

Day 3 - API Integration and Data Migration

The focus for Day 3 was integrating APIs and migrating data into Sanity CMS to build a functional marketplace backend. This process helped replicate real-world practices and prepared me to handle diverse client requirements, such as integrating headless APIs or migrating existing data from popular eCommerce platforms.

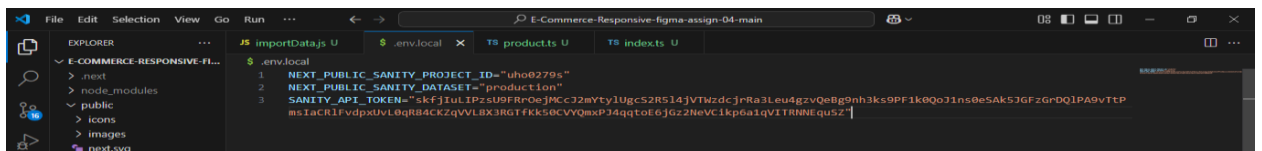
API Overview (Template 1)

- **Migration Script:** <https://github.com/developerhammadrehman/template1/blob/main/importData.js>
- **Schema:** <https://github.com/developerhammadrehman/template1/blob/main/src/sanity/schemaTypes/products.ts>
- **GitHub Repository:** <https://github.com/developerhammadrehman/template1/tree/main>

Steps for Day 3

1. Understand the Provided API:

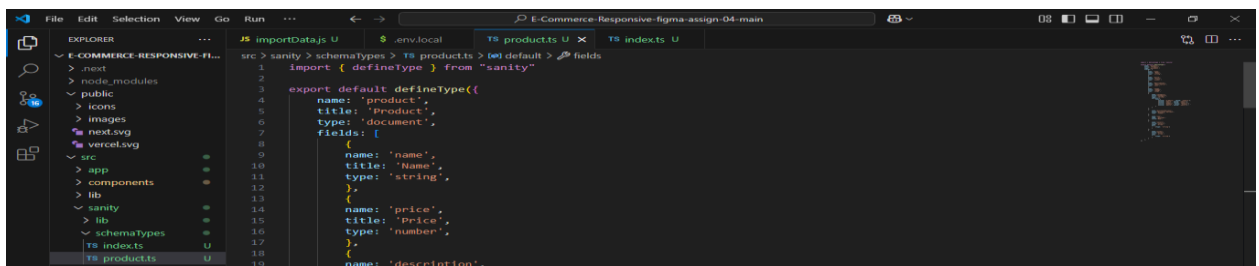
- Reviewed API documentation for Template 1.
- Key endpoints identified:
 - **Product Listings:** /products
 - **Categories:** /categories



2. Validate and Adjust Your Schema:

- Compared existing Sanity CMS schema with API data structure.
- Adjustments made to schema fields to ensure compatibility:
- Created a file in schema types>Product.ts

Product.ts Snippet:



