

## DAY:3 API INTEGRATION

### API Integration Report

API Integration Steps:

#### 1. Integrated **Product** and **Category APIs**:

o Fetched data from APIs using fetch() in the migration script. o Validated the API responses to ensure they matched the schema fields in Sanity CMS.

2: Updated Sanity Schemas: o Resolved field conflicts between manually created schema and API response. o Modified fields in products and categories schemas to align with API structure:

- Products Schema Adjustments: ▪ Added priceWithoutDiscount, badge, tags, etc.
- Categories Schema Adjustments: ▪ Added image as a reference field for category images. o Ensured proper relationships between products and categories.

3: 3. Implemented Migration Script: o Created migrate.mjs to automate data migration. o Used the Sanity client to upload images and save product and category data.

### DATA MIGRATION STEPS

#### 1. **Set Up Sanity in Your Next.js Project**

Ensure Sanity is integrated with your Next.js project and all necessary dependencies are installed.

#### 2. **Export Data from the Source Sanity Project**

Use the Sanity CLI to export your dataset to a file.

#### 3. **Set Up the Target Sanity Project**

Create or prepare the target Sanity project or dataset for migration.

#### 4. **Import Data into the Target Dataset**

Import the exported dataset file into the new Sanity project or dataset using the Sanity CLI.

#### 5. **Update API Configuration in Next.js**

Update your Next.js project to connect to the new Sanity project or dataset.

## 6. Verify the Data

Fetch data from the new Sanity dataset in your Next.js app to ensure the migration is successful.

## 7. Test and Validate

Test your Next.js app to confirm the data is displayed correctly, and check the data in Sanity Studio.

## 8. Handle Schema Updates

If the schemas differ, update them in the target project to match the data structure.

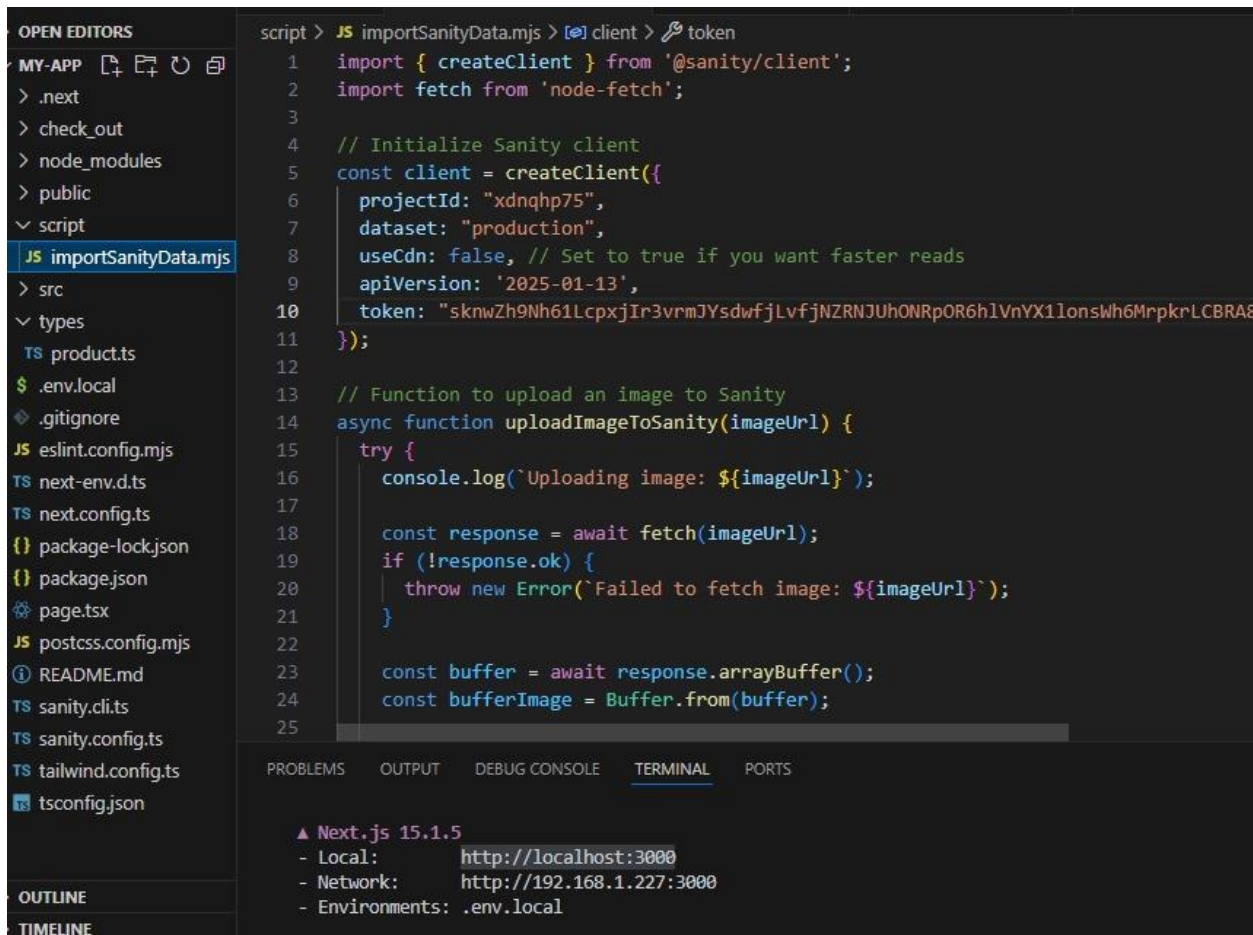
## 9. Clean Up Temporary Files

Remove any exported files or unused data to keep your environment tidy.

