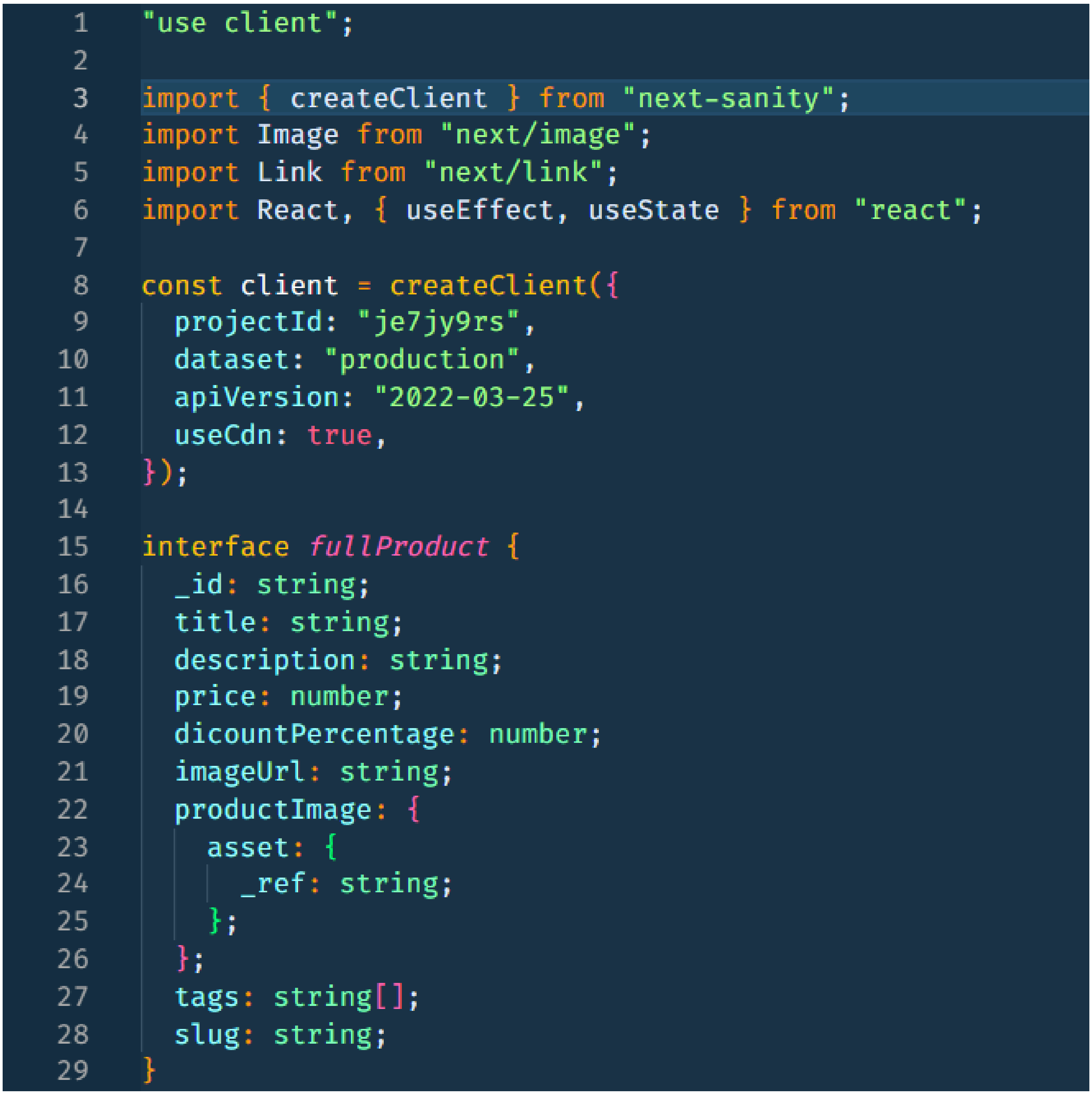
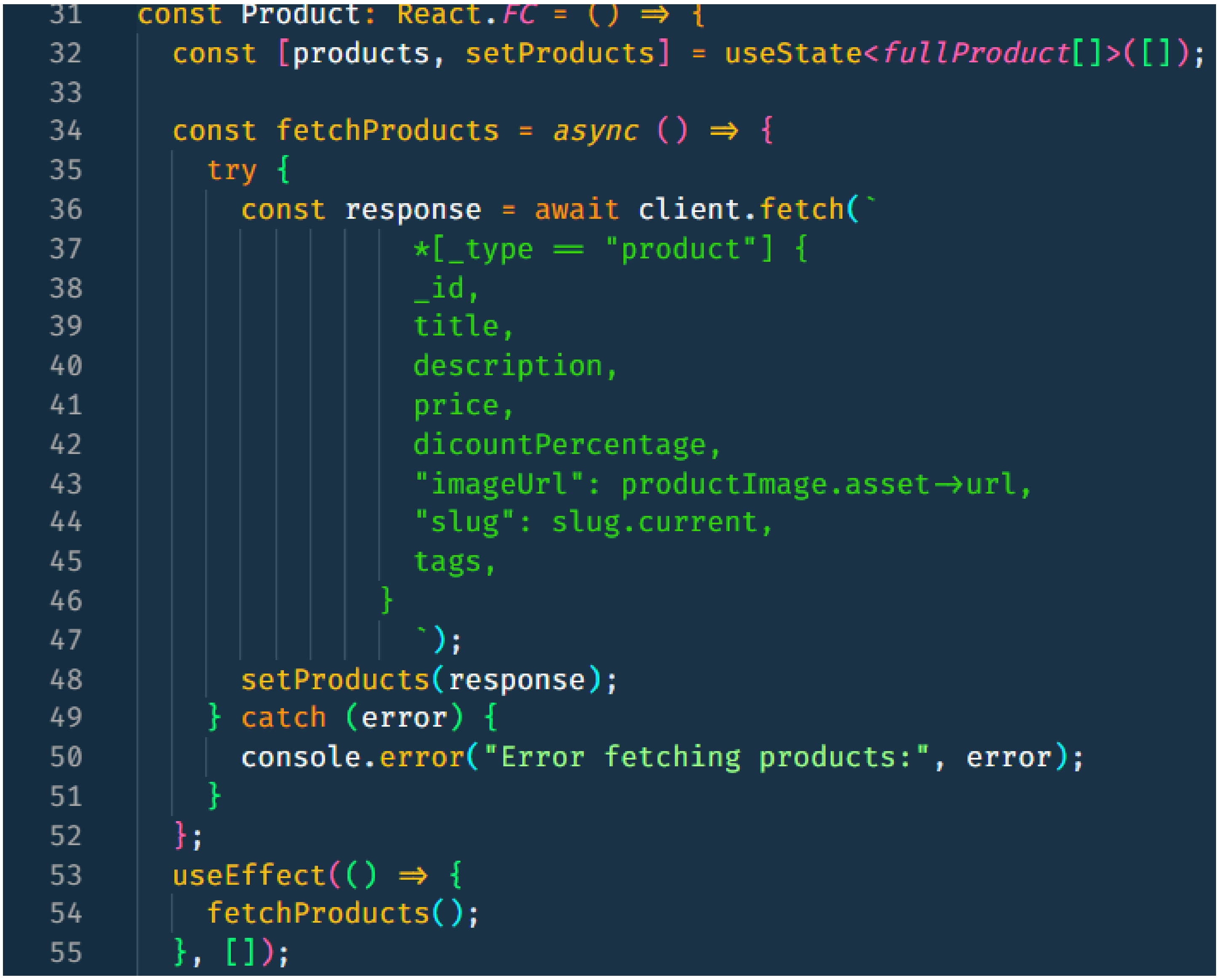
API Integration Report – Bandage