3. Data Migration Steps and Tools Used

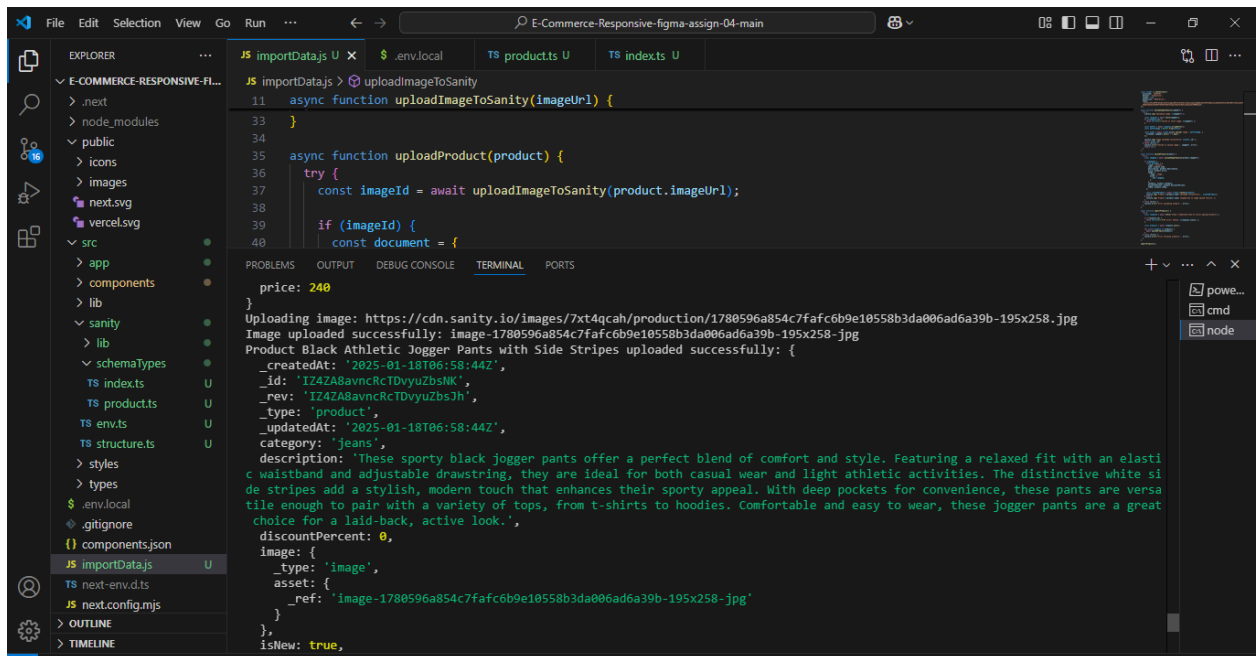
Tools Used:

- **Sanity Client:** Used for interacting with the Sanity CMS.
- **Node.js and Fetch API:** Used to fetch data from an external API and uploaded it to Sanity.
- **Env.:** For storing sensitive data.

Steps:

1. **Environment setup:** Installed dependencies and configured environment variables.
2. **Sanity Client Setup:** Generated a project id and token.
3. **Addition of ID and Token:** Added ID and Token in env.local .
4. **ImportData.js:** Created a file named ImportData.js to import the data from the /api/products (provided API) endpoint into my Sanity project. Also provided the ID and Token in the code.
5. **Run the Command:** then in cmd I run this command which fetched the data. (node importData.js)
6. **Verify Data:** After migration, I verified data in sanity to ensure the migration is successful.

API Response in console:

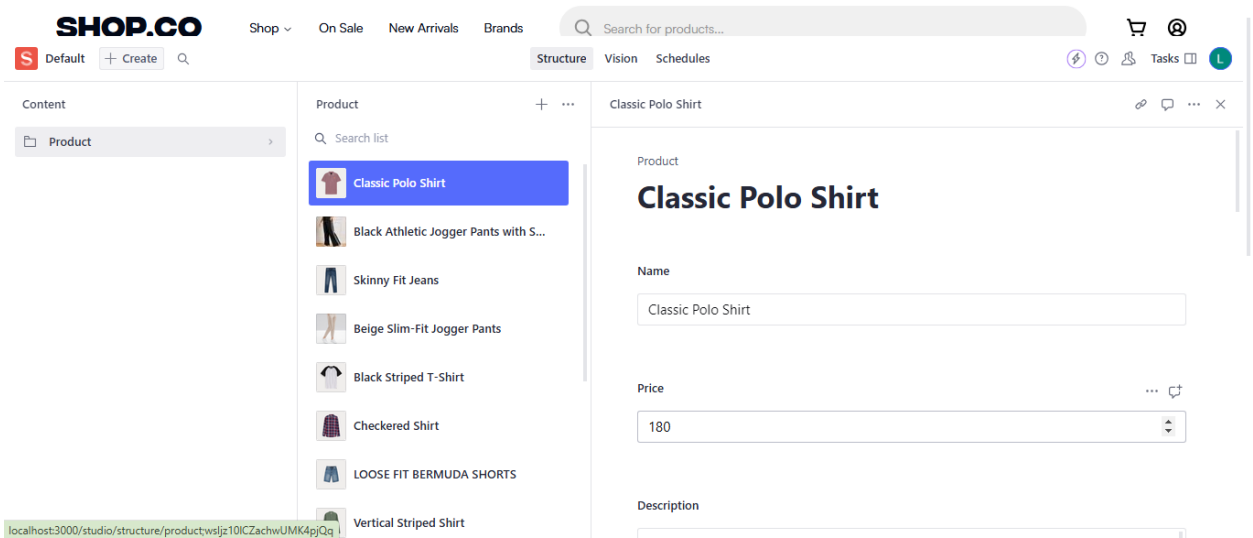


The screenshot shows a VS Code editor with a project named 'E-Commerce-Responsive-figma-assign-04-main'. The Explorer panel on the left shows the file structure, including 'src' and 'sanity' folders. The main editor displays a TypeScript file 'importData.js' with code for uploading images and products to Sanity. The Terminal panel at the bottom shows the output of the command 'node importData.js', which includes the following JSON response:

