



API INTEGRATION REPORT - URBANSTITCH

Prepared By: Mustafa Qazi

Student ID: 00010241



First published: PKT 05:30 PM, January 18, 2025

Document Revision Information

Version	Date	Amendment	Author
1.0	11:30 PM, January 18, 2025	Initial release	Mustafa Qazi

Table of Contents

System Architecture Overview	2
Detailed API Requirements Document.....	3
Sanity Schemas Design	10
Screenshots of API Integration and Data Migration Process.....	13

1. System Architecture Overview

Frontend:

- Framework: **Next.js**
 - Server-Side Rendering (SSR) and Incremental Static Regeneration (ISR) for SEO and performance.
 - Dynamic routing for product categories and detail pages.
- Styling: **Tailwind CSS** for rapid UI development.
- State Management: **Context API** or for cart and user session handling.

Backend:

- API Routes: Built into Next.js to handle server-side logic.
 - `/api/products`: Fetch product data from **Sanity CMS**.
 - `/api/cart`: Manage cart operations (add, remove, update).
 - `/api/checkout`: Process payments using Stripe / or any other payment gateway
 - `/api/shipping`: Integrate with ShipEngine for shipping operations.
 - `/api/orders`: Save and retrieve order details.

2. Detailed API Requirements Document

Overview

- **Platform Name:** UrbanStitch Marketplace API
- **Frontend:** Next.js
- **Backend & Content Management:** Sanity CMS
- **Shipping Integration:** ShipEngine API
- **Payment Integration:** Stripe API
- **Authentication:** JSON Web Tokens (JWT) or API Key (based on user roles)
- **Base URL:**
 - Development & Production: vercel link here
- **Content Type:** JSON (application/json)

Feature	EndPoint	Method	Description	Parameters	Response Example
Authentication	/auth/login	POST	User Login	email, password, userid, name, role	<pre>{ "email": "user@example.com", "password": "password123" } Response (200): { "token": "eyJ2345exampletoken", "expires_in": 3600, "user": { "id": "u1", "name": "Ali Asghar", "email": "aliasghar@gmail.com", "role": "customer" } } Response (401): { "error": "Invalid email or password" }</pre>
Product Management	/products	GET	Get All Products	id, name, desc, price, category, size, colors, image, instock	<pre>Response (200): [{ "id": "p1", "name": "Casual T-Shirt", "description": "100% cotton unisex t-shirt", "price": 19.99, "category": "T-Shirts", "sizes": ["S", "M", "L", "XL"], "colors": ["White", "Black"],</pre>

					<pre> "image": "https://cdn.example.com/tshirt1.jpg", "in_stock": true }, { "id": "p2", "name": "Denim Jacket", "description": "Premium quality denim jacket", "price": 59.99, "category": "Jackets", "sizes": ["M", "L"], "colors": ["Blue"], "image": "https://cdn.example.com/jacket1.jpg", "in_stock": false }] </pre>
Product Management	/products/{id}	GET	Get Products by ID	id, name, desc, price, category, size, colors, image, instock	<ul style="list-style-type: none"> • Response (200): • { • "id": "p1", • "name": "Casual T-Shirt", • "description": "100% cotton unisex t-shirt", • "price": 19.99, • "category": "T-Shirts", • "sizes": ["S", "M", "L", "XL"], • "colors": ["White", "Black"], • "image": • "https://cdn.example.com/tshirt1.jpg", • "in_stock": true • }

Product Management	/products	POST	Create Product (Admin Only)	<p>Method: POST</p> <p>Request Body:</p> <pre>{ "name": "Leather Boots", "description": "Genuine leather boots for men and women", "price": 129.99, "category": "Footwear", "sizes": ["8", "9", "10", "11"], "colors": ["Brown", "Black"], "image": "https://cdn.example.com/boots1.jpg", "in_stock": true }</pre> <p>Response (201):</p> <pre>{ "message": "Product created successfully", "product": { "id": "p3", "name": "Leather Boots", "price": 129.99 } }</pre>
Category Management	/categories	GET	Get All Categories	<p>Response (200):</p> <pre>[{ "id": "c1", "name": "T-Shirts" }, { "id": "c2", "name": "Jackets" }, { "id": "c3", "name": "Footwear" }]</pre>