1) **Understand the Provided API**

- **Reviewed API Documentation:** Carefully examined the API documentation for the assigned template, focusing on:
  - **Endpoints:** Identified key endpoints such as /products.
  - **Methods:** Determined the HTTP methods used for each endpoint (GET, POST, PUT, DELETE).
  - **Request/Response Formats:** Analyzed the expected request payloads and the structure of the API responses.



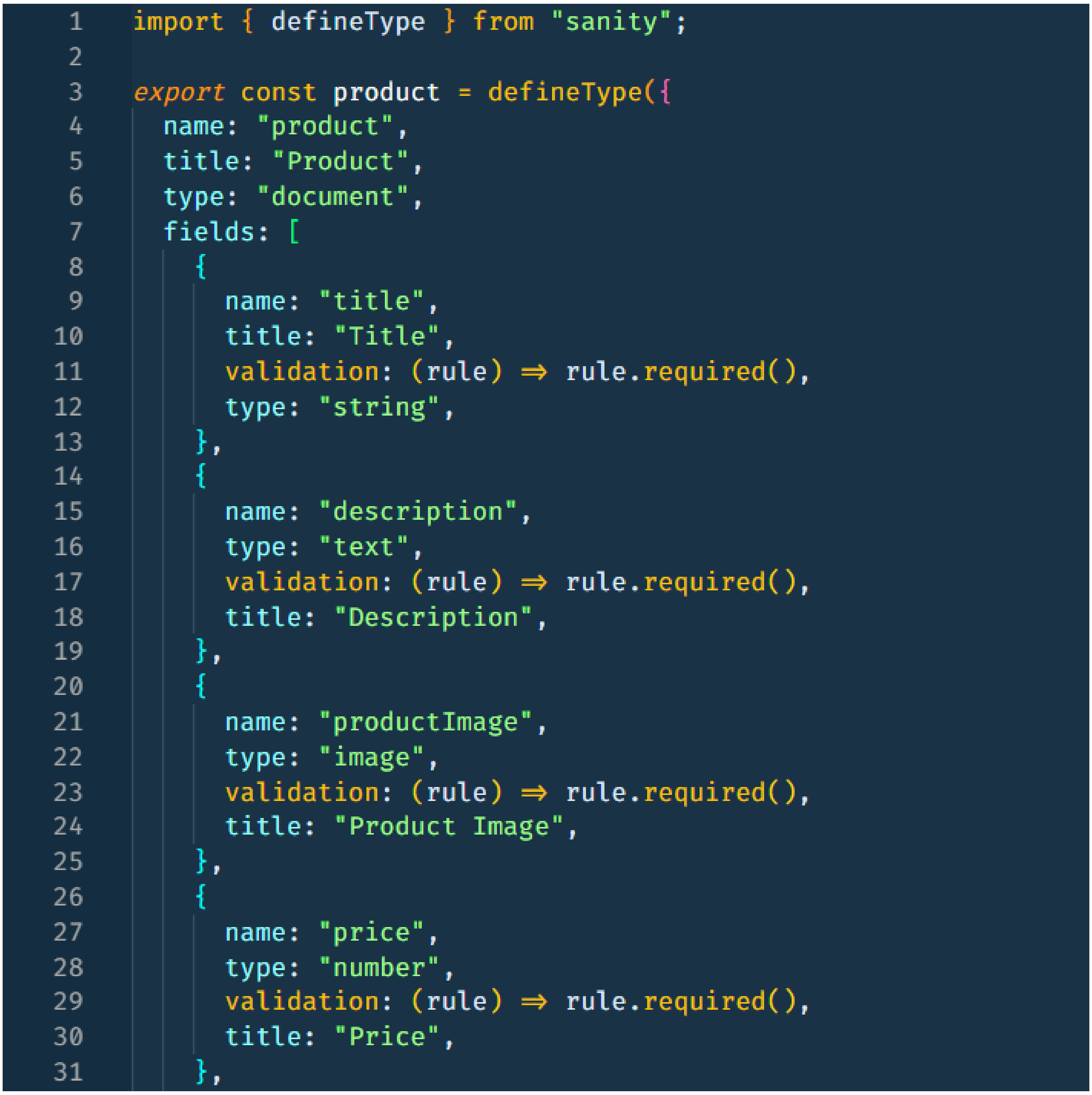
```tsx
"use client";

import { createClient } from "next-sanity";
import Image from "next/image";
import Link from "next/link";
import React, { useEffect, useState } from "react";

const client = createClient({
  projectId: "je7jy9rs",
  dataset: "production",
  apiVersion: "2022-03-25",
  useCdn: true,
});

interface fullProduct {
  _id: string;
  title: string;
  description: string;
  price: number;
