ok) {
      throw new Error(`Failed to fetch image: ${imageUrl}`);
    }

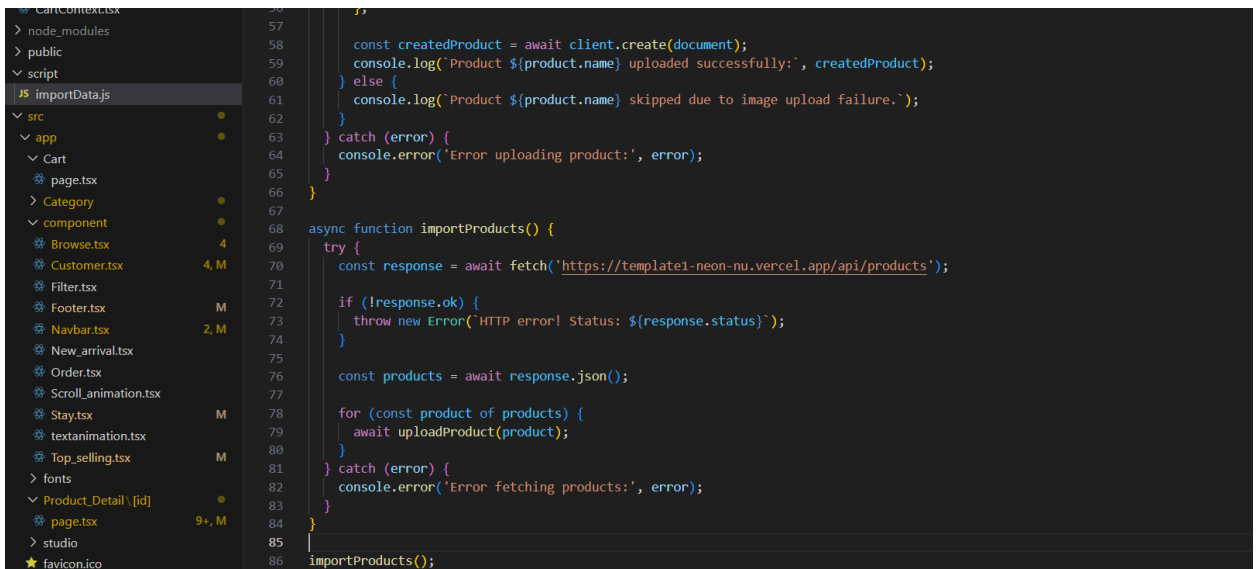
    const buffer = await response.arrayBuffer();
    const bufferImage = Buffer.from(buffer);

    const asset = await client.assets.upload('image', bufferImage, {
      filename: imageUrl.split('/').pop(),
    });

    console.log(`Image uploaded successfully: ${asset._id}`);
    return asset._id;
  } catch (error) {
    console.error('Failed to upload image:', imageUrl, error);
    return null;
  }
}

```

Verification of Data: The exported JSON structure was reviewed to ensure that all fields were populated correctly. This step ensured that the data would be accurately displayed when fetched and rendered on the frontend.

