Hafiz Muhammad Umar

424905

Friday 9-12 (Morning)

API Integration Report - [HEKTO WEBSITE]

Reviewed API Documentation:

- I carefully read the provided API documentation for my assigned template to understand the available endpoint (/products).
- I identified the structure of the data returned by the API, including field names and data types.

Set Up API Calls:

- I used Thunder client to test the API endpoint and ensure the data was being returned correctly. I created utility functions in my Next.js project to fetch data from the API.
- I used fetch to make GET requests to the API endpoints and stored the responses in variables. I logged the API responses in the console to verify the data structure.

Compared API Data with Sanity Schema:

- I reviewed the API data structure and compared it with the existing schema in Sanity CMS. I identified mismatches in field names and data types.
- I updated the Sanity schema to match the API data structure. For example:
  - API Field: product_title → Sanity Field: name
  - API Field: price → Sanity Field: price (with proper data type)

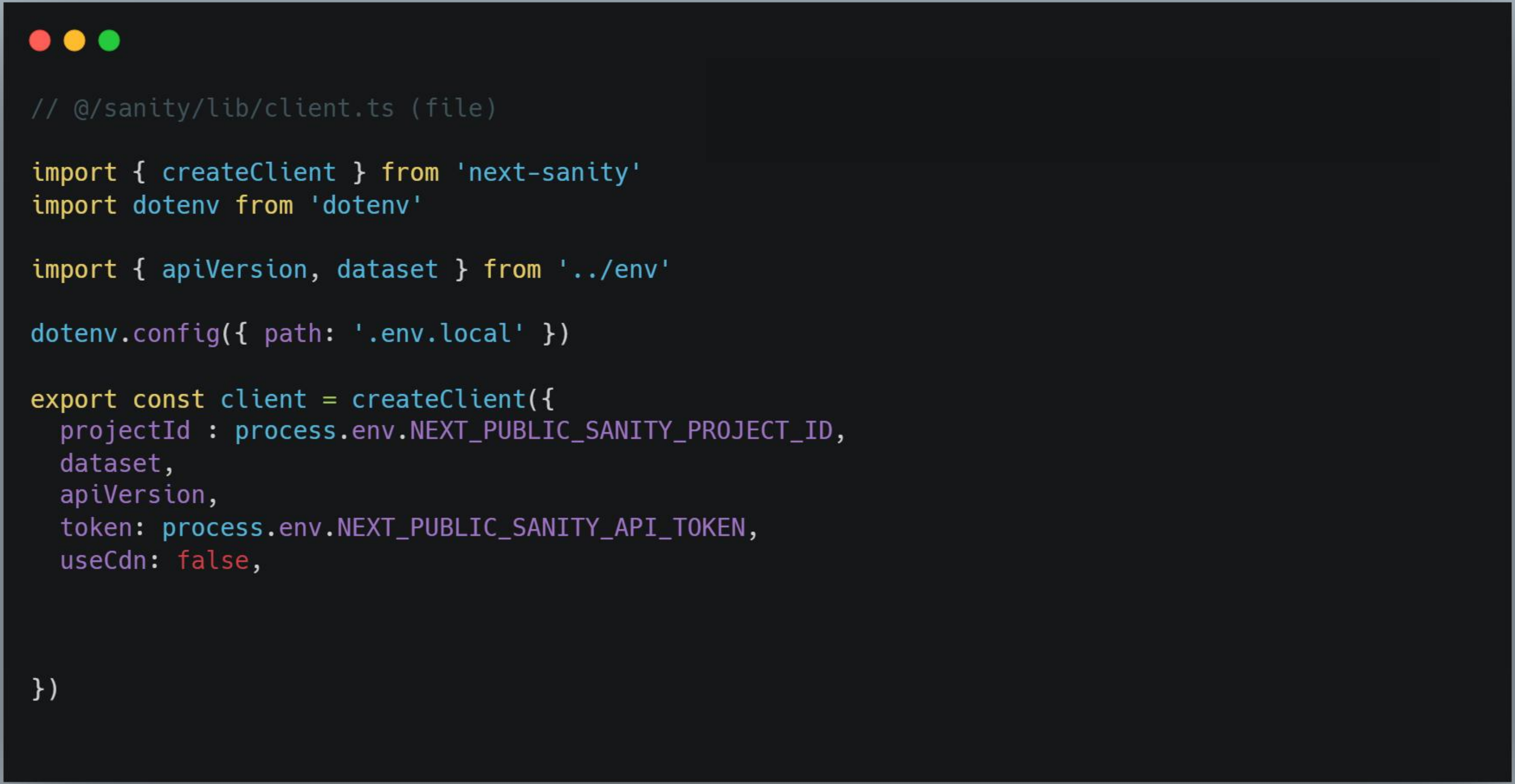
I added new fields in Sanity CMS to accommodate additional data from the API , because the API is not enough to complete my website products and their details .

To migrate data from the API to Sanity CMS, I followed these steps:

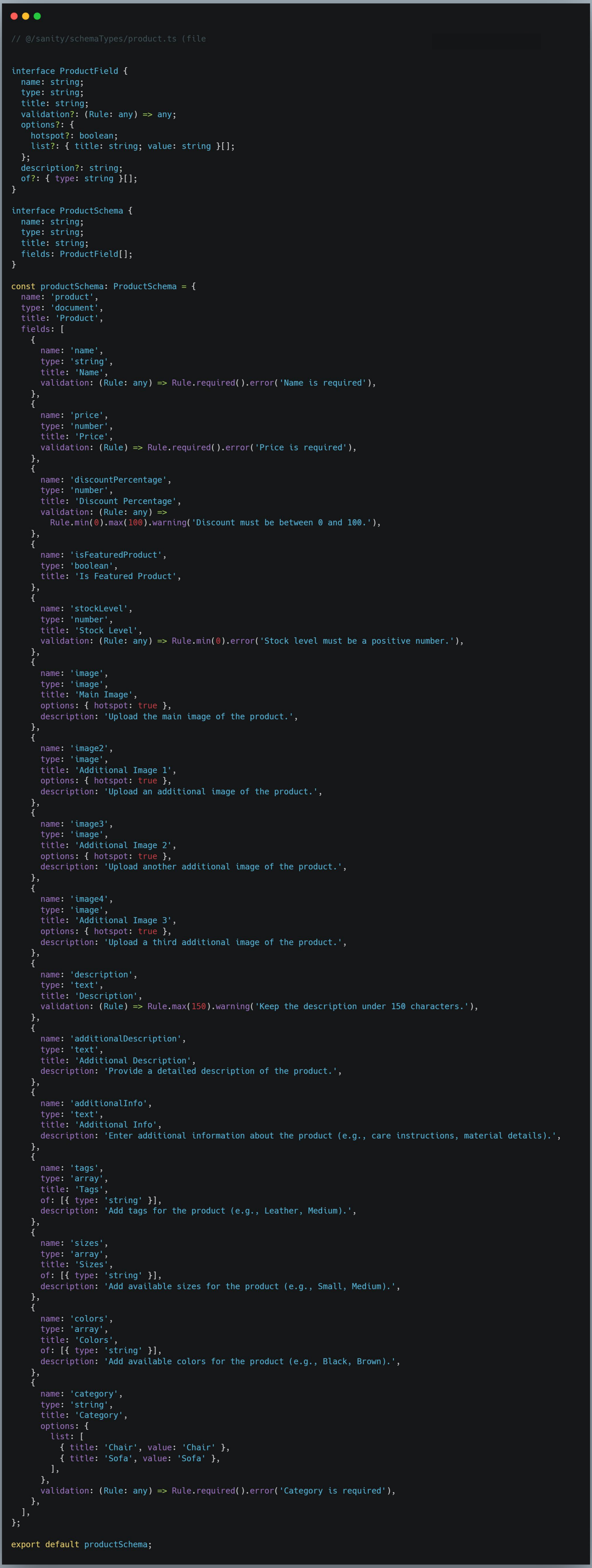
- I decided to use the provided API to fetch data and write a script to import it into Sanity CMS.
- I created a script Folder and then i created a migration (.mjs) file to fetch data from the API and transform it into the format required by Sanity CMS.
- I used the Sanity client library to upload the data to the CMS. I ran the migration script to import product data, categories, and other relevant information into Sanity CMS.
- I verified the imported data by checking the Sanity dashboard and ensuring all fields were correctly populated.

In this project, I successfully integrated the provided API into my Next.js frontend and migrated data into Sanity CMS. I adjusted the schema to match the API data structure and ensured the data was accurately displayed in the frontend. This exercise helped me gain practical experience in API integration, data migration, and schema validation, which are essential skills for building scalable marketplaces.

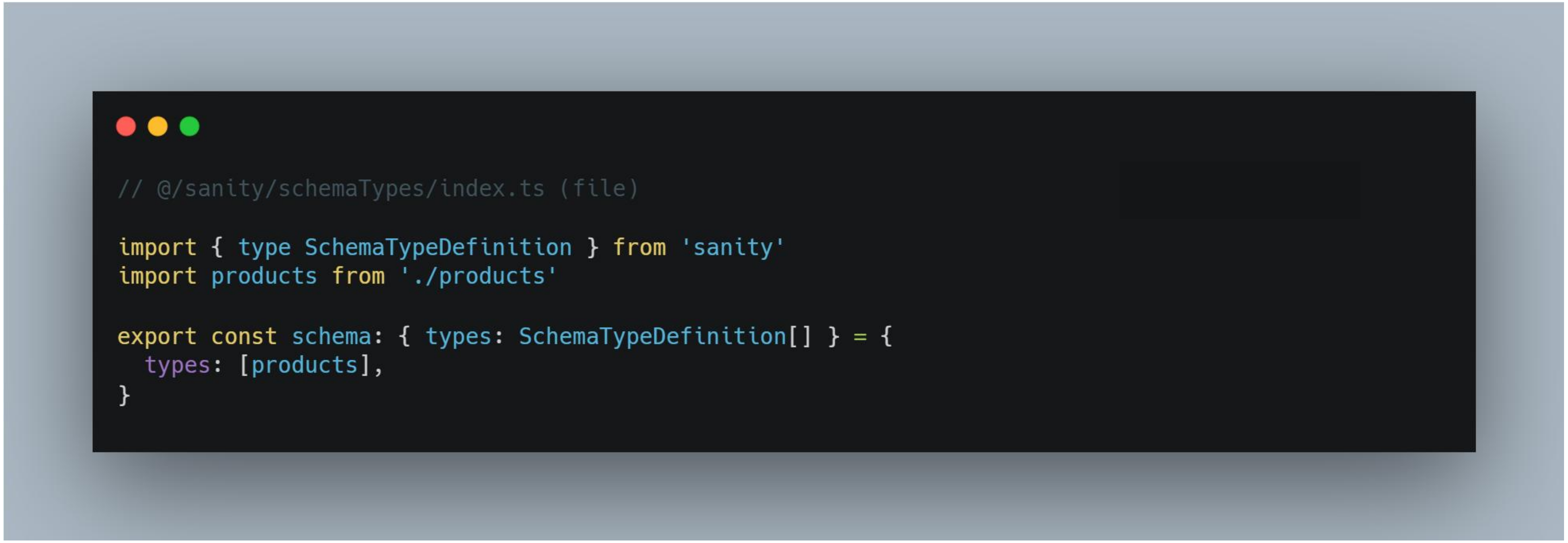
Task	**Status**
API Understanding	✓
Schema Validation	✓
Data Migration	✓
API Integration in Next.js	✓
Submission Preparation	✓