  dicountPercentage: number;
  imageUrl: string;
  productImage: {
    asset: {
      _ref: string;
    };
  };
  tags: string[];
  slug: string;
}
```



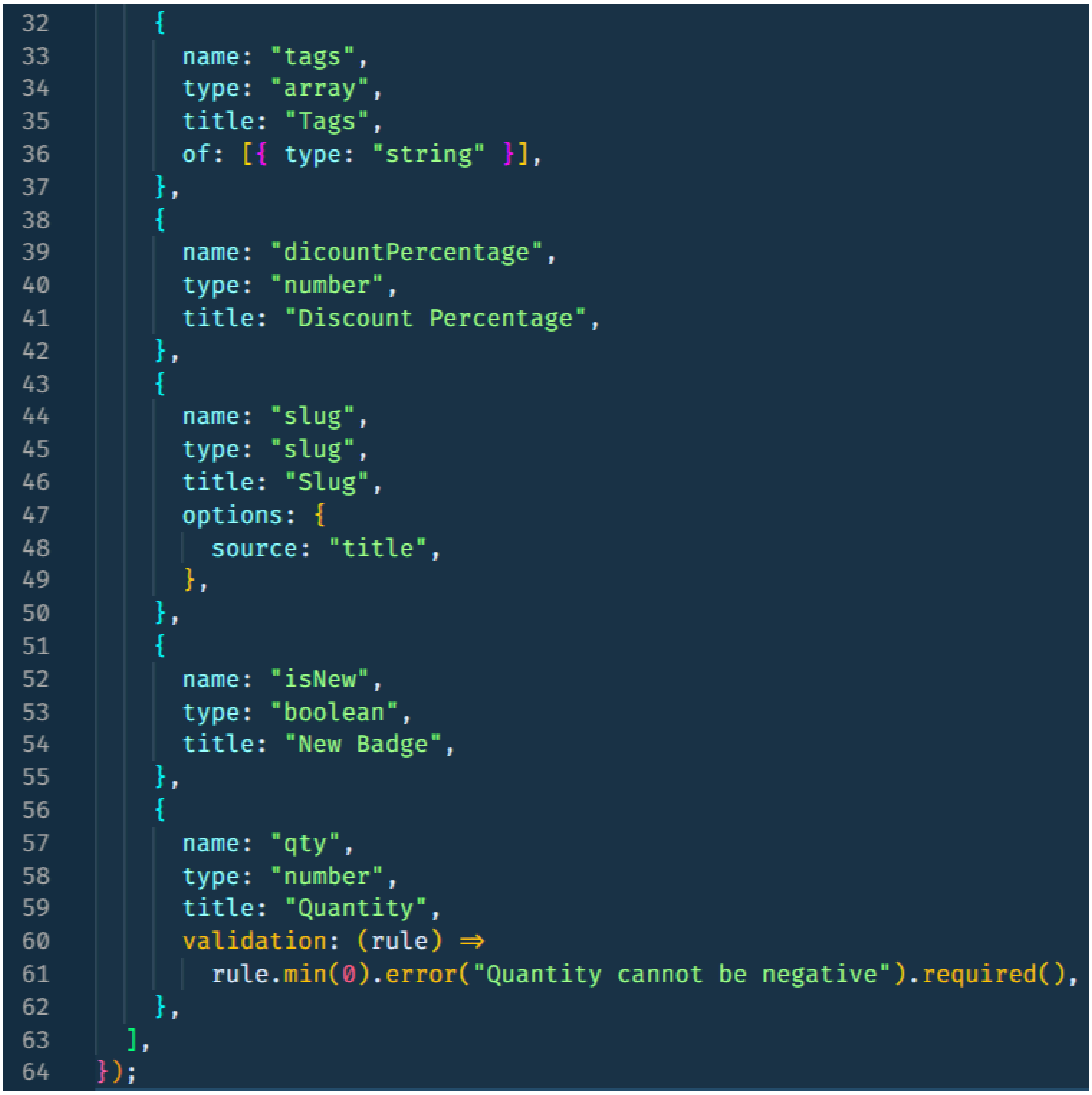
```
const Product: React.FC = () => {
  const [products, setProducts] = useState<fullProduct[]>([]);

  const fetchProducts = async () => {
    try {
      const response = await client.fetch(`
                *[_type == "product"] {
                _id,
                title,
                description,
                price,
                dicountPercentage,
                "imageUrl": productImage.asset->url,
                "slug": slug.current,
                tags,
              }
                `);
      setProducts(response);
    } catch (error) {
      console.error("Error fetching products:", error);
    }
  };
  useEffect(() => {
    fetchProducts();
  }, []);
```

