Day 3 – API Integration Report – TrueBuy

1. API Integration Process

Step 1: Understanding the API

- Reviewed the provided API documentation to identify relevant endpoints for product data migration.
- Focused on the /products endpoint to fetch product details such as name, price, description, image, etc.
- Tested the endpoint using Postman to verify the API response structure.

Step 2: Creating Utility Functions

- Built utility functions in the Next.js project to fetch data from the API.
- Example code snippet for fetching data:

```
import axios from 'axios';

export const fetchProducts = async () => {
 try {
  const response = await axios.get(https://template1-neon-nu.vercel.app/api/products);
  return response.data;
 } catch (error) {
  console.error('Error fetching products:', error);
  return [];
 }

};
```

Step 3: Testing API Integration

- Used browser developer tools and Postman to test the data retrieval.
- Ensured that the API calls returned accurate data without errors.

2. Adjustments Made to Schemas

Original Schema Validation

- Compared the existing Sanity CMS schema with the API response.
- Identified differences in field names and data types:
  - API Field: imageUrl → Schema Field: image
  - API Field: isNew → Schema Field: new

Schema Updates

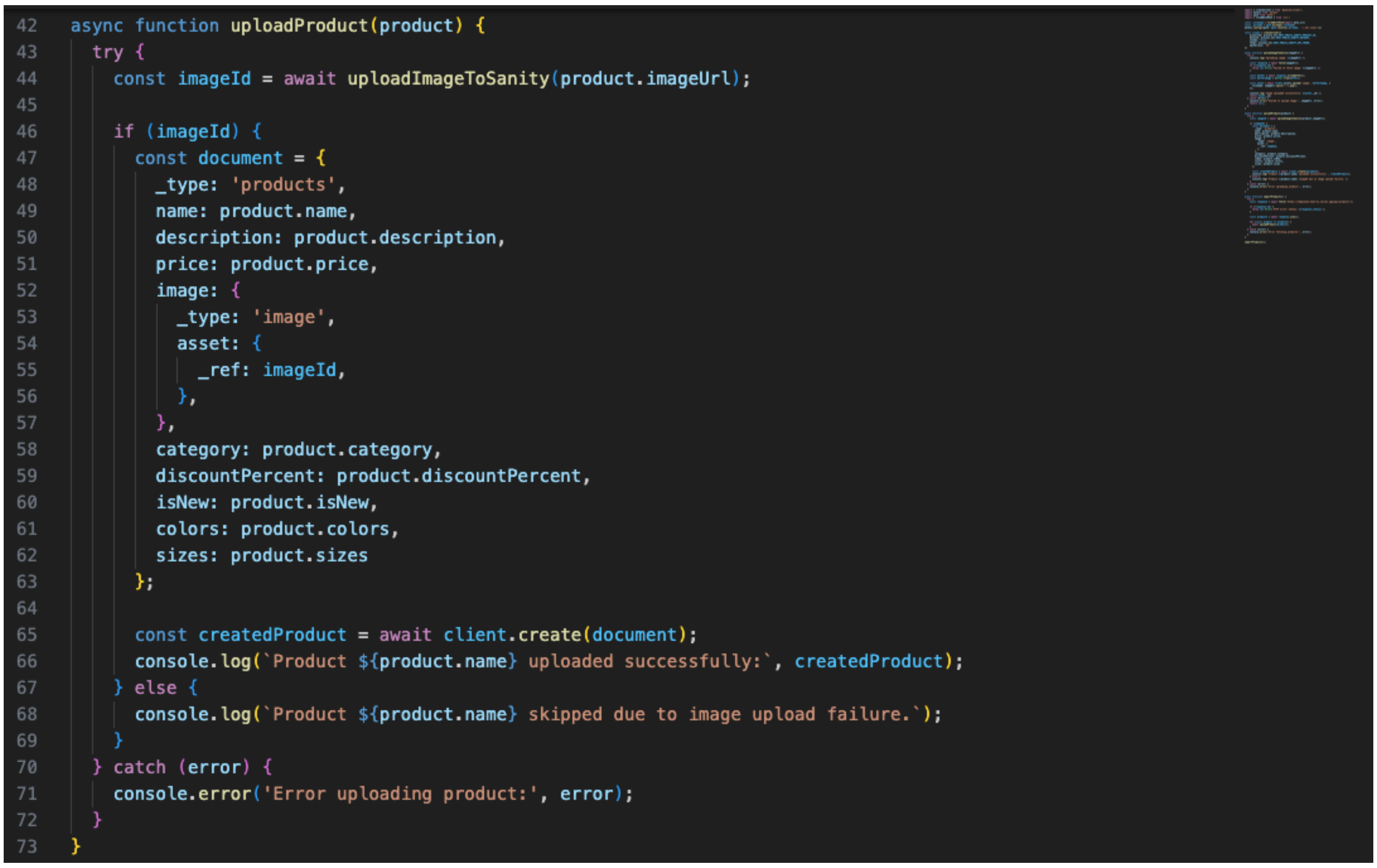
- Updated the Sanity CMS schema to match the API data.

