Day 03 Goal:

Integrating Mock API an Migrating Data to Sanity CMS

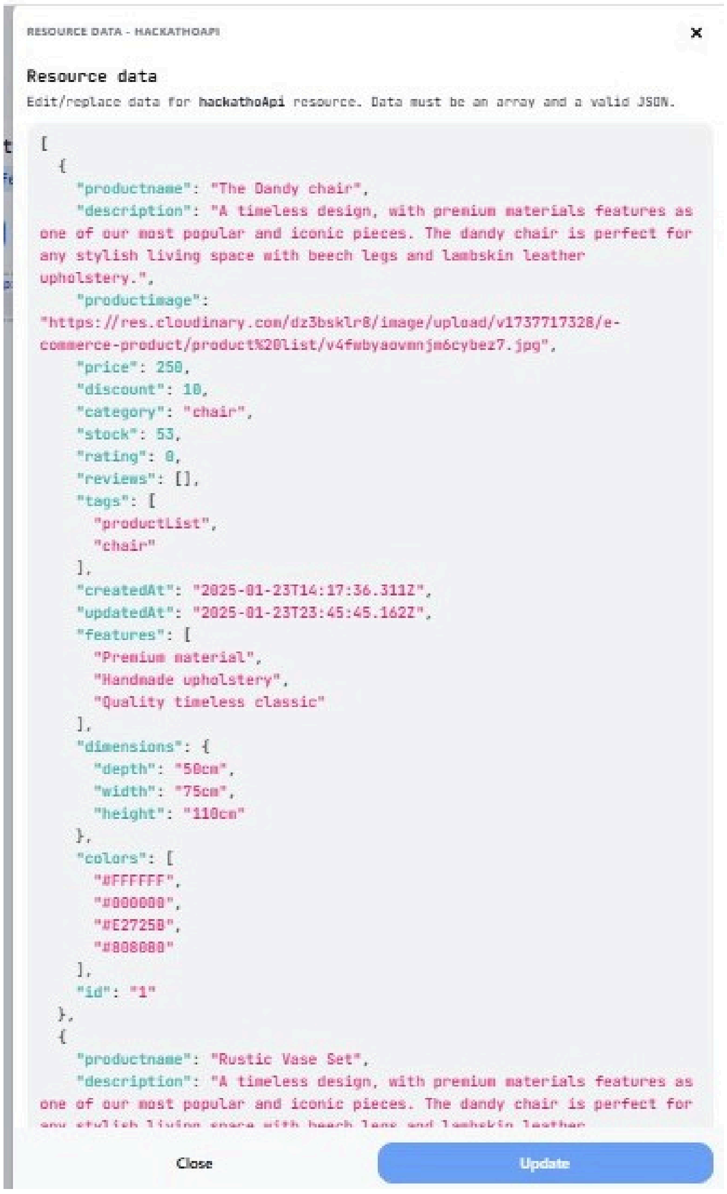
The primary objective of Day 03 is to integrate a Mock API into your project and migrate its data to Sanity CMS. This involves setting up the structure of the Mock API and aligning it with the product schema in Sanity.

Key Sections to On:

---

1. Understanding the API

The first step is to determine the data needed for our marketplace. While it's possible to add data directly through Sanity Studio, using a Mock API offers several advantages, especially when dealing with bundled datasets. This external service will help us create an API, which will then be integrated into the project. Once integrated, the data will be migrated into Sanity for better management and scalability.