	/categories	POST	Create Category (Admin Only)		Request Body: { "name": "Accessories" } Response (201): { "message": "Category created successfully", "category": { "id": "c4", "name": "Accessories" } }
Cart Management	/cart/add	POST	Add to Cart		Request Body: { "userId": "u1", "productId": "p1", "quantity": 2 } Response (200): { "message": "Product added to cart successfully", "cart": { "userId": "u1", "items": [{ "productId": "p1", "quantity": 2 }] } }
Order Management	/orders	POST	Create Order		Request Body: { "userId": "u1", "products": [{ "productId": "p1", "quantity": 2 }, { "productId": "p2", "quantity": 1 }], "shippingAddress": { "line1": "123 Fashion St", "city": "Los Angeles", "state": "CA", "zip": "90001", "country": "US" } }

					Response (201): { "message": "Order created successfully", "order": { "id": "o1", "totalAmount": 99.97 } }
Payment Processing (Stripe API)	/payments/create-intent	POST	Create Payment Intent		Request Body: { "amount": 99.97, "currency": "USD" } Response (200): { "clientSecret": "pi_123456789_secret_abcdefgh" }

Shipping Management (Ship Engine API)	/shipping/rates	POST	Get Shipping Rates		<p>Request Body:</p> <pre>{ "from": { "postalCode": "90001", "countryCode": "US" }, "to": { "postalCode": "10001", "countryCode": "US" }, "weight": { "value": 2.5, "unit": "pounds" } }</pre> <p>Response (200):</p> <pre>[{ "carrier": "UPS", "service": "Ground", "rate": 12.99, "estimatedDeliveryDays": 3 }, { "carrier": "FedEx", "service": "Express", "rate": 24.99, "estimatedDeliveryDays": 1 }]</pre>
--	-----------------	------	--------------------	--	--

	/shipping/create-label	POST	Create Shipping Label		<pre>{ "shipment": { "fromAddress": { "line1": "Warehouse 1", "city": "Los Angeles", "state": "CA", "zip": "90001", "country": "US" }, "toAddress": { "line1": "123 Fashion St", "city": "New York", "state": "NY", "zip": "10001", "country": "US" }, "weight": { "value": 2.5, "unit": "pounds" }, "carrier": "UPS", "serviceCode": "ground" } }</pre> <p>Response (200):</p> <pre>{ "label": { "labelUrl": "https://cdn.example.com/shipping_label.pdf", "trackingNumber": "1Z12345E1234567890" } }</pre>
Error Handling			General Error Format		<p>Response (4xx/5xx):</p> <pre>{ "error": "Error description", "details": "Optional details about the error" }</pre>

3. Sanity Schemas Design

1. Product Schema

Defines the structure for products in the marketplace, including key attributes like name, price, category, sizes, and colors.

1. Product Schema

Field Name	Type	Title	Validation	Additional Options
name	string	Product Name	Required, max 100 characters	N/A
slug	slug	Slug	Required, max length 96	Source: name
description	text	Description	Required, max 500 characters	N/A
price	number	Price	Required, minimum 0	N/A
category	reference	Category	Required	References category type
sizes	array	Available Sizes	N/A	List: XS, S, M, L, XL, XXL
colors	array	Available Colors	N/A	Of type string
inStock	boolean	In Stock	N/A	Default value: true
images	array	Product Images	N/A	Of type image, hotspot enabled

2. Category Schema

Organizes products into categories, such as "T-Shirts," "Jackets," or "Footwear."

Field Name	Type	Title	Validation	Additional Options
name	string	Category Name	Required, max 50 characters	N/A
slug	slug	Slug	Required, max length 96	Source: name
description	text	Description	Max 200 characters	N/A

3. Customer Schema

Manages customer profiles, including contact information and addresses.

Field Name	Type	Title	Validation	Additional Options
name	string	Full Name	Required	N/A
email	string	Email	Required, must be a valid email	N/A
phone	string	Phone Number	Required, regex validation for phone numbers	Format: ^\+?\d{10,15}\$
addresses	array	Addresses	N/A	Array of objects with fields: line1, city, state, zip, country

4. Order Schema

Tracks customer orders, including product details, shipping, and payment status.

Field Name	Type	Title	Validation	Additional Options
orderId	string	Order ID	Required	N/A
customer	reference	Customer	Required	References customer type
products	array	Products	N/A	Array of objects with fields: product, quantity, price
totalAmount	number	Total Amount	Required, minimum 0	N/A
shippingAddress	object	Shipping Address	N/A	Fields: line1, city, state, zip, country
status	string	Order Status	N/A	Options: Pending, Processing, Shipped, Delivered, Canceled
paymentStatus	string	Payment Status	N/A	Options: Pending, Paid, Failed, Refunded

5. Review Schema

Stores customer reviews for products.

