Day 3 - API Integration Report - General E-Commerce Marketplace

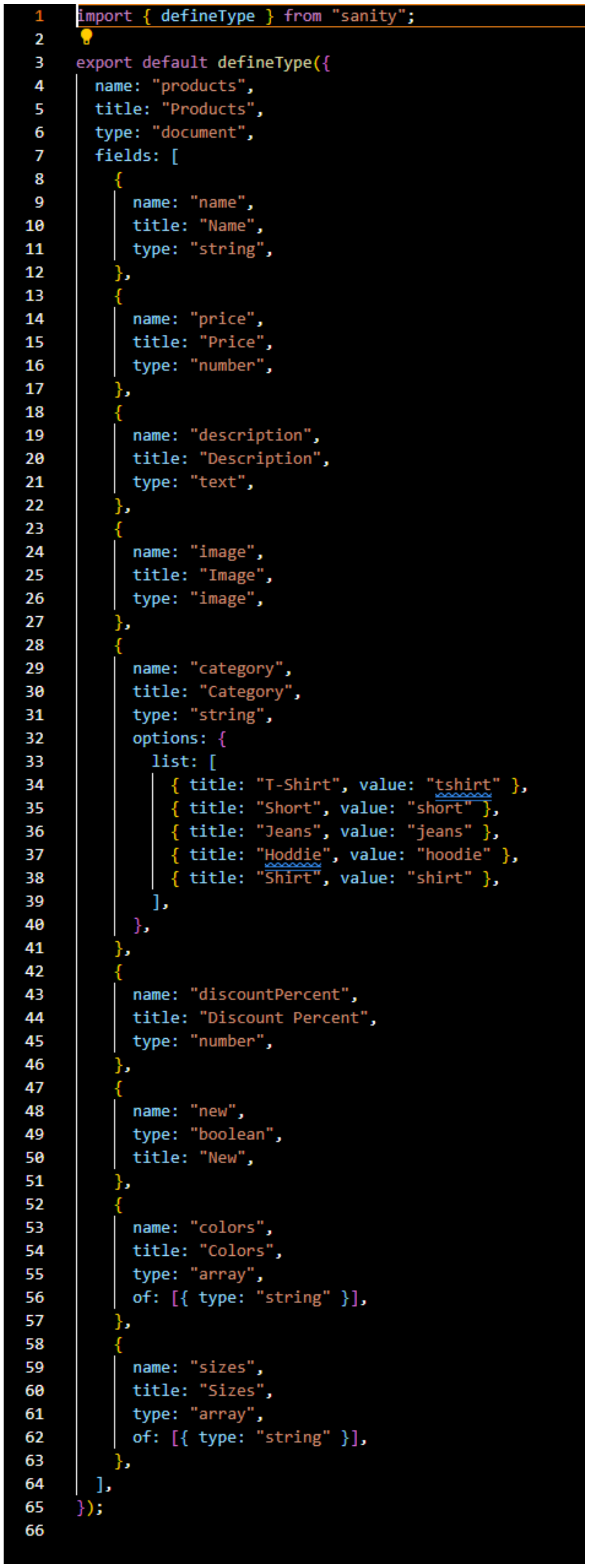
Steps for Day 3

1. Understand the Provided API

- **Review API Documentation:**
  - **Analyzed the documentation for endpoints, including /products, /orders, and /shipment.**
  - **Ensured understanding of payload structures, methods, and response formats.**
- **Key Endpoints Identified:**
  - **Product Listings: Endpoint: /products (GET) - Fetches product details including name, price, and stock.**
  - **Orders: Endpoint: /orders (POST) - Submits customer orders, including product details and customer information.**
  - **Shipment Tracking: Endpoint: /shipment (GET) - Provides shipment statuses.**

2. Validate and Adjust Your Schema

- **Schema Validation:**
  - **Compared the existing Sanity CMS schema with API response structures.**
  - **Adjusted schema fields to match API fields. For example:**
    - **API Field: product_title -> Schema Field: name**
    - **API Field: price_usd -> Schema Field: price**
- **Sanity Schema:**