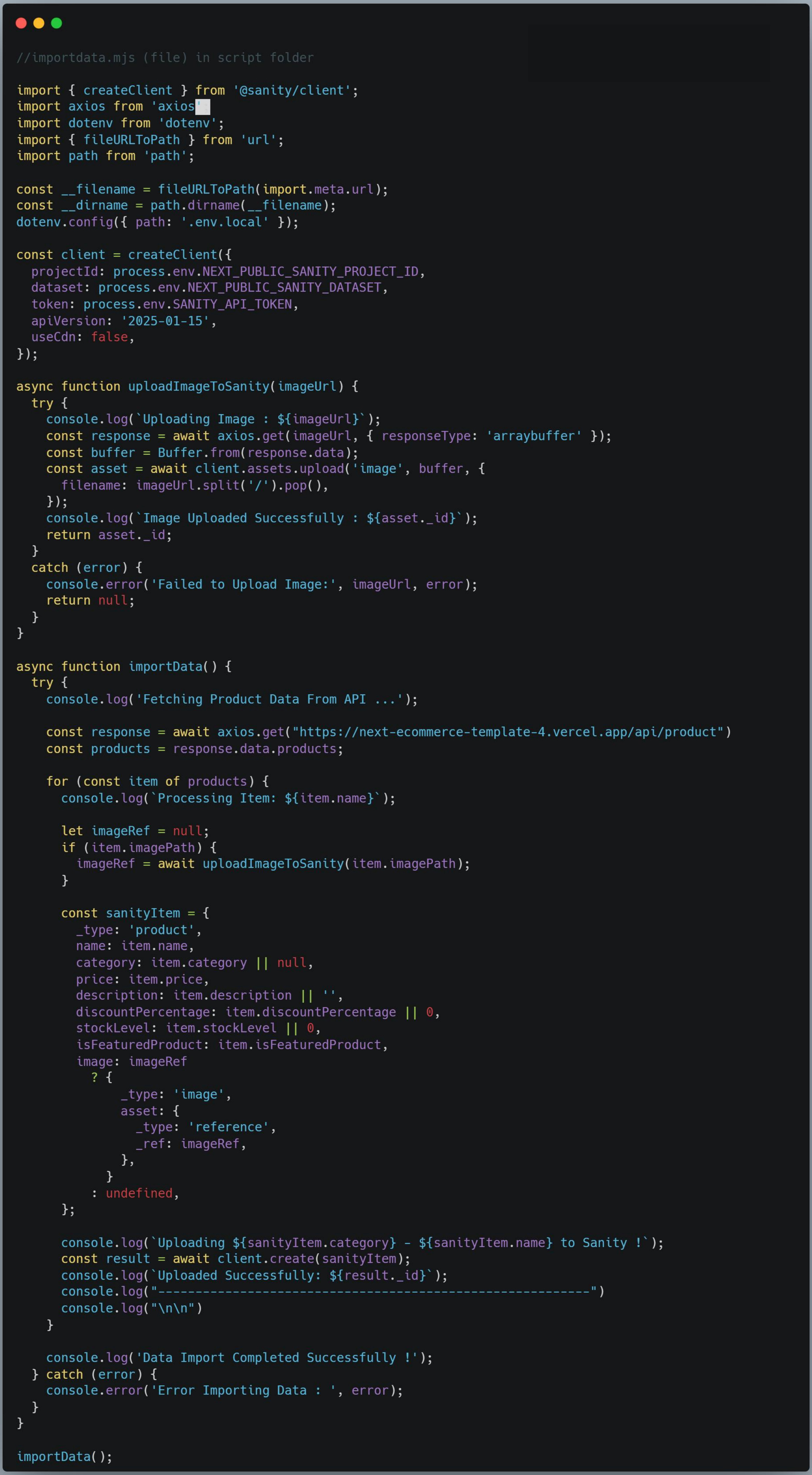
```
// @/sanity/lib/client.ts (file)

import { createClient } from 'next-sanity'
import dotenv from 'dotenv'

import { apiVersion, dataset } from '../env'

dotenv.config({ path: '.env.local' })

export const client = createClient({
  projectId : process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
  dataset,
  apiVersion,
  token: process.env.NEXT_PUBLIC_SANITY_API_TOKEN,
  useCdn: false,



})
```