Field Name	Type	Title	Validation	Additional Options
product	reference	Product	Required	References product type
customer	reference	Customer	Required	References customer type
rating	number	Rating	Required, minimum 1, maximum 5	N/A

Field Name	Type	Title	Validation	Additional Options
reviewText	text	Review Text	Max 300 characters	N/A

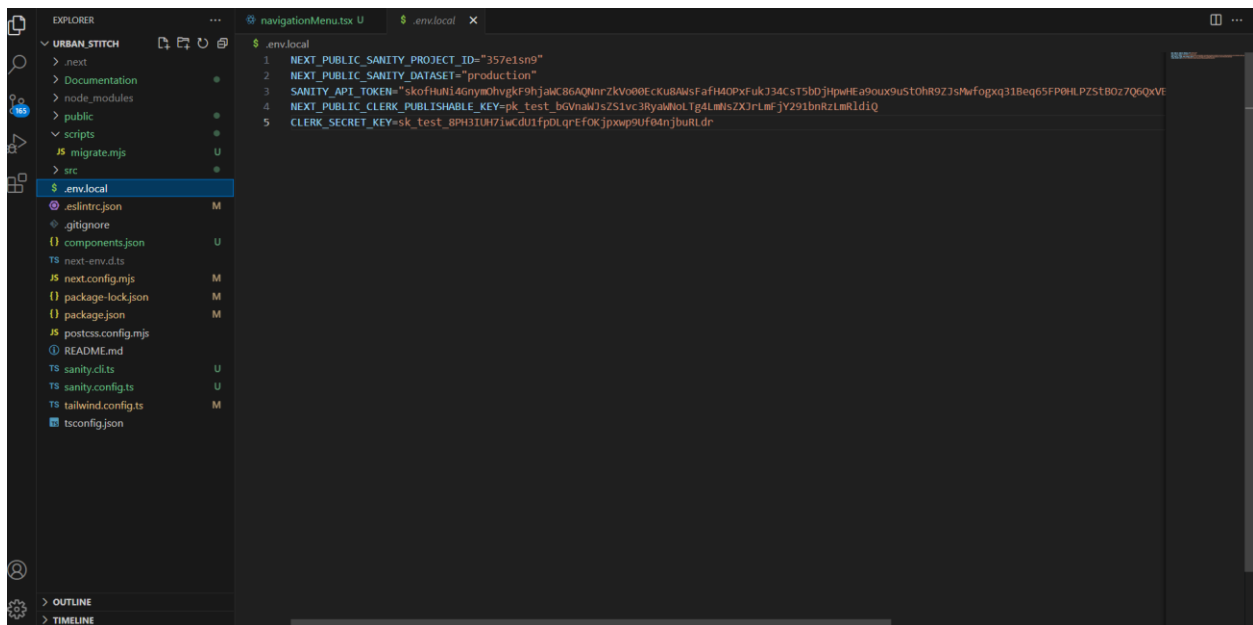
6. Banner Schema

For Homepage Promotions

Field Name	Type	Title	Validation	Additional Options
title	string	Title	Required	N/A
image	image	Banner Image	N/A	Hotspot enabled
link	url	Redirect Link	N/A	N/A