## 10. Perform Additional Customizations

Use queries or filters if specific data transformations are needed during migration.

## SCREENSHOTS

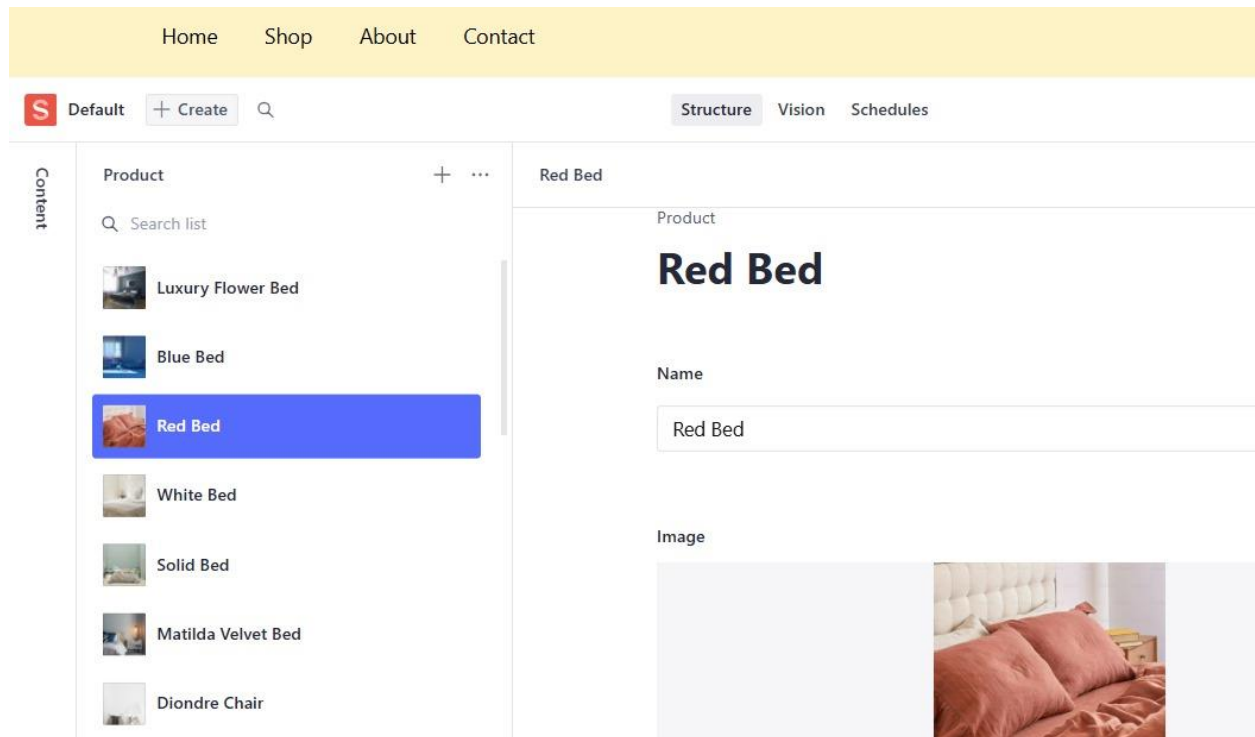


```
script > JS importSanityData.mjs > [client] token
1  import { createClient } from '@sanity/client';
2  import fetch from 'node-fetch';
3
4  // Initialize Sanity client
5  const client = createClient({
6    projectId: "xdnqhp75",
7    dataset: "production",
8    useCdn: false, // Set to true if you want faster reads
9    apiVersion: '2025-01-13',
10   token: "sknwZh9Nh61LcpXjIr3vrmJYsdwfjLvFjNZRNJUhoNRpOR6h1VnYX1lonsWh6MrpkrLCBRA8
11 });
12
13 // Function to upload an image to Sanity
14 async function uploadImageToSanity(imageUrl) {
15   try {
16     console.log(`Uploading image: ${imageUrl}`);
17
18     const response = await fetch(imageUrl);
19     if (!response.ok) {
20       throw new Error(`Failed to fetch image: ${imageUrl}`);
21     }
22
23     const buffer = await response.arrayBuffer();
24     const bufferImage = Buffer.from(buffer);
25   }
26 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
▲ Next.js 15.1.5
- Local:      http://localhost:3000
- Network:    http://192.168.1.227:3000
- Environments: .env.local
```

MARIYAMMAHIDA  
ROLLNO;00498560  
CLASS TIMING:MON(2-5PM)



### Challenges Faced:

- Resolved .env.local conflict by explicitly specifying the .env path in the script
- Adjusted Sanity schemas to match API data fields.
- Handled image uploads and mapped categories properly.
- Decisions Made:
- Used .env for compatibility with the migration script.
- Updated schemas to resolve field conflicts