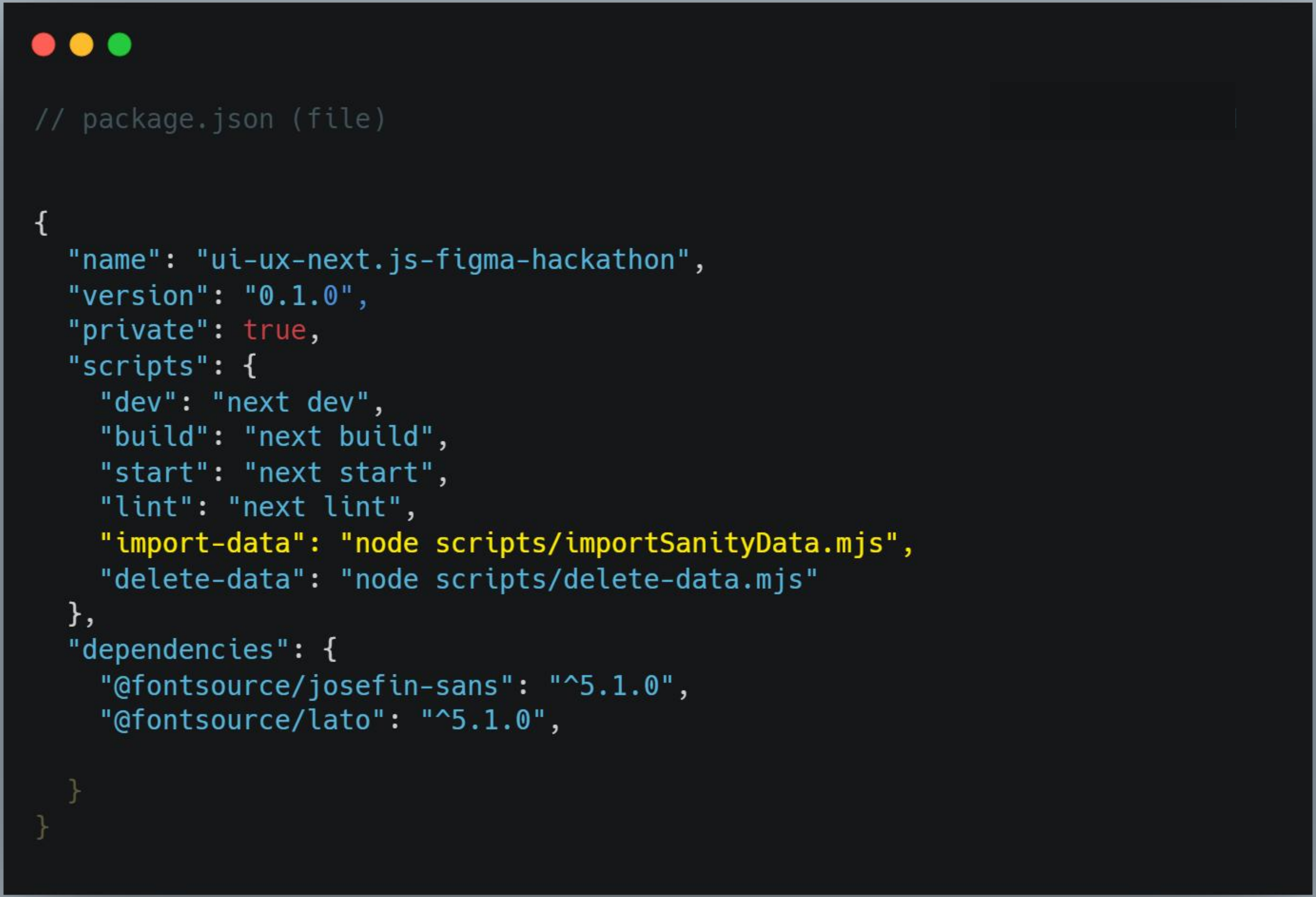
```ts
// @/sanity/schemaTypes/product.ts (file


interface ProductField {
  name: string;
  type: string;
  title: string;
  validation?: (Rule: any) => any;
  options?: {
    hotspot?: boolean;
    list?: { title: string; value: string }[];
  };
  description?: string;
  of?: { type: string }[];
}

interface ProductSchema {
  name: string;
  type: string;
  title: string;
  fields: ProductField[];
}

const productSchema: ProductSchema = {
  name: 'product',
  type: 'document',
  title: 'Product',
  fields: [
    {
      name: 'name',
      type: 'string',
      title: 'Name',
      validation: (Rule: any) => Rule.required().error('Name is required'),
    },
    {
      name: 'price',
      type: 'number',
      title: 'Price',
      validation: (Rule) => Rule.required().error('Price is required'),
    },
    {
      name: 'discountPercentage',
      type: 'number',
      title: 'Discount Percentage',
      validation: (Rule: any) =>
        Rule.min(0).max(100).warning('Discount must be between 0 and 100.'),
    },
    {
      name: 'isFeaturedProduct',
      type: 'boolean',
      title: 'Is Featured Product',
    },
    {
      name: 'stockLevel',
      type: 'number',
      title: 'Stock Level',
      validation: (Rule: any) => Rule.min(0).error('Stock level must be a positive number.'),
    },
    {
      name: 'image',
      type: 'image',
      title: 'Main Image',
      options: { hotspot: true },
      description: 'Upload the main image of the product.',
    },
    {
      name: 'image2',
      type: 'image',
      title: 'Additional Image 1',
      options: { hotspot: true },
      description: 'Upload an additional image of the product.',
    },
    {
      name: 'image3',
      type: 'image',
      title: 'Additional Image 2',
      options: { hotspot: true },
      description: 'Upload another additional image of the product.',
    },
    {
      name: 'image4',
      type: 'image',
      title: 'Additional Image 3',
      options: { hotspot: true },
      description: 'Upload a third additional image of the product.',
    },
    {
      name: 'description',
      type: 'text',
      title: 'Description',
      validation: (Rule) => Rule.max(150).warning('Keep the description under 150 characters.'),
    },
    {
      name: 'additionalDescription',
      type: 'text',
      title: 'Additional Description',
      description: 'Provide a detailed description of the product.',
    },
    {
      name: 'additionalInfo',
      type: 'text',
      title: 'Additional Info',
      description: 'Enter additional information about the product (e.g., care instructions, material details).',
    },
    {
      name: 'tags',
      type: 'array',
      title: 'Tags',
      of: [{ type: 'string' }],
      description: 'Add tags for the product (e.g., Leather, Medium).',
    },
    {
      name: 'sizes',
      type: 'array',
      title: 'Sizes',
      of: [{ type: 'string' }],
      description: 'Add available sizes for the product (e.g., Small, Medium).',
    },
    {
      name: 'colors',
      type: 'array',
      title: 'Colors',
      of: [{ type: 'string' }],
      description: 'Add available colors for the product (e.g., Black, Brown).',
    },
    {
      name: 'category',
      type: 'string',
      title: 'Category',
      options: {
        list: [
          { title: 'Chair', value: 'Chair' },
          { title: 'Sofa', value: 'Sofa' },
        ],
      },
      validation: (Rule: any) => Rule.required().error('Category is required'),
    },
  ],
};

export default productSchema;
```



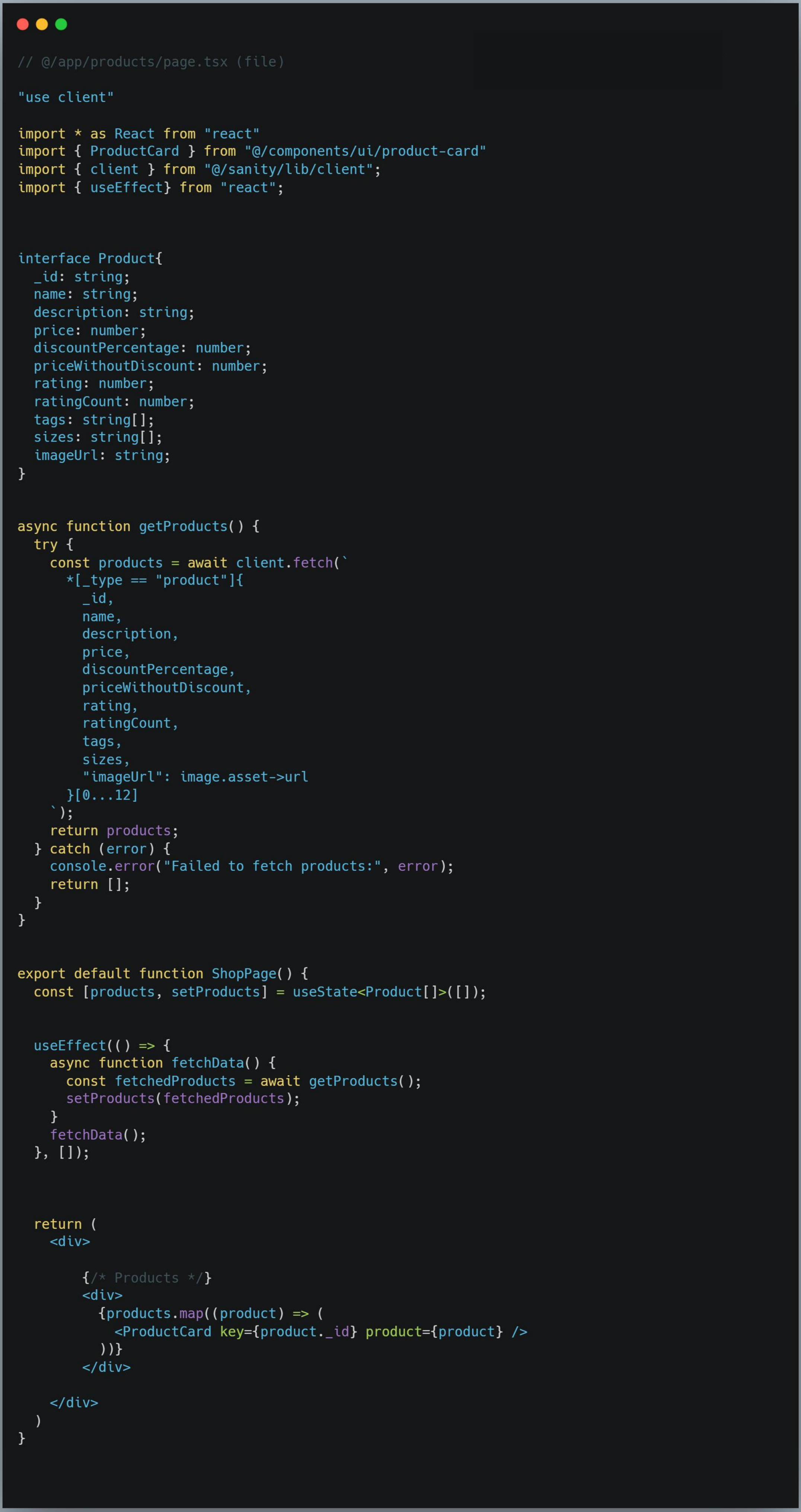
```ts
// @/sanity/schemaTypes/index.ts (file)

import { type SchemaTypeDefinition } from 'sanity'
import products from './products'

export const schema: { types: SchemaTypeDefinition[] } = {
  types: [products],
}
```