Screenshots of API Integration and Data Migration Process

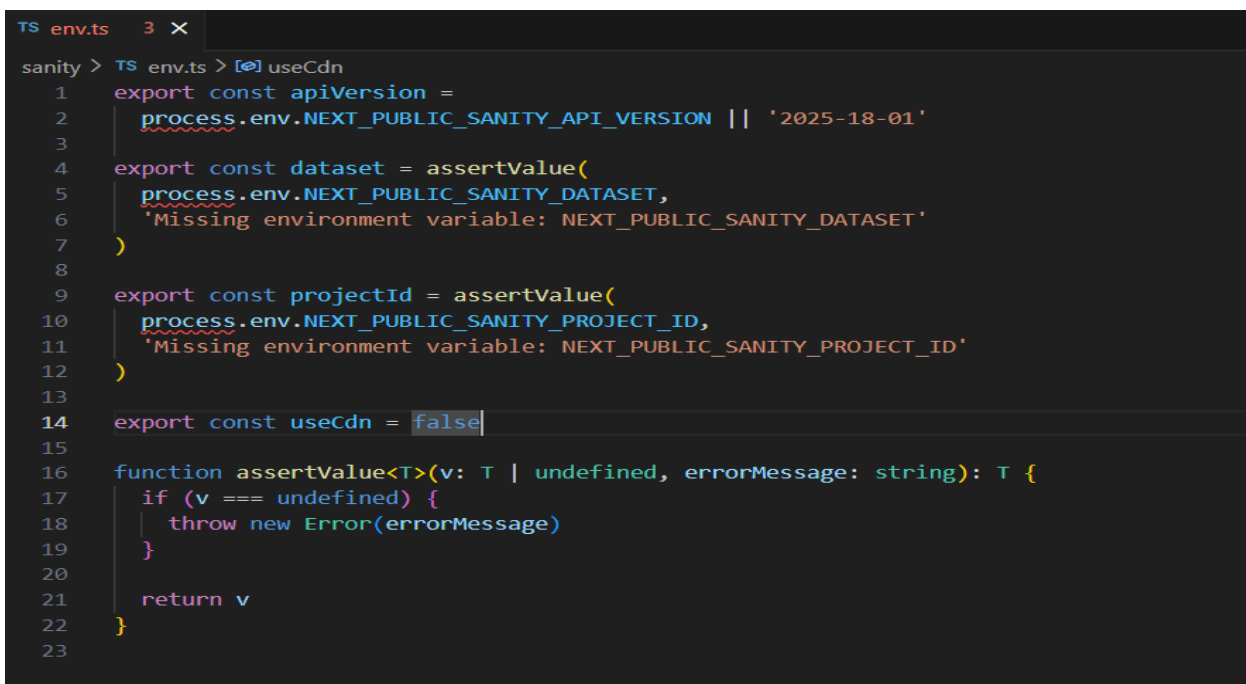
Use .env file



The screenshot shows a VS Code editor with the Explorer sidebar on the left. The file explorer shows a project structure with folders like .next, Documentation, node_modules, public, scripts, and migrate.mjs. The .env.local file is selected and its content is displayed in the main editor area. The file contains five lines of environment variables for Next.js and Sanity.

```
1 NEXT_PUBLIC_SANITY_PROJECT_ID="357e1sn9"  
2 NEXT_PUBLIC_SANITY_DATASET="production"  
3 SANITY_API_TOKEN="skofhuti4gnyrchvgkf9hjawk86AQnnrZkV000Ecku8AksFafH4DPxFukJ34Cs15bDjHpwHEa9oux9ust0HR9ZJ3sMwfgpq31Beq65FP0HLpZStB0z7Q6QxVE"  
4 NEXT_PUBLIC_CLERK_PUBLISHABLE_KEY=pk_test_b6VnawJsz51vc3RyamWot1g4Lmhs2XJrLmf_jY291bnRzLmRldiQ  
5 CLERK_SECRET_KEY=sk_test_8PH31UH71wcdU1fp0LqrEF0Kjpxw9Uf04nJbuRLdr
```