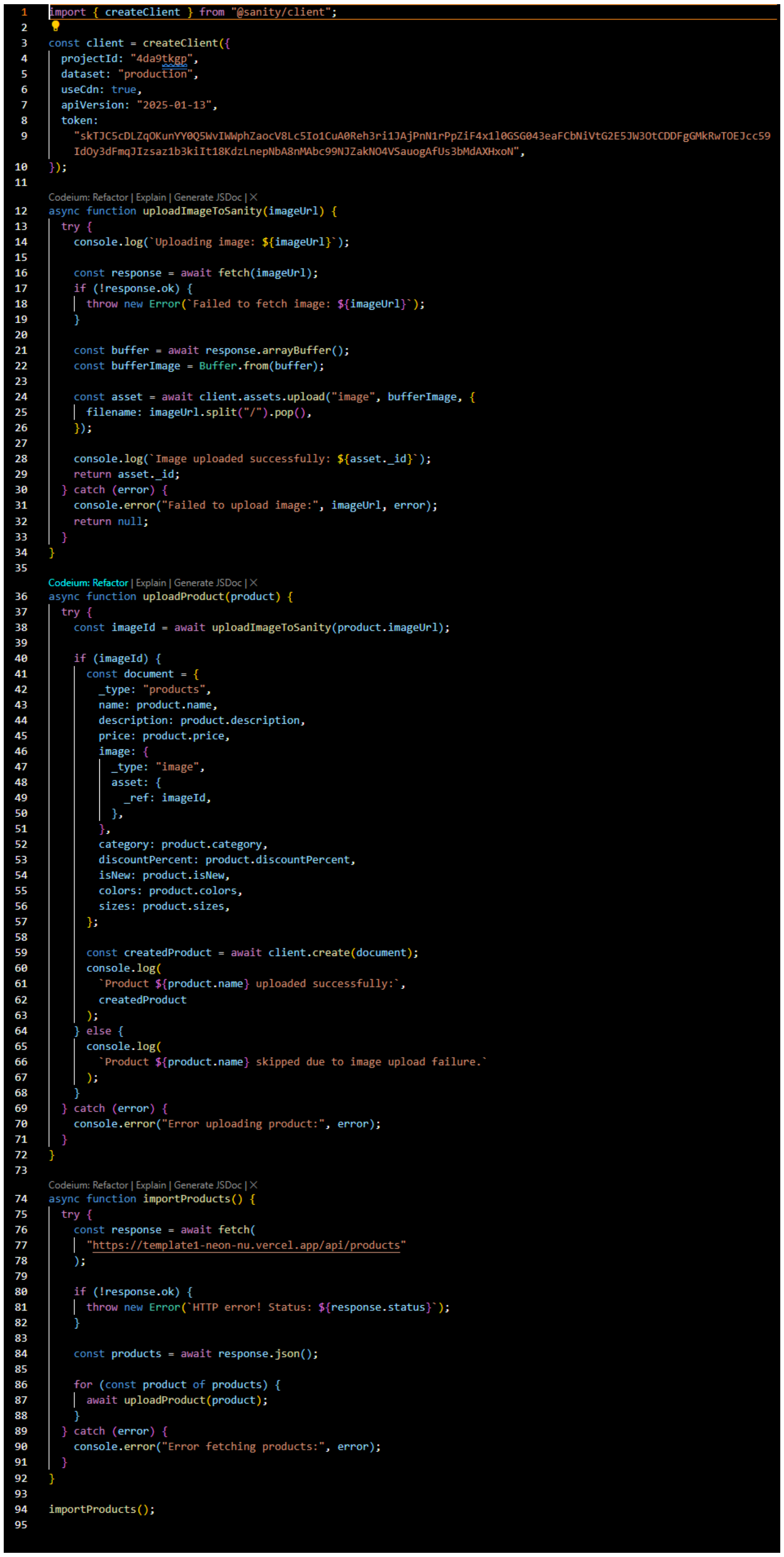


```
import { defineType } from "sanity";

export default defineType({
  name: "products",
  title: "Products",
  type: "document",
  fields: [
    {
      name: "name",
      title: "Name",
      type: "string",
    },
    {
      name: "price",
      title: "Price",
      type: "number",
    },
    {
      name: "description",
      title: "Description",
      type: "text",
    },
    {
      name: "image",
      title: "Image",
      type: "image",
    },
    {
      name: "category",
      title: "Category",
      type: "string",
      options: {
        list: [
          { title: "T-Shirt", value: "tshirt" },
          { title: "Short", value: "short" },
          { title: "Jeans", value: "jeans" },
          { title: "Hoddie", value: "hoodie" },
          { title: "Shirt", value: "shirt" },
        ],
      },
    },
    {
      name: "discountPercent",
      title: "Discount Percent",
      type: "number",
    },
    {
      name: "new",
      type: "boolean",
      title: "New",
    },
    {
      name: "colors",
      title: "Colors",
      type: "array",
      of: [{ type: "string" }],
    },
    {
      name: "sizes",
      title: "Sizes",
      type: "array",
      of: [{ type: "string" }],
    },
  ],
});
```

3. Data Migration Options

- **Developed scripts to fetch, transform, and upload data from the API to Sanity CMS using @sanity/client.**



```javascript
import { createClient } from "@sanity/client";

const client = createClient({
  projectId: "4da9tkgp",
  dataset: "production",
  useCdn: true,
  apiVersion: "2025-01-13",
  token:
    "skTJC5cDLZqOKunYY0Q5WvIWWphZaocV8Lc5Io1CuA0Reh3ri1JAjPnN1rPpZiF4x1l0GSG043eaFCbNiVtG2E5JW3OtCDDFgGMkRwTOEJcc59
    IdOy3dFmqJIzsaz1b3kiIt18KdzLnepNbA8nMAbc99NJZakNO4VSauogAfUs3bMdAXHxoN",
});

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: "products",
        name: product.name,
        description: product.description,
        price: product.price,
        image: {
          _type: "image",
          asset: {
            _ref: imageId,
          },
        },
        category: product.category,
        discountPercent: product.discountPercent,
        isNew: product.isNew,
        colors: product.colors,
        sizes: product.sizes,
      };

      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.`
      );
    }
  } catch (error) {
    console.error("Error uploading product:", error);
  }
}

async function importProducts() {
  try {
    const response = await fetch(
      "https://template1-neon-nu.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();
```

4. API Integration in Next.js

- **Utility Functions:**
  - **Created reusable utility functions for API calls.**
- **Component Rendering:**
  - **Displayed API data dynamically in components.**
- **Testing:**
  - **Tested endpoints using Postman and browser dev tools to verify data consistency.**

5. Error Handling Tips

- **Centralized error logging for easier debugging.**
- **Implemented skeleton loaders for better user experience during data fetches.**