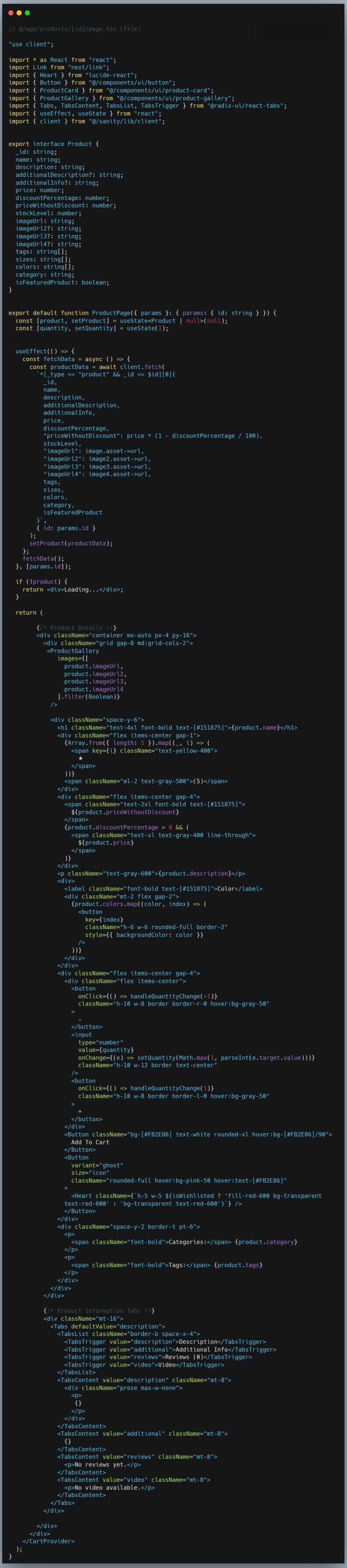
```javascript
//importdata.mjs (file) in script folder

import { createClient } from '@sanity/client';
import axios from 'axios';
import dotenv from 'dotenv';
import { fileURLToPath } from 'url';
import path from 'path';

const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);
dotenv.config({ path: '.env.local' });

const client = createClient({
  projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
  dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
  token: process.env.SANITY_API_TOKEN,
  apiVersion: '2025-01-15',
  useCdn: false,
});

async function uploadImageToSanity(imageUrl) {
  try {
    console.log(`Uploading Image : ${imageUrl}`);
    const response = await axios.get(imageUrl, { responseType: 'arraybuffer' });
    const buffer = Buffer.from(response.data);
    const asset = await client.assets.upload('image', buffer, {
      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;
  }
}

async function importData() {
  try {
    console.log('Fetching Product Data From API ...');

    const response = await axios.get("https://next-ecommerce-template-4.vercel.app/api/product")
    const products = response.data.products;

    for (const item of products) {
      console.log(`Processing Item: ${item.name}`);

      let imageRef = null;
      if (item.imagePath) {
        imageRef = await uploadImageToSanity(item.imagePath);
      }

      const sanityItem = {
        _type: 'product',
        name: item.name,
        category: item.category || null,
        price: item.price,
        description: item.description || '',
        discountPercentage: item.discountPercentage || 0,
        stockLevel: item.stockLevel || 0,
        isFeaturedProduct: item.isFeaturedProduct,
        image: imageRef
          ? {
              _type: 'image',
              asset: {
                _type: 'reference',
                _ref: imageRef,
              },
            }
          : undefined,
      };

      console.log(`Uploading ${sanityItem.category} - ${sanityItem.name} to Sanity !`);
      const result = await client.create(sanityItem);
      console.log(`Uploaded Successfully: ${result._id}`);
      console.log("----------------------------------------------------------")
      console.log("\n\n")
    }

    console.log('Data Import Completed Successfully !');
  } catch (error) {
    console.error('Error Importing Data : ', error);
  }
}

importData();
```