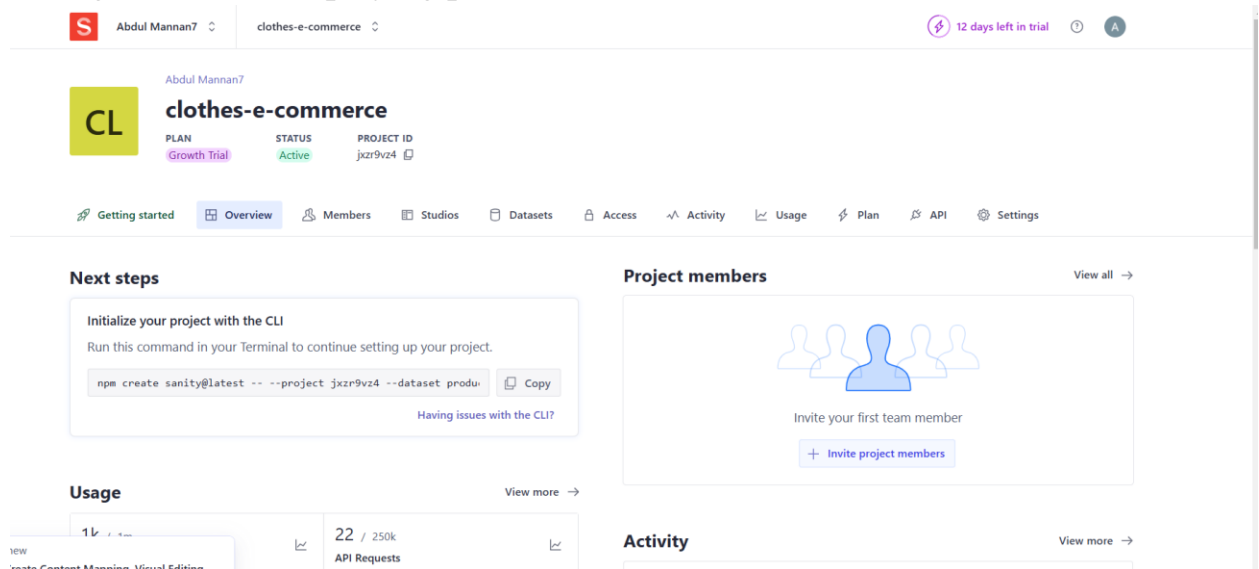


The screenshot shows a code editor with a file explorer on the left and code on the right. The file explorer lists files like `CartContext.tsx`, `node_modules`, `public`, `script`, `importData.js`, `src`, `app`, `Cart`, `page.tsx`, `Category`, `component`, `Browse.tsx`, `Customer.tsx`, `Filter.tsx`, `Footer.tsx`, `Navbar.tsx`, `New_arrival.tsx`, `Order.tsx`, `Scroll_animation.tsx`, `Stay.tsx`, `textanimation.tsx`, `Top_selling.tsx`, `fonts`, `Product_Detail\ [id]`, `page.tsx`, `studio`, and `favicon.ico`. The code on the right is as follows:

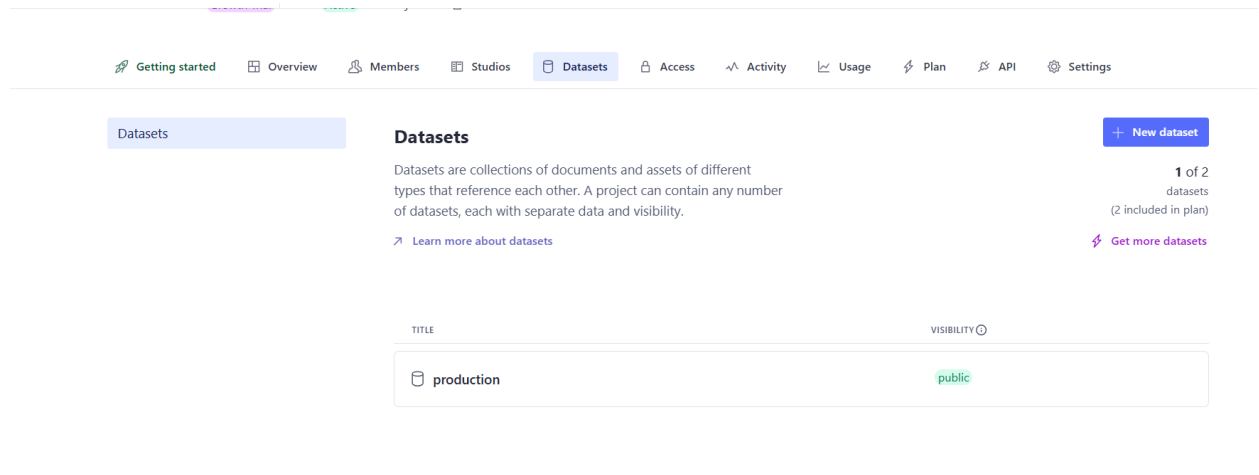
```

},
const createdProduct = await client.create(document);
console.log('Product ${product.name} uploaded successfully:', createdProduct);
} else {
  console.log('Product ${product.name} skipped due to image upload failure.');
```