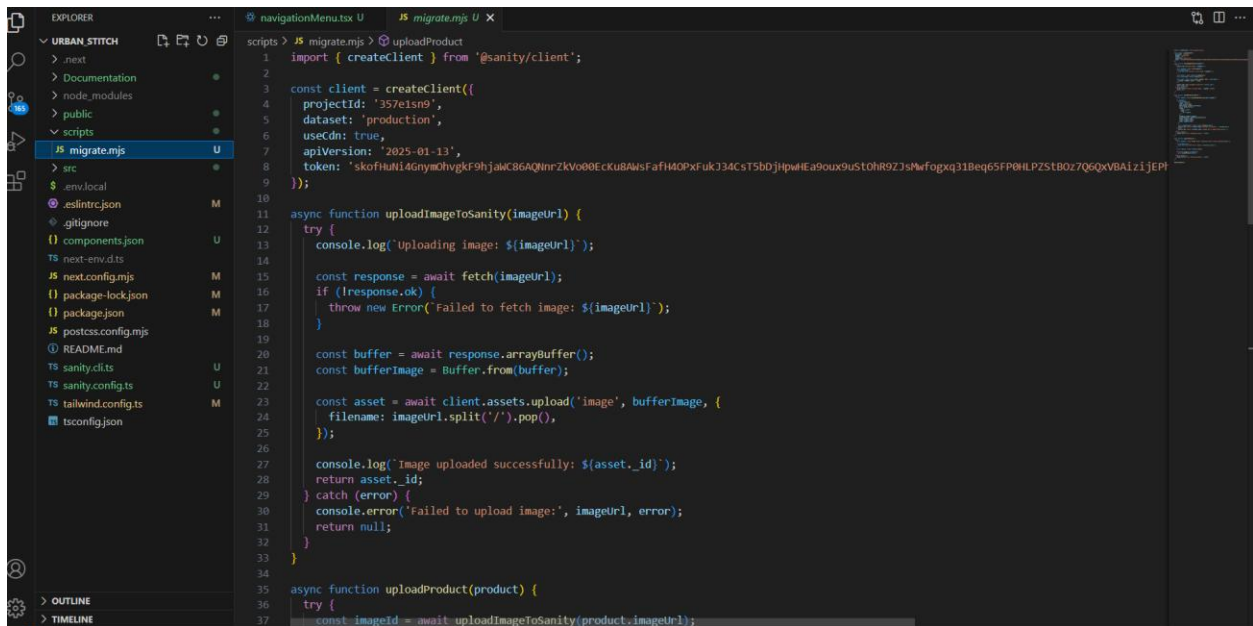
Migration code for sanity



The screenshot shows a VS Code editor with a file named env.ts. The code defines environment variables for Sanity and a utility function for asserting values. The variables are NEXT_PUBLIC_SANITY_API_VERSION, NEXT_PUBLIC_SANITY_DATASET, NEXT_PUBLIC_SANITY_PROJECT_ID, and useCdn. The useCdn variable is currently set to false.

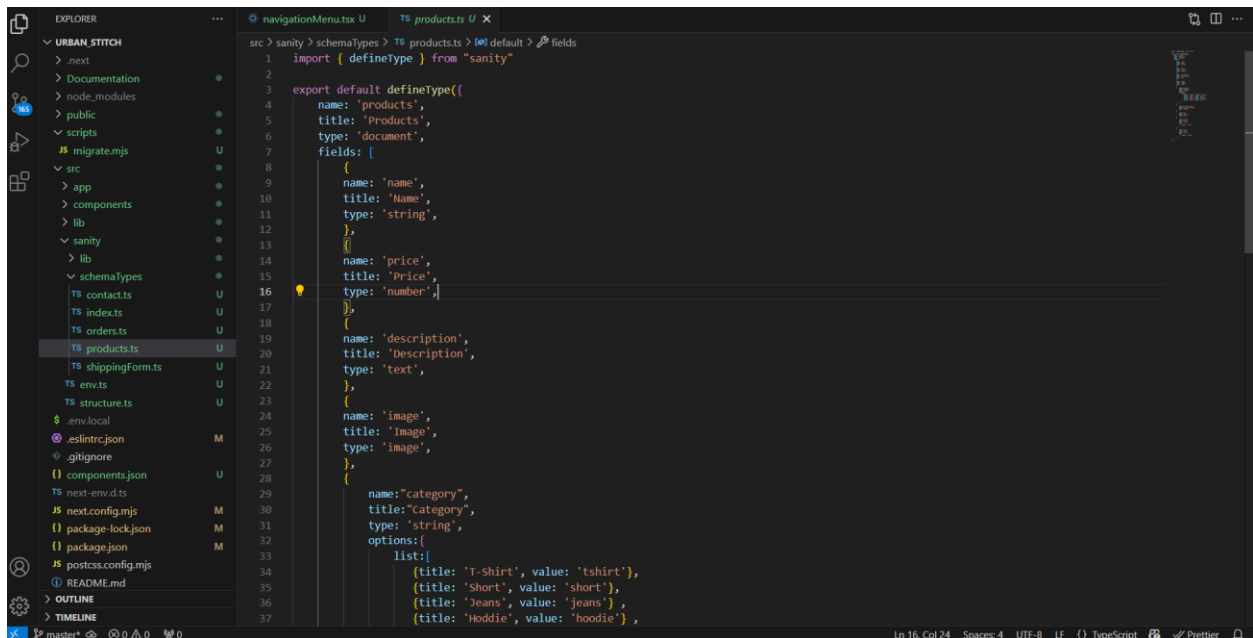
```
1 export const apiVersion =  
2   process.env.NEXT_PUBLIC_SANITY_API_VERSION || '2025-18-01'  
3  
4 export const dataset = assertValue(  
5   process.env.NEXT_PUBLIC_SANITY_DATASET,  
6   'Missing environment variable: NEXT_PUBLIC_SANITY_DATASET'  
7 )  
8  
9 export const projectId = assertValue(  
10  process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,  
11  'Missing environment variable: NEXT_PUBLIC_SANITY_PROJECT_ID'  
12 )  
13  
14 export const useCdn = false  
15  
16 function assertValue<T>(v: T | undefined, errorMessage: string): T {  
17   if (v === undefined) {  
18     throw new Error(errorMessage)  
19   }  
20  
21   return v  
22 }  
23
```