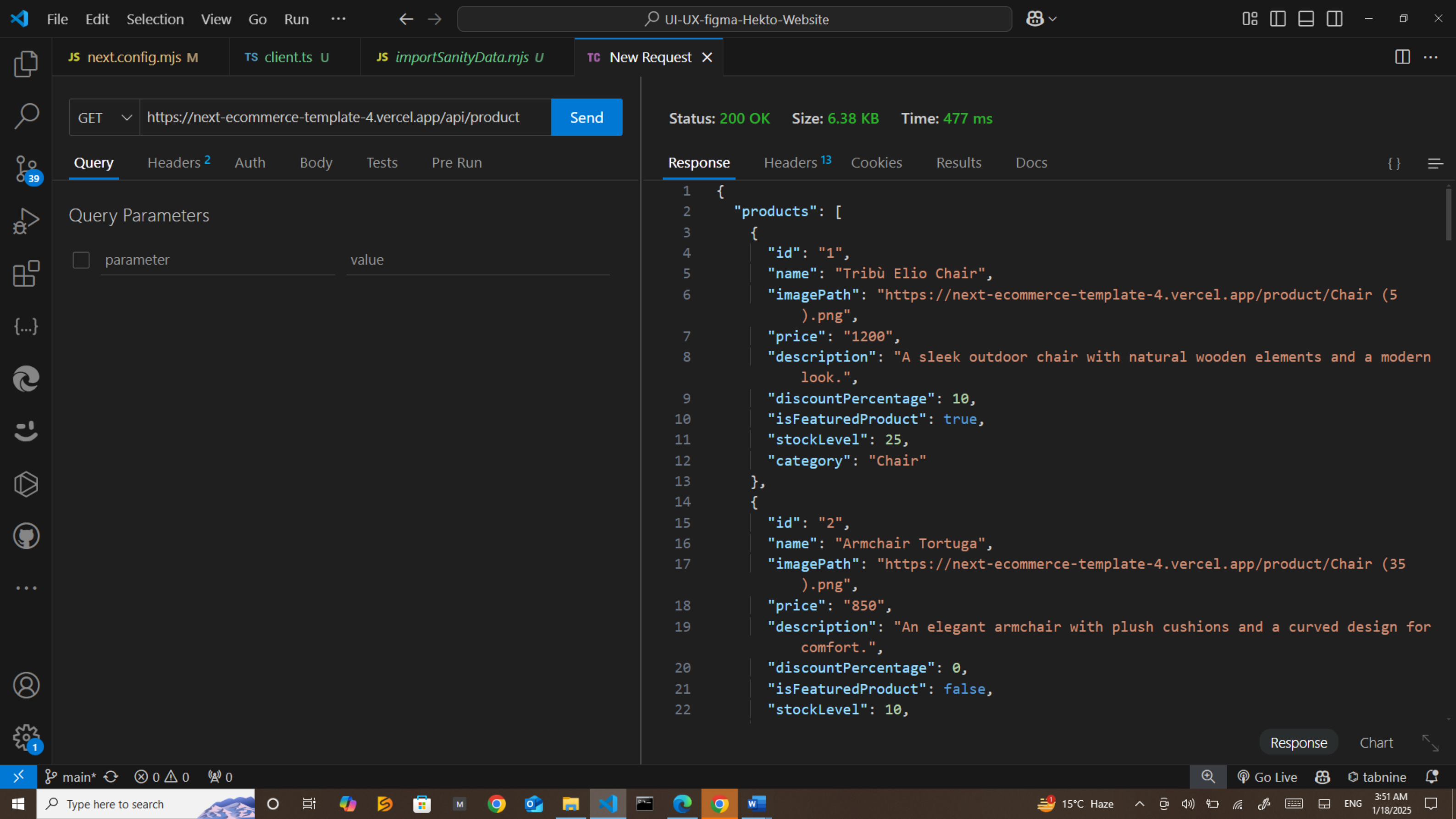
```
// package.json (file)


{
  "name": "ui-ux-next.js-figma-hackathon",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint",
    "import-data": "node scripts/importSanityData.mjs",
    "delete-data": "node scripts/delete-data.mjs"
  },
  "dependencies": {
    "@fontsource/josefin-sans": "^5.1.0",
    "@fontsource/lato": "^5.1.0",

  }
}
```



```tsx
// @/app/products/page.tsx (file)

"use client"

import * as React from "react"
import { ProductCard } from "@/components/ui/product-card"
import { client } from "@/sanity/lib/client";
import { useEffect} from "react";



interface Product{
  _id: string;
  name: string;
  description: string;
  price: number;
  discountPercentage: number;
  priceWithoutDiscount: number;
  rating: number;
  ratingCount: number;
  tags: string[];
  sizes: string[];
  imageUrl: string;
}


async function getProducts() {
  try {
    const products = await client.fetch(`
      *[_type == "product"]{
        _id,
        name,
        description,
        price,
        discountPercentage,
        priceWithoutDiscount,
        rating,
        ratingCount,
        tags,
        sizes,
        "imageUrl": image.asset->url
      }[0...12]
    `);
    return products;
  } catch (error) {
    console.error("Failed to fetch products:", error);
    return [];
  }
}


export default function ShopPage() {
  const [products, setProducts] = useState<Product[]>([]);


  useEffect(() => {
    async function fetchData() {
      const fetchedProducts = await getProducts();
      setProducts(fetchedProducts);
    }
    fetchData();
  }, []);



  return (
    <div>

        {/* Products */}
        <div>
          {products.map((product) => (
            <ProductCard key={product._id} product={product} />
          ))}
        </div>

    </div>
  )
}
```



```tsx
// @/app/products/[id]/page.tsx (file)

"use client";

import * as React from "react";
import Link from "next/link";
import { Heart } from "lucide-react";
import { Button } from "@/components/ui/button";
import { ProductCard } from "@/components/ui/product-card";
import { ProductGallery } from "@/components/ui/product-gallery";
import { Tabs, TabsContent, TabsList, TabsTrigger } from "@radix-ui/react-tabs";
import { useEffect, useState } from "react";
import { client } from "@/sanity/lib/client";


export interface Product {
  _id: string;
  name: string;
  description: string;
  additionalDescription?: string;
  additionalInfo?: string;
  price: number;
  discountPercentage: number;
  priceWithoutDiscount: number;
  stockLevel: number;
  imageUrl: string;
  imageUrl2?: string;
  imageUrl3?: string;
  imageUrl4?: string;
  tags: string[];
  sizes: string[];
  colors: string[];
  category: string;
  isFeaturedProduct: boolean;
}


export default function ProductPage({ params }: { params: { id: string } }) {
  const [product, setProduct] = useState<Product | null>(null);
  const [quantity, setQuantity] = useState(1);


  useEffect(() => {
    const fetchData = async () => {
      const productData = await client.fetch(
        `*[_type == "product" && _id == $id][0]{
          _id,
          name,
          description,
          additionalDescription,
          additionalInfo,
          price,
          discountPercentage,
          "priceWithoutDiscount": price * (1 - discountPercentage / 100),
          stockLevel,
          "imageUrl": image.asset->url,
          "imageUrl2": image2.asset->url,
          "imageUrl3": image3.asset->url,
          "imageUrl4": image4.asset->url,
          tags,
          sizes,
          colors,
          category,
          isFeaturedProduct
        }`,
        { id: params.id }
      );
      setProduct(productData);
    };
    fetchData();
  }, [params.id]);

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

  return (

        {/* Product Details */}
        <div className="container mx-auto px-4 py-16">
          <div className="grid gap-8 md:grid-cols-2">
           <ProductGallery
              images={[
                product.imageUrl,
                product.imageUrl2,
                product.imageUrl3,
                product.imageUrl4
              ].filter(Boolean)}
            />

            <div className="space-y-6">
              <h1 className="text-4xl font-bold text-[#151875]">{product.name}</h1>
              <div className="flex items-center gap-1">