3. Migration Steps and Tools Used

Step 1: Fetching Data from API

- Wrote a script to fetch data from the API and transform it to match the Sanity schema.

Migration Script Example:



```javascript
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);
  }
}
```

Step 2: Validating Data in Sanity CMS

- Imported data was verified in the Sanity CMS dashboard.
- Checked that all fields were populated correctly and matched the API data.

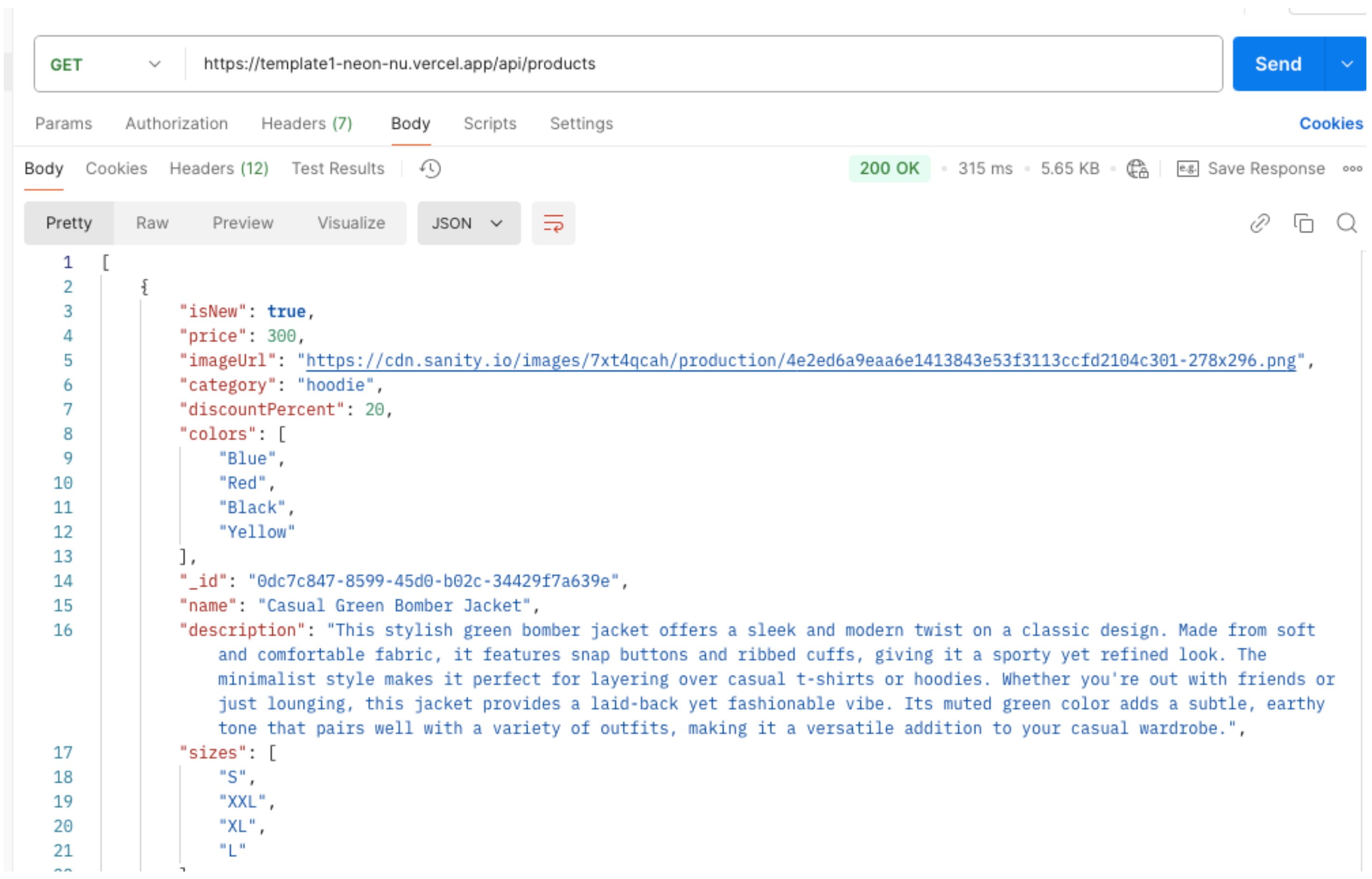
Step 3: Troubleshooting

- Logged errors in the migration script to handle invalid or missing data gracefully.
- Adjusted mappings for any mismatched fields during testing.