Migrate.mjs file for sanity data migration



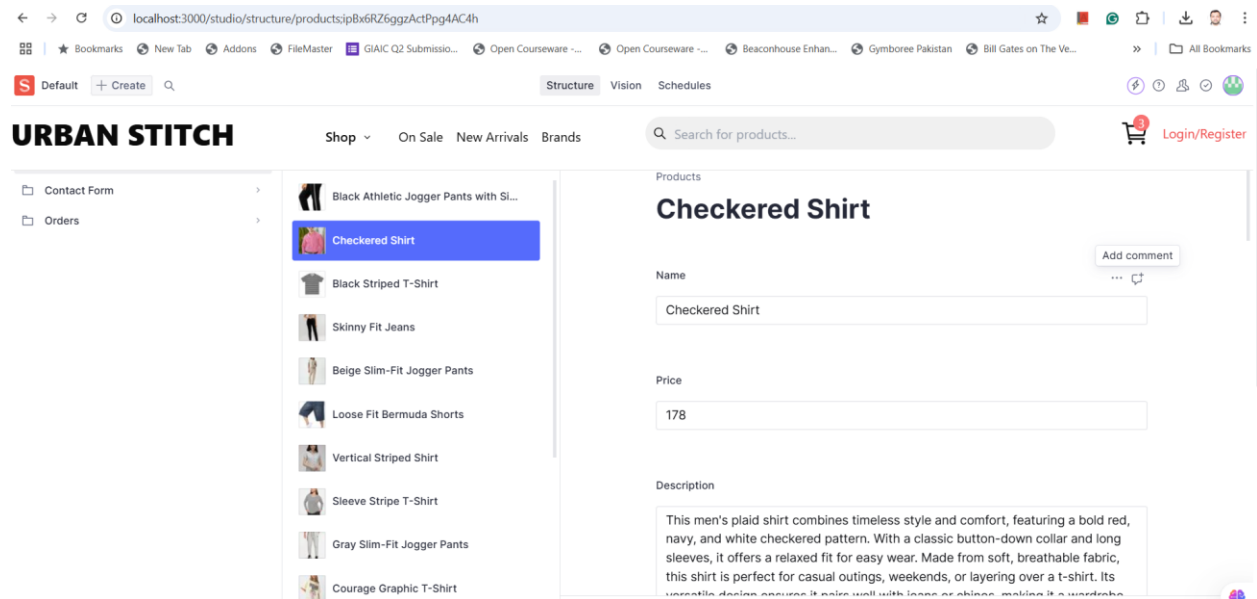
```
1 import { createClient } from '@sanity/client';
2
3 const client = createClient({
4   projectId: '357e1sn9',
5   dataset: 'production',
6   useCdn: true,
7   apiVersion: '2025-01-13',
8   token: 'skofHuH14GnymOhvgkF9hjdwC86AQInrZkVo00Ecku8Aw5FaTH4OPxFukC34CsT5bDjHpwHEa9oux9ustLOHR9ZJ3sWf0gqx31Beq65FP0HLPZStBOz7Q6QxVBA1ziJEPH',
9 });
10
11 async function uploadImageToSanity(imageUrl) {
12   try {
13     console.log('uploading image: ${imageUrl}');
14
15     const response = await fetch(imageUrl);
16     if (!response.ok) {
17       throw new Error('Failed to fetch image: ${imageUrl}');
18     }
19
20     const buffer = await response.arrayBuffer();
21     const bufferImage = Buffer.from(buffer);
22
23     const asset = await client.assets.upload('image', bufferImage, {
24       filename: imageUrl.split('/').pop(),
25     });
26
27     console.log('Image uploaded successfully: ${asset._id}');
28     return asset._id;
29   } catch (error) {
30     console.error('Failed to upload image:', imageUrl, error);
31     return null;
32   }
33 }
34
35 async function uploadProduct(product) {
36   try {
37     const imageId = await uploadImageToSanity(product.imageUrl);
```

./products api fetch from sanity



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

Products being displayed in sanity cms



GROQ Query