2) **Validate and Adjust Schema**

• **Schema Comparison:** Compared the existing Sanity CMS schema defined on Day 2 with the data structure provided by the API.



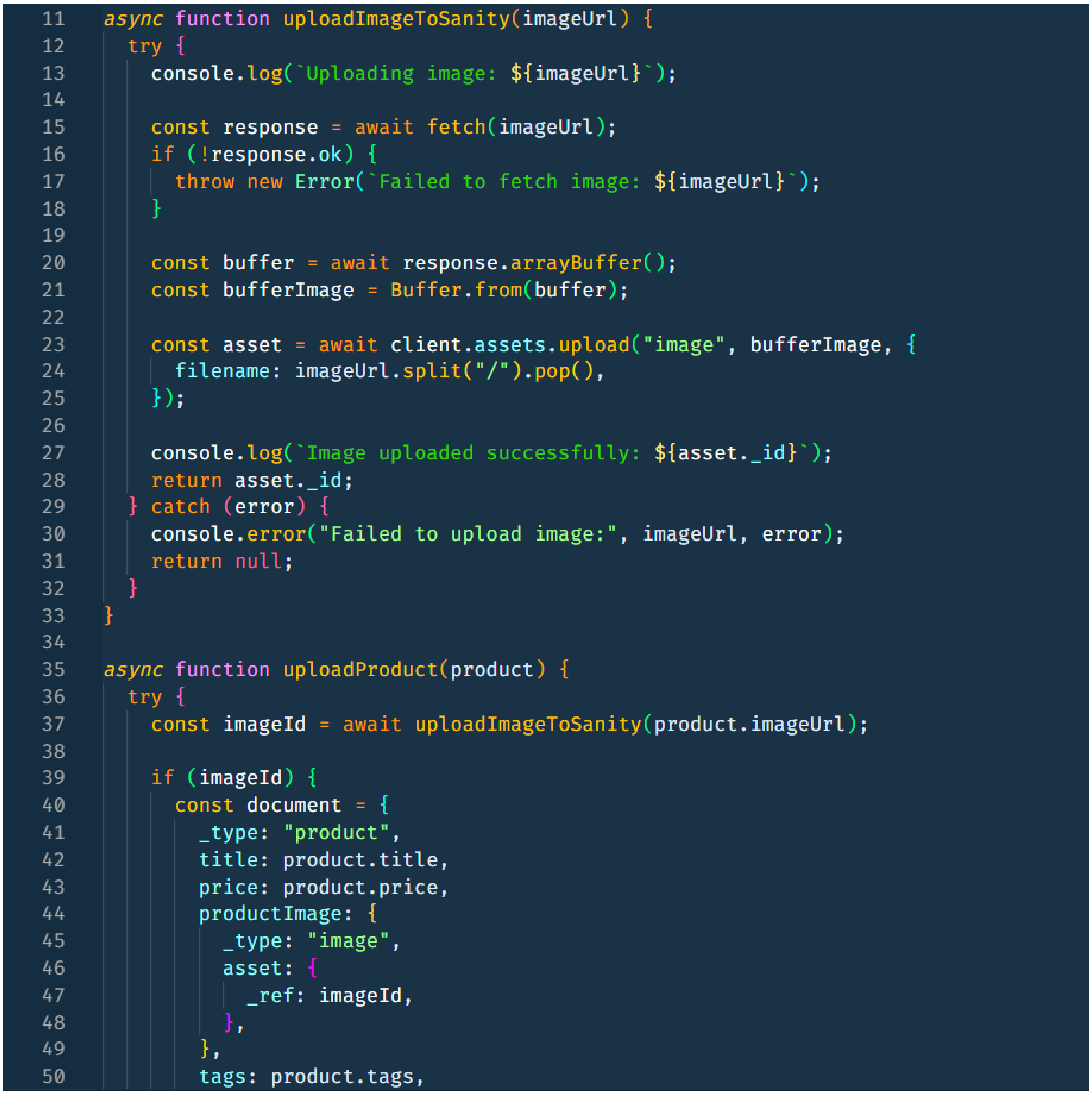
```
import { defineType } from "sanity";

export const product = defineType({
  name: "product",
  title: "Product",
  type: "document",
  fields: [
    {
      name: "title",
      title: "Title",
      validation: (rule) => rule.required(),
      type: "string",
    },
    {
      name: "description",
      type: "text",
      validation: (rule) => rule.required(),
      title: "Description",
    },
    {
      name: "productImage",
      type: "image",
      validation: (rule) => rule.required(),
      title: "Product Image",
    },
    {
      name: "price",
      type: "number",
      validation: (rule) => rule.required(),
      title: "Price",
    },
```