4. Screenshots Description

1. API Calls

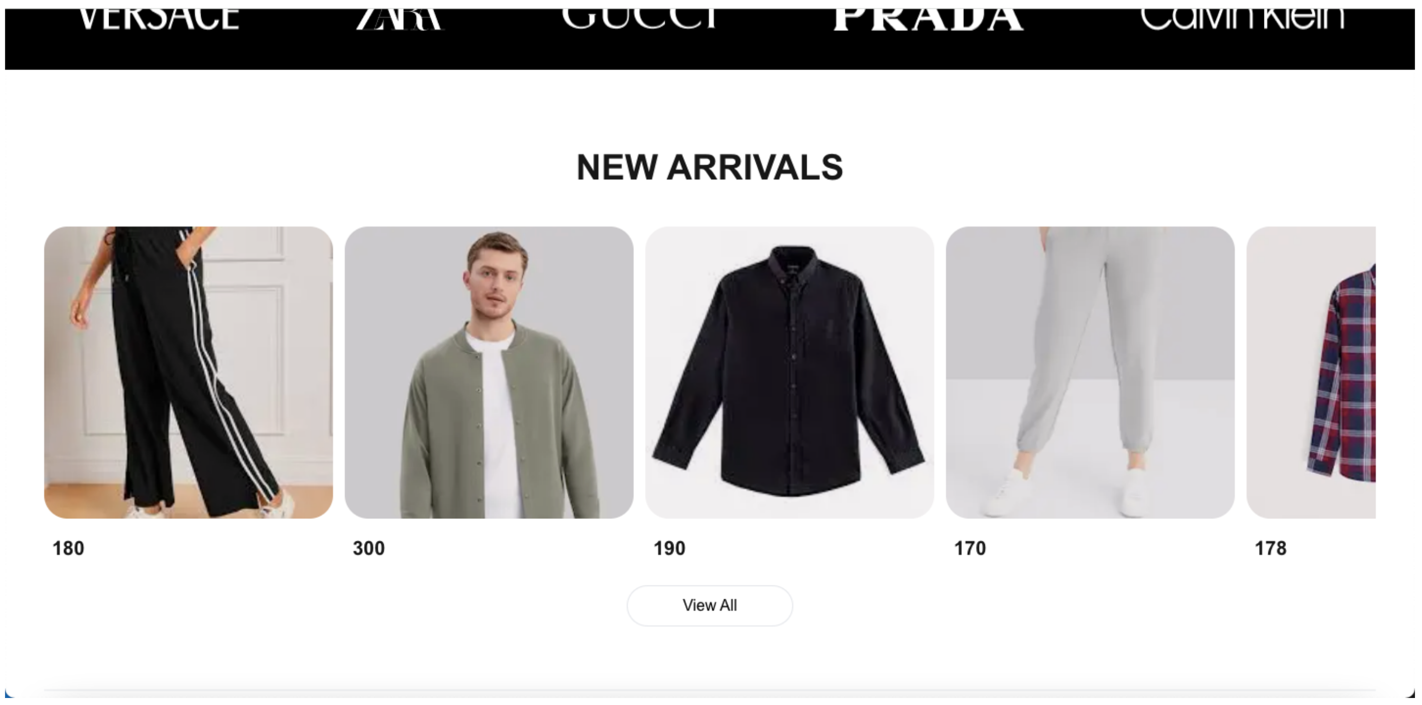
- Show Postman screenshots of successful API requests and responses for the /products endpoint.



```json
[
    {
        "isNew": true,
        "price": 300,
        "imageUrl": "https://cdn.sanity.io/images/7xt4qcah/production/4e2ed6a9eaa6e1413843e53f3113ccfd2104c301-278x296.png",
        "category": "hoodie",
        "discountPercent": 20,
        "colors": [
            "Blue",
            "Red",
            "Black",
            "Yellow"
        ],
        "_id": "0dc7c847-8599-45d0-b02c-34429f7a639e",
        "name": "Casual Green Bomber Jacket",
        "description": "This stylish green bomber jacket offers a sleek and modern twist on a classic design. Made from soft
            and comfortable fabric, it features snap buttons and ribbed cuffs, giving it a sporty yet refined look. The
            minimalist style makes it perfect for layering over casual t-shirts or hoodies. Whether you're out with friends or
            just lounging, this jacket provides a laid-back yet fashionable vibe. Its muted green color adds a subtle, earthy
            tone that pairs well with a variety of outfits, making it a versatile addition to your casual wardrobe.",
        "sizes": [
            "S",
            "XXL",
            "XL",
            "L"
```

2. Data Displayed in Frontend

- Capture screenshot of:



3. Populated Sanity CMS Fields

- Include screenshots of:
  - Sanity CMS dashboard with populated product entries.
  - Detailed view of a single product entry in Sanity.

Products

Classic Polo Shirt

Name

Classic Polo Shirt

Price

180

Description

Classic Polo Shirt

Upgrade your wardrobe with this timeless classic polo shirt, perfect for any occasion. Crafted from premium-quality fabric, it offers a soft, breathable, and comfortable fit that lasts all day. Featuring a stylish collar, button placket, and