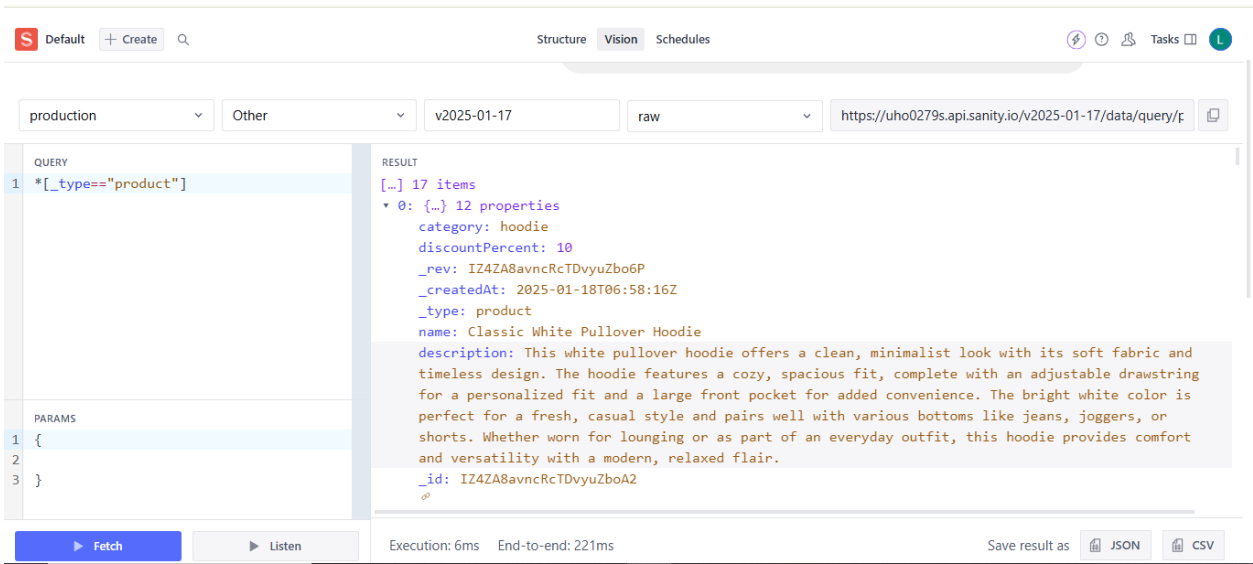
```
{
  "price": 240,
  "description": "These sporty black jogger pants offer a perfect blend of comfort and style. Featuring a relaxed fit with an elastic waistband and adjustable drawstring, they are ideal for both casual wear and light athletic activities. The distinctive white side stripes add a stylish, modern touch that enhances their sporty appeal. With deep pockets for convenience, these pants are versatile enough to pair with a variety of tops, from t-shirts to hoodies. Comfortable and easy to wear, these jogger pants are a great choice for a laid-back, active look.",
  "discountPercent": 0,
  "image": {
    "_type": "image",
    "asset": {
      "_ref": "image-1780596a854c7fafc6b9e10558b3da006ad6a39b-195x258-jpg"
    }
  },
  "isNew": true,
  "_id": "I242A8avncRcTDvyyuZbs3h",
  "_rev": "I242A8avncRcTDvyyuZbs3h",
  "_type": "product",
  "_updatedAt": "2025-01-18T06:58:44Z",
  "_createdAt": "2025-01-18T06:58:44Z"
}
```

The terminal also shows the image upload URL: 'https://cdn.sanity.io/images/7xt4qcah/production/1780596a854c7fafc6b9e10558b3da006ad6a39b-195x258-jpg'.

Sanity Response:



Groq Query:



NextJs(Frontend):