                {Array.from({ length: 5 }).map((_, i) => (
                  <span key={i} className="text-yellow-400">
                    ★
                  </span>
                ))}
                <span className="ml-2 text-gray-500">(5)</span>
              </div>
              <div className="flex items-center gap-4">
                <span className="text-2xl font-bold text-[#151875]">
                  ${product.priceWithoutDiscount}
                </span>
                {product.discountPercentage > 0 && (
                  <span className="text-xl text-gray-400 line-through">
                    ${product.price}
                  </span>
                )}
              </div>
              <p className="text-gray-600">{product.description}</p>
              <div>
                <label className="font-bold text-[#151875]">Color</label>
                <div className="mt-2 flex gap-2">
                  {product.colors.map((color, index) => (
                    <button
                      key={index}
                      className="h-6 w-6 rounded-full border-2"
                      style={{ backgroundColor: color }}
                    />
                  ))}
                </div>
              </div>
              <div className="flex items-center gap-4">
                <div className="flex items-center">
                  <button
                    onClick={() => handleQuantityChange(-1)}
                    className="h-10 w-8 border border-r-0 hover:bg-gray-50"
                  >
                    -
                  </button>
                  <input
                    type="number"
                    value={quantity}
                    onChange={(e) => setQuantity(Math.max(1, parseInt(e.target.value)))}
                    className="h-10 w-12 border text-center"
                  />
                  <button
                    onClick={() => handleQuantityChange(1)}
                    className="h-10 w-8 border border-l-0 hover:bg-gray-50"
                  >
                    +
                  </button>
                </div>
                <Button className="bg-[#FB2E86] text-white rounded-xl hover:bg-[#FB2E86]/90">
                  Add To Cart
                </Button>
                <Button
                  variant="ghost"
                  size="icon"
                  className="rounded-full hover:bg-pink-50 hover:text-[#FB2E86]"
                >
                  <Heart className={`h-5 w-5 ${isWishlisted ? 'fill-red-600 bg-transparent
                text-red-600' : 'bg-transparent text-red-600'}`} />
                </Button>
              </div>
              <div className="space-y-2 border-t pt-6">
                <p>
                  <span className="font-bold">Categories:</span> {product.category}
                </p>
                <p>
                  <span className="font-bold">Tags:</span> {product.tags}
                </p>
              </div>
            </div>
          </div>

          {/* Product Information Tabs */}
          <div className="mt-16">
            <Tabs defaultValue="description">
              <TabsList className="border-b space-x-4">
                <TabsTrigger value="description">Description</TabsTrigger>
                <TabsTrigger value="additional">Additional Info</TabsTrigger>
                <TabsTrigger value="reviews">Reviews (0)</TabsTrigger>
                <TabsTrigger value="video">Video</TabsTrigger>
              </TabsList>
              <TabsContent value="description" className="mt-8">
                <div className="prose max-w-none">
                  <p>
                   {}
                  </p>
                </div>
              </TabsContent>
              <TabsContent value="additional" className="mt-8">
                {}
              </TabsContent>
              <TabsContent value="reviews" className="mt-8">
                <p>No reviews yet.</p>
              </TabsContent>
              <TabsContent value="video" className="mt-8">
                <p>No video available.</p>
              </TabsContent>
            </Tabs>
          </div>

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



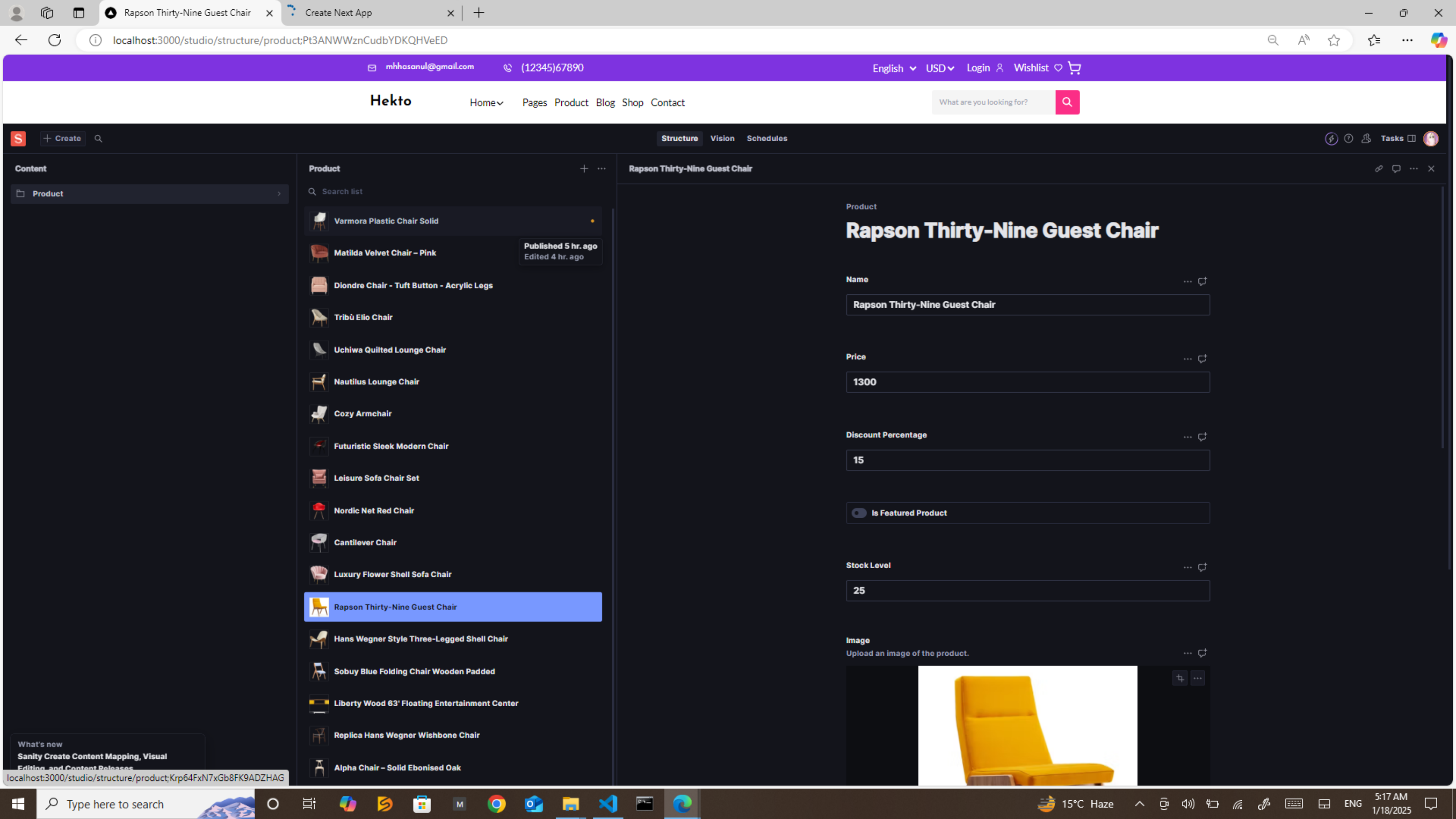
GET https://next-ecommerce-template-4.vercel.app/api/product Send

Query Parameters

parameter value

Status: 200 OK Size: 6.38 KB Time: 477 ms