Tools Used: Sanity Studio: Used for schema creation, content management, and displaying product data



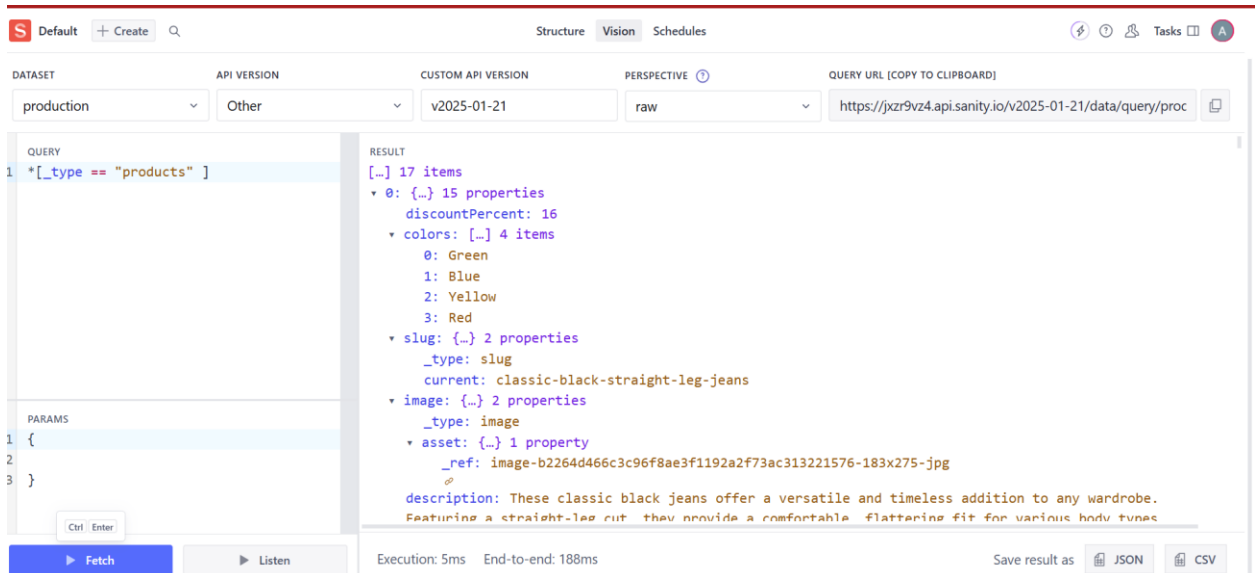
Sanity database :



Conclusion :

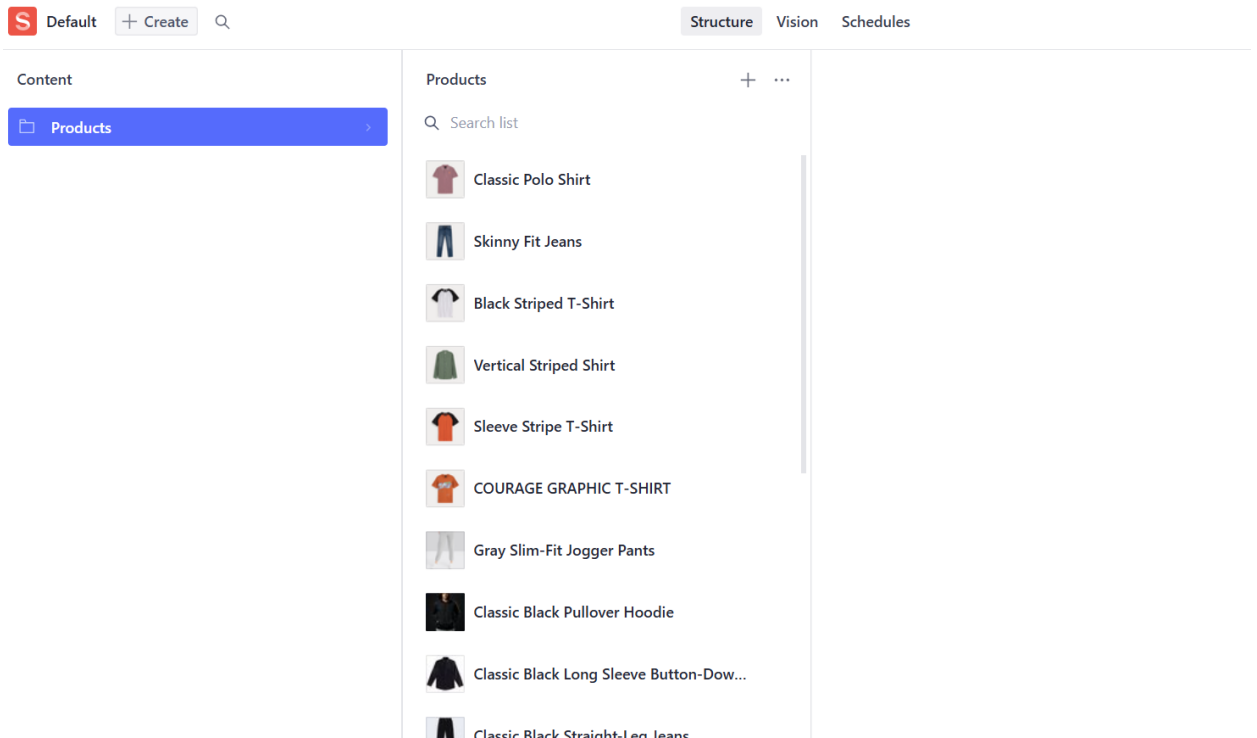
The API integration and data migration were completed successfully, making my e-commerce website more efficient and scalable. This setup made it easier to add and update product data, while the migration ensured all data stayed correct and consistent. Now, my website is more dynamic and simpler to manage.

Sanity vision :



The screenshot displays the Sanity Vision interface. At the top, there are tabs for 'Structure', 'Vision', and 'Schedules'. Below these, there are dropdown menus for 'DATASET' (set to 'production'), 'API VERSION' (set to 'Other'), 'CUSTOM API VERSION' (set to 'v2025-01-21'), and 'PERSPECTIVE' (set to 'raw'). A 'QUERY URL (COPY TO CLIPBOARD)' field shows the URL: 'https://jxzt9vz4.api.sanity.io/v2025-01-21/data/query/proc'. The main area is divided into two panels: 'QUERY' and 'RESULT'. The 'QUERY' panel shows a query: '1 *[_type == "products"]'. The 'RESULT' panel shows a JSON response with 17 items. The first item is expanded, showing properties like 'discountPercent: 16', 'colors: [Green, Blue, Yellow, Red]', 'slug: classic-black-straight-leg-jeans', and 'image: image-b2264d466c3c96f8ae3f1192a2f73ac313221576-183x275-jpg'. A description is also provided: 'These classic black jeans offer a versatile and timeless addition to any wardrobe. Featuring a straight-leg cut, they provide a comfortable, flattering fit for various body types.' At the bottom, there are buttons for 'Fetch' and 'Listen', and a status bar showing 'Execution: 5ms End-to-end: 188ms'. There are also buttons for 'Save result as' and 'JSON' and 'CSV'.