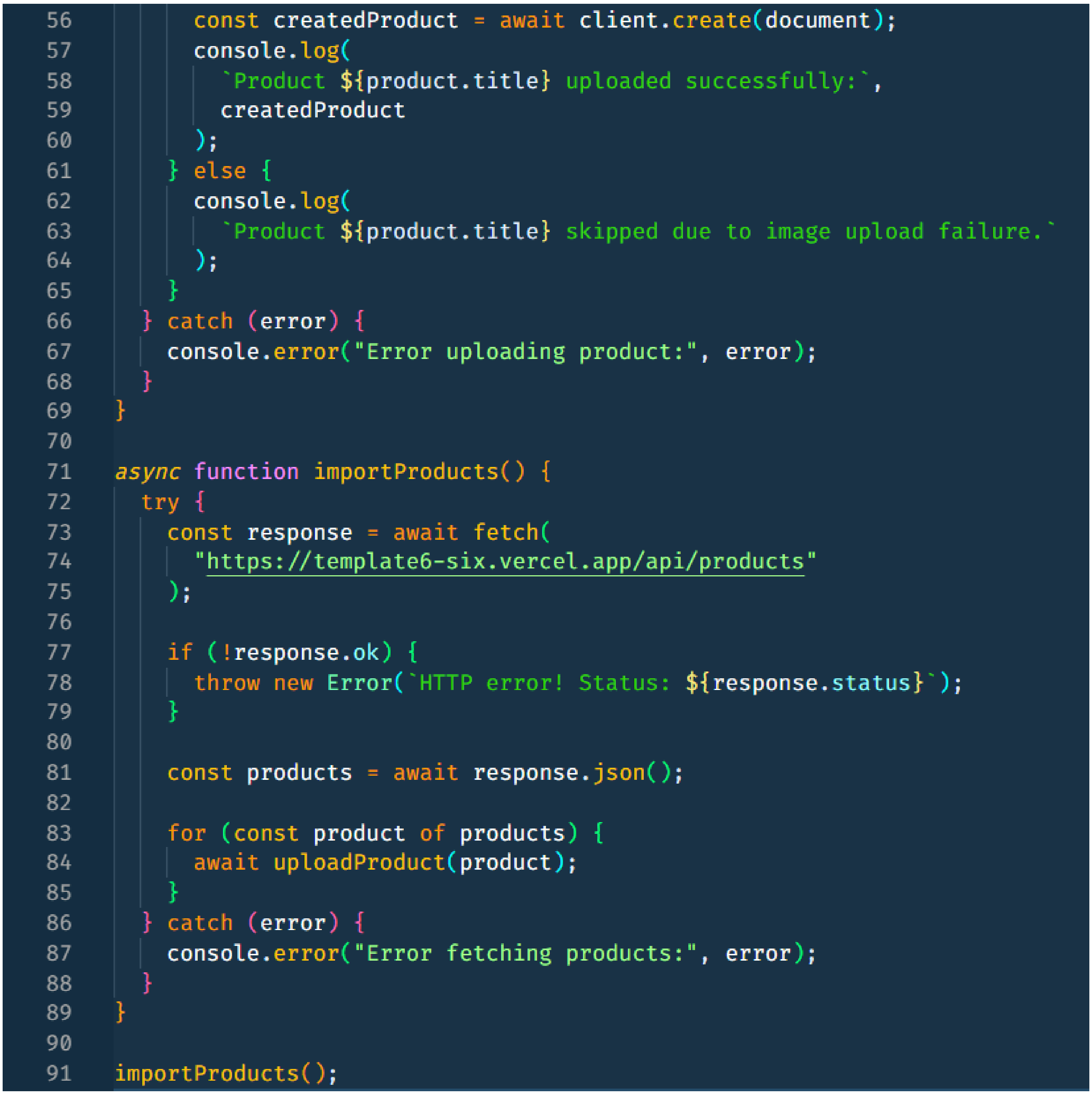
```
    {
      name: "tags",
      type: "array",
      title: "Tags",
      of: [{ type: "string" }],
    },
    {
      name: "dicountPercentage",
      type: "number",
      title: "Discount Percentage",
    },
    {
      name: "slug",
      type: "slug",
      title: "Slug",
      options: {
        source: "title",
      },
    },
    {
      name: "isNew",
      type: "boolean",
      title: "New Badge",
    },
    {
      name: "qty",
      type: "number",
      title: "Quantity",
      validation: (rule) =>
        rule.min(0).error("Quantity cannot be negative").required(),
    },
  ],
});
```