```
{
  "products": [
    {
      "id": "1",
      "name": "Tribù Elio Chair",
      "imagePath": "https://next-ecommerce-template-4.vercel.app/product/Chair (5
        ).png",
      "price": "1200",
      "description": "A sleek outdoor chair with natural wooden elements and a modern
        look.",
      "discountPercentage": 10,
      "isFeaturedProduct": true,
      "stockLevel": 25,
      "category": "Chair"
    },
    {
      "id": "2",
      "name": "Armchair Tortuga",
      "imagePath": "https://next-ecommerce-template-4.vercel.app/product/Chair (35
        ).png",
      "price": "850",
      "description": "An elegant armchair with plush cushions and a curved design for
        comfort.",
      "discountPercentage": 0,
      "isFeaturedProduct": false,
      "stockLevel": 10,
```



Rapson Thirty-Nine Guest Chair | Create Next App

localhost:3000/studio/structure/product;Pt3ANWWznCudbYDKQHVeED

mhhasanul@gmail.com (12345)67890 English USD Login Wishlist

Hekto Home Pages Product Blog Shop Contact What are you looking for?

Create Structure Vision Schedules Tasks

Content
- Product

Product
Search list
- Varmora Plastic Chair Solid
- Matilda Velvet Chair – Pink (Published 5 hr. ago, Edited 4 hr. ago)
- Diondre Chair - Tuft Button - Acrylic Legs
- Tribù Elio Chair
- Uchiwa Quilted Lounge Chair
- Nautilus Lounge Chair
- Cozy Armchair
- Futuristic Sleek Modern Chair
- Leisure Sofa Chair Set
- Nordic Net Red Chair
- Cantilever Chair
- Luxury Flower Shell Sofa Chair
- Rapson Thirty-Nine Guest Chair
- Hans Wegner Style Three-Legged Shell Chair
- Sobuy Blue Folding Chair Wooden Padded
- Liberty Wood 63' Floating Entertainment Center
- Replica Hans Wegner Wishbone Chair
- Alpha Chair – Solid Ebonised Oak

What's new
Sanity Create Content Mapping, Visual Editing, and Content Releases

localhost:3000/studio/structure/product;Krp64FxN7xGb8FK9ADZHAG

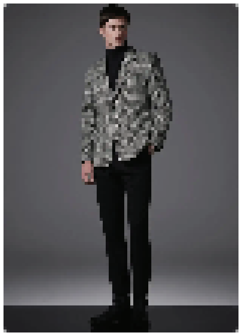
Rapson Thirty-Nine Guest Chair

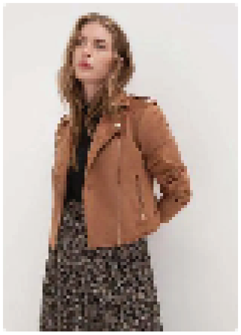
Product
Rapson Thirty-Nine Guest Chair

Name	Rapson Thirty-Nine Guest Chair
Price	1300
Discount Percentage	15
Is Featured Product	
Stock Level	25

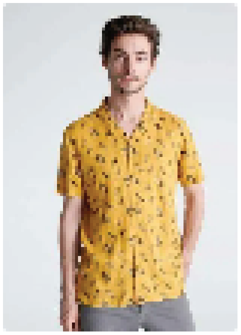
Image
Upload an image of the product.

Type here to search 15°C Haze ENG 5:17 AM 1/18/2025

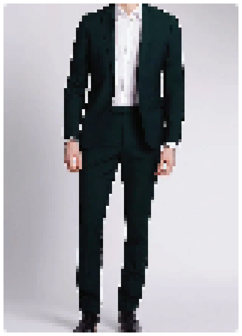

Jacket Dress
Department
~~$23~~ $12.5

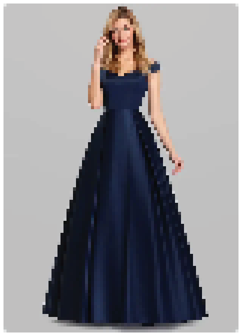

Designer Dress
Department
~~$28.5~~ $19.5

Formal Gown
Department
~~$25~~ $15


Summer Dress
Department
~~$30~~ $20

Party Dress
Department
~~$18~~ $12

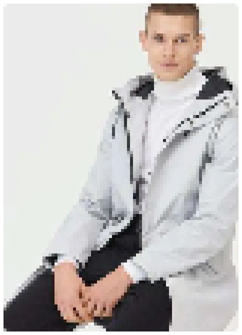

Office Dress
Department
~~$35~~ $27


Evening Gown
Department
~~$27~~ $18

Casual Dress
Department
~~$20~~ $10

Cocktail Dress
Department
~~$24~~ $16

Traditional Dress
Department
~~$28~~ $20


Winter Coat
Department
~~$22~~ $12

Maxi Dress
Department
~~$19~~ $9