Image

Category

T-Shirt

Discount Percent

0

- White
- Black
- Green
- Yellow

+ Add item

Sizes

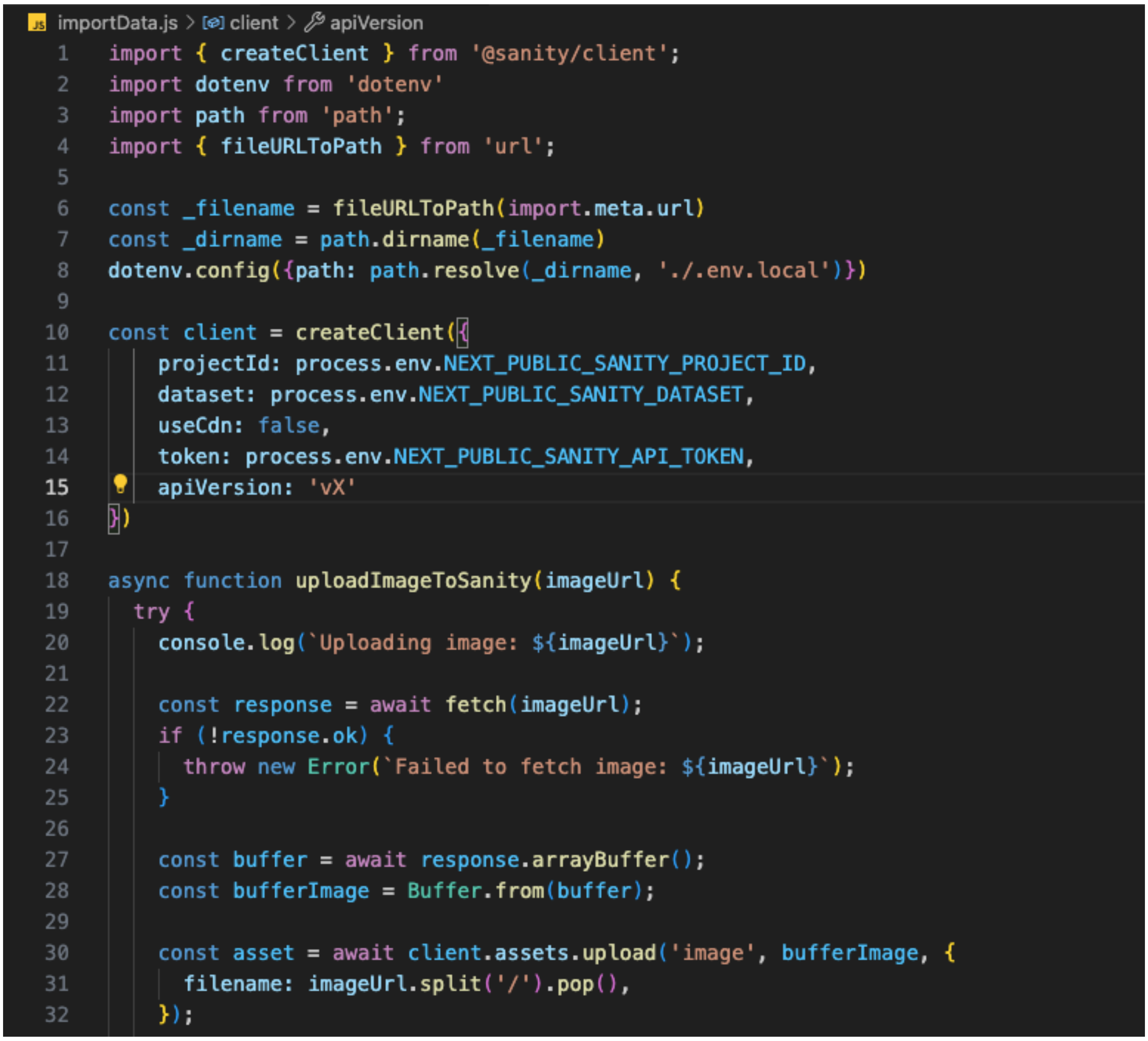
- L
- XXL
- S
- M

5. Code Snippets

Utility Function for API Calls

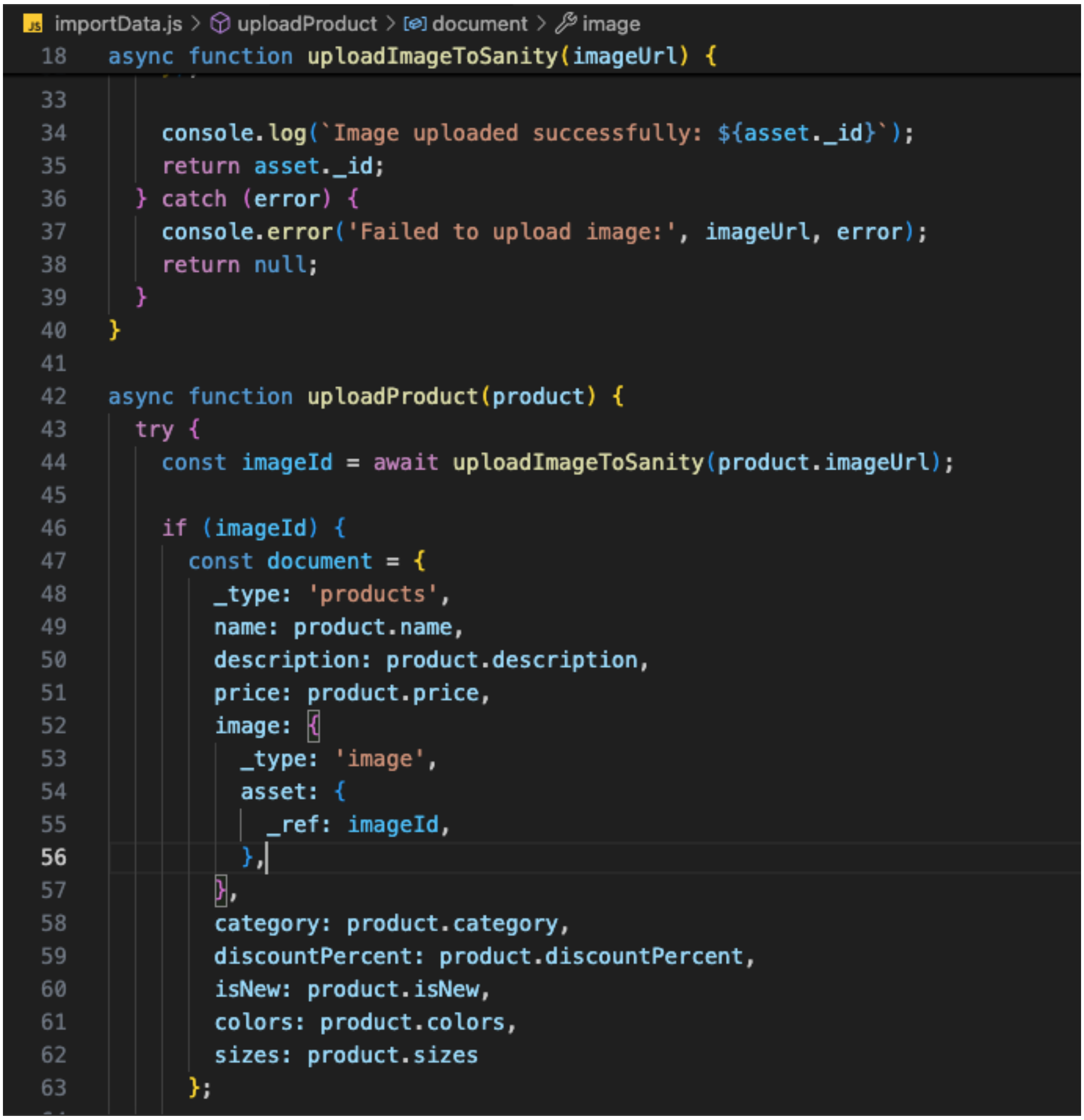
```
import axios from 'axios';
export const fetchProducts = async () => {
 try {
  const response = await axios.get('https://example.com/api/products');
  return response.data;
 } catch (error) {
  console.error('Error fetching products:', error);
  return [];
 }
};
```