3) **Data Migration**

- **Chosen Method:** Selected "Using the Provided API" as the primary data migration method.



```javascript
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;
  }
}

async function uploadProduct(product) {
  try {
    const imageId = await uploadImageToSanity(product.imageUrl);

    if (imageId) {
      const document = {
        _type: "product",
        title: product.title,
        price: product.price,
        productImage: {
          _type: "image",
          asset: {
            _ref: imageId,
          },
        },
        tags: product.tags,
```



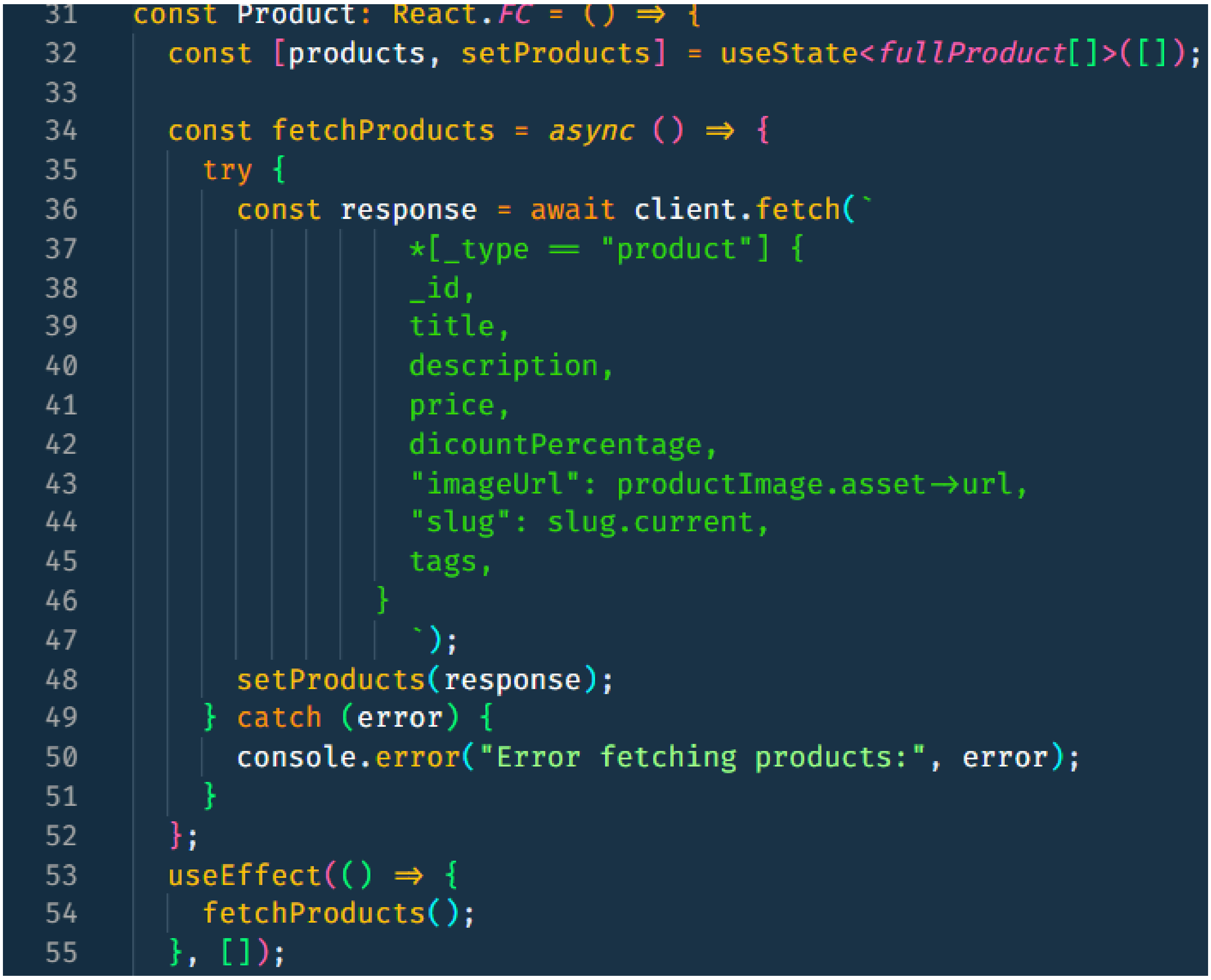
```
      const createdProduct = await client.create(document);
      console.log(
        `Product ${product.title} uploaded successfully:`,
        createdProduct
      );
    } else {
      console.log(
        `Product ${product.title} skipped due to image upload failure.`
      );
    }
  } catch (error) {
    console.error("Error uploading product:", error);
  }
}

async function importProducts() {
  try {
    const response = await fetch(
      "https://template6-six.vercel.app/api/products"
    );

    if (!response.ok) {
      throw new Error(`HTTP error! Status: ${response.status}`);
    }

    const products = await response.json();

    for (const product of products) {
      await uploadProduct(product);
    }
  } catch (error) {
    console.error("Error fetching products:", error);
  }
}

importProducts();
```

- Data Validation:
  - Thoroughly validated the imported data in Sanity CMS to ensure accuracy and consistency.
  - Checked for missing fields, incorrect data types, and any other discrepancies.