Sanity products :



Website Main products :

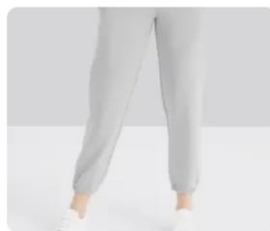
NEW ARRIVALS



Classic Black Straight-Leg Jeans

★★★★★ 4.5/5

170rs 16%



Gray Slim-Fit Jogger Pants

★★★★★ 4.5/5

170rs 15%



Skinny Fit Jeans

★★★★★ 4.5/5

240rs



Black Athletic Jogger Pants with Side Stripes

★★★★★ 4.5/5

180rs

View all

Top Selling



COURAGE GRAPHIC T-SHIRT
★★★★★ 4.5/5
145rs



Vertical Striped Shirt
★★★★★ 4.5/5
229rs 50%



LOOSE FIT BERMUDA SHORTS
★★★★★ 4.5/5
78rs 20%



Black Striped T-Shirt
★★★★★ 4.5/5
120rs

[View all](#)

Category page:

[Home](#) > [Casual](#)

Casual

Filters



T-shirts >

Shorts >

Shirts >

Hoodie >

Jeans >

Price



Colors



Gray Slim-Fit Jogger Pants
★★★★★ 3.5/5
170rs -15%

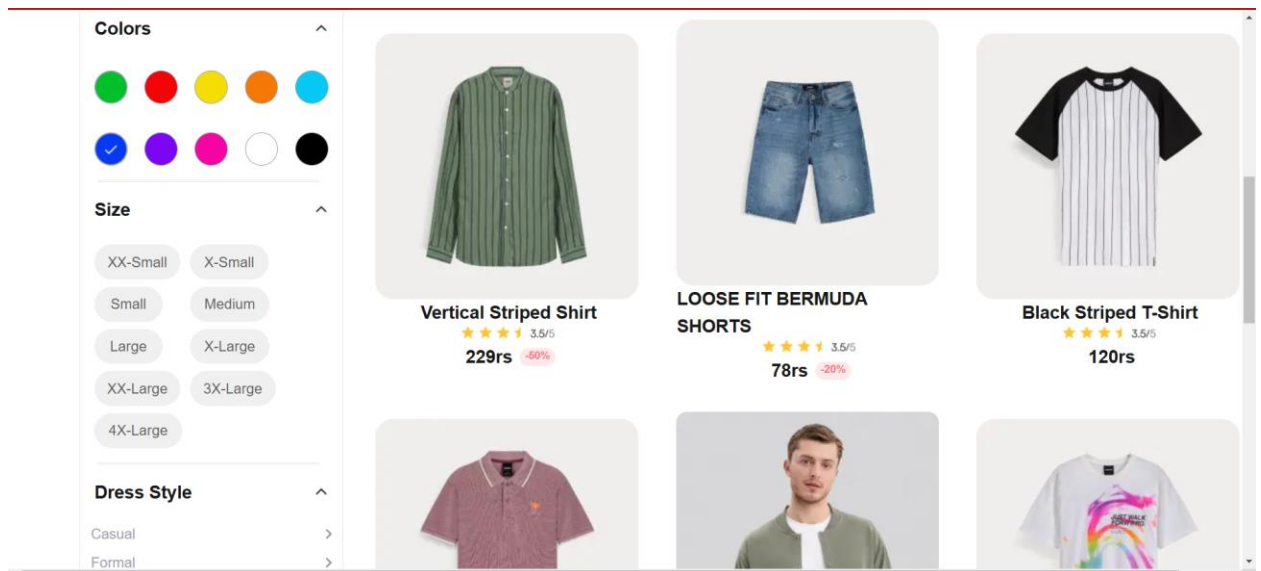


Skinny Fit Jeans
★★★★★ 3.5/5
240rs



COURAGE GRAPHIC T-SHIRT
★★★★★ 3.5/5
145rs





Product detail page:

[Home](#) > [Shop](#) > [Men](#) > [T-Shirt](#)



Casual Green Bomber Jacket

★★★★★ 4.5/5

300rs -20%

This graphic t-shirt is perfect for any occasion. Crafted from a soft and breathable fabric, it offers superior comfort and style.

Select Colors



Choose Size

S XXL XL L

- 1 +

Add to Cart