Migration Script

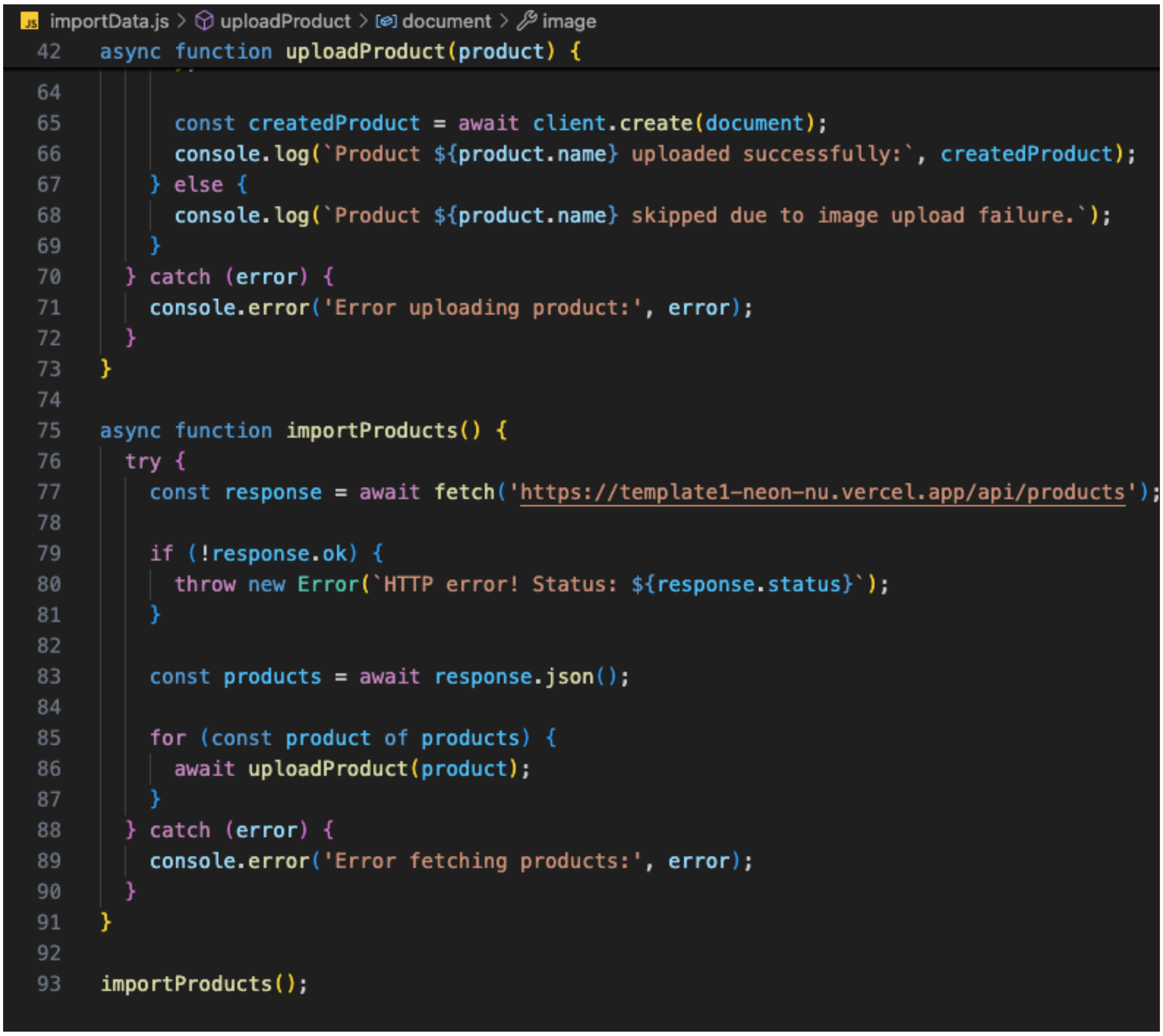


importData.js > client > apiVersion

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

const _filename = fileURLToPath(import.meta.url)
const _dirname = path.dirname(_filename)
dotenv.config({path: path.resolve(_dirname, './.env.local')})

const client = createClient({
    projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
    dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
    useCdn: false,
    token: process.env.NEXT_PUBLIC_SANITY_API_TOKEN,
    apiVersion: 'vX'
})

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(),
    });
```


importData.js > uploadProduct > document > image

```javascript
async function uploadImageToSanity(imageUrl) {

    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
      };
```



```
async function uploadProduct(product) {

      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();
```

This document outlines the API integration and data migration process for Day 3, providing a detailed report of steps, schema adjustments, and code snippets for easy reference .