4) **API Integration in Next.js**

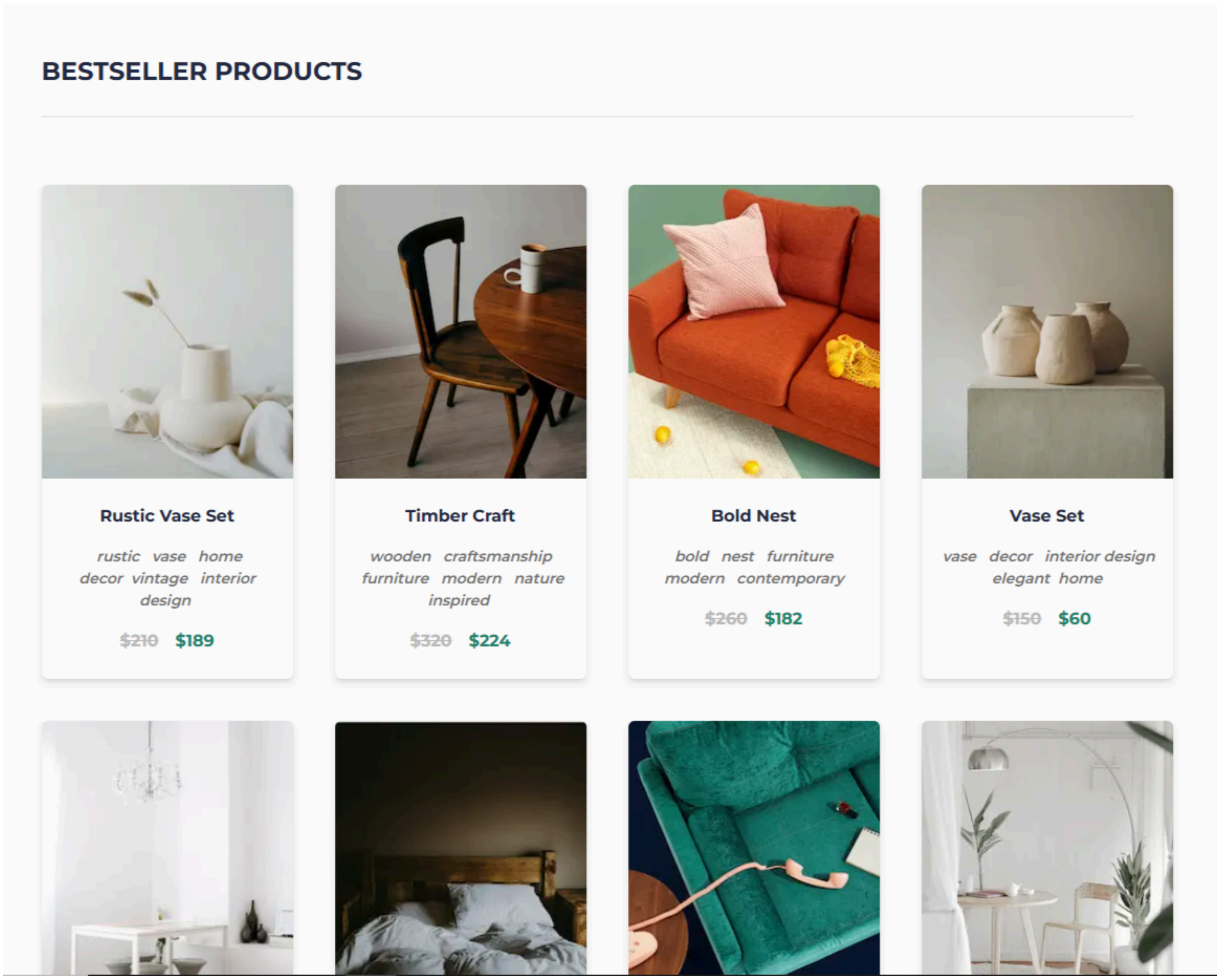
- Utility Functions: Created reusable utility functions in Next.js to:
  - Fetch data from the API endpoints.

- Handle API requests and responses (e.g., error handling, data parsing).
- Cache API responses to improve performance.



```
const Product: React.FC = () => {
  const [products, setProducts] = useState<fullProduct[]>([]);

  const fetchProducts = async () => {
    try {
      const response = await client.fetch(`
                *[_type == "product"] {
                _id,
                title,
                description,
                price,
                dicountPercentage,
                "imageUrl": productImage.asset->url,
                "slug": slug.current,
                tags,
              }
                `);
      setProducts(response);
    } catch (error) {
      console.error("Error fetching products:", error);
    }
  };
  useEffect(() => {
    fetchProducts();
  }, []);
```

- **Component Rendering:** Integrated the API utility functions into the frontend components.
  - Used the fetched data to dynamically render product listings, category pages, and other components.
  - Implemented data fetching and loading states to provide a smooth user experience.



BESTSELLER PRODUCTS

Rustic Vase Set

rustic vase home decor vintage interior design

~~$210~~ $189

Timber Craft

wooden craftsmanship furniture modern nature inspired

~~$320~~ $224

Bold Nest

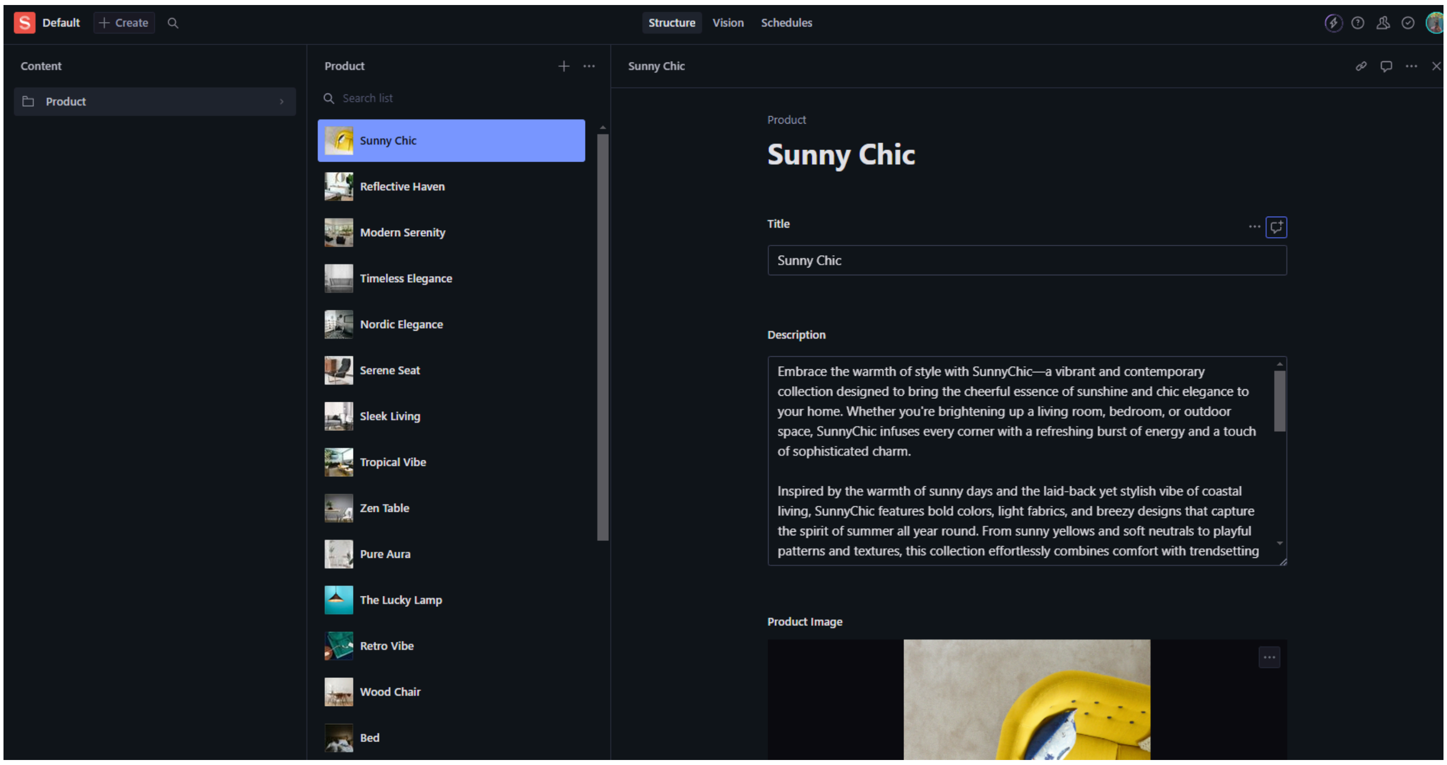
bold nest furniture modern contemporary

~~$260~~ $182

Vase Set

vase decor interior design elegant home

~~$150~~ $60



Sunny Chic

Title: Sunny Chic

Description:

Embrace the warmth of style with SunnyChic—a vibrant and contemporary collection designed to bring the cheerful essence of sunshine and chic elegance to your home. Whether you're brightening up a living room, bedroom, or outdoor space, SunnyChic infuses every corner with a refreshing burst of energy and a touch of sophisticated charm.

Inspired by the warmth of sunny days and the laid-back yet stylish vibe of coastal living, SunnyChic features bold colors, light fabrics, and breezy designs that capture the spirit of summer all year round. From sunny yellows and soft neutrals to playful patterns and textures, this collection effortlessly combines comfort with trendsetting

Product Image

- **API Testing:**

- Utilized tools like Postman and browser developer tools to test API endpoints and verify data integrity.
- Logged API responses to identify and resolve any issues.