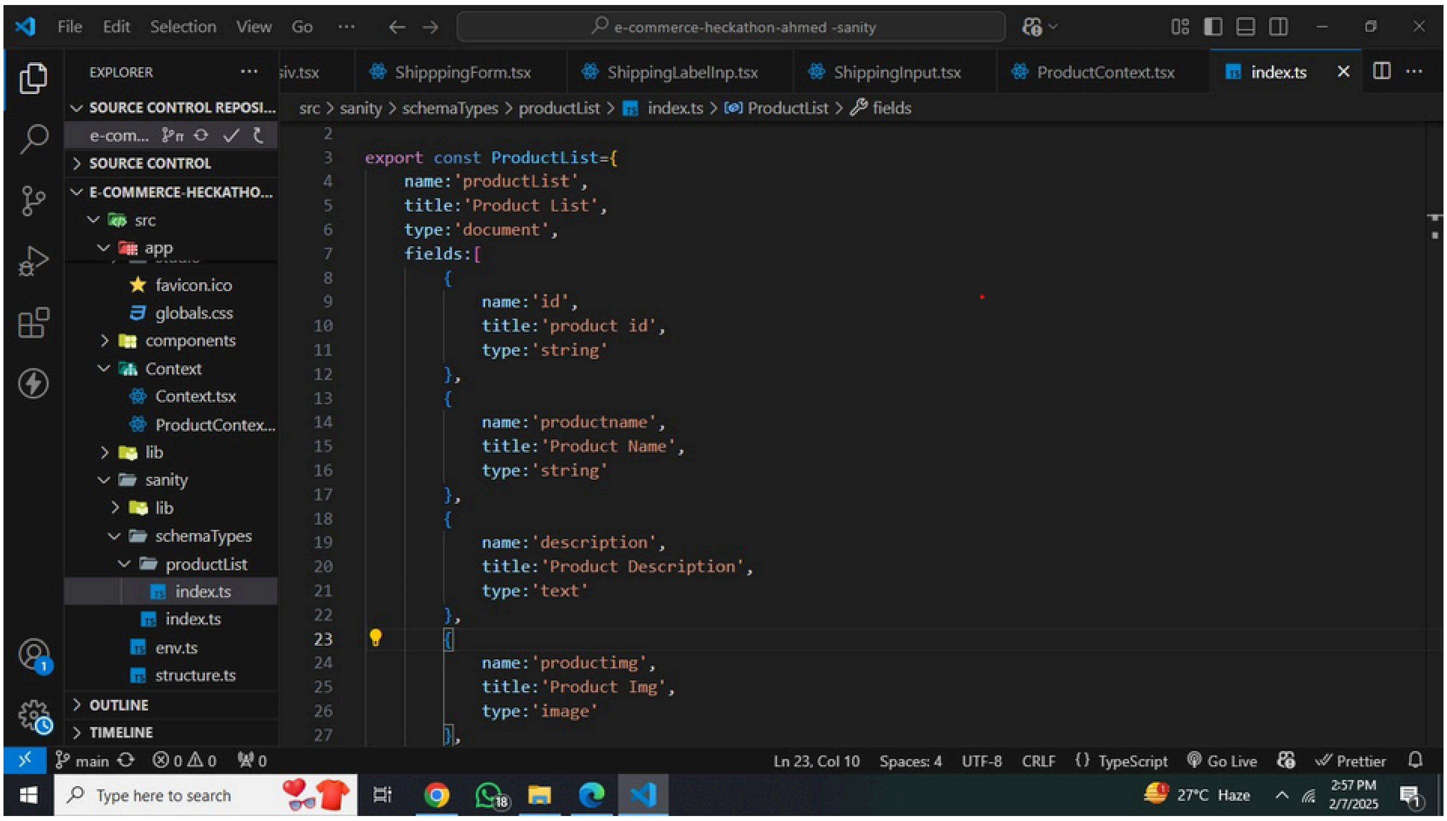
Resource data

Edit/replace data for **hackathoApi** resource. Data must be an array and a valid JSON.

```json
[
  {
    "productname": "The Dandy chair",
    "description": "A timeless design, with premium materials features as
one of our most popular and iconic pieces. The dandy chair is perfect for
any stylish living space with beech legs and lambskin leather
upholstery.",
    "productimage":
"https://res.cloudinary.com/dz3bsklr8/image/upload/v1737717328/e-
commerce-product/product%20list/v4fwbyaovmnjm6cybez7.jpg",
    "price": 250,
    "discount": 10,
    "category": "chair",
    "stock": 53,
    "rating": 0,
    "reviews": [],
    "tags": [
      "productList",
      "chair"
    ],
    "createdAt": "2025-01-23T14:17:36.311Z",
    "updatedAt": "2025-01-23T23:45:45.162Z",
    "features": [
      "Premium material",
      "Handmade upholstery",
      "Quality timeless classic"
    ],
    "dimensions": {
      "depth": "50cm",
      "width": "75cm",
      "height": "110cm"
    },
    "colors": [
      "#FFFFFF",
      "#000000",
      "#E27250",
      "#808080"
    ],
    "id": "1"
  },
  {
    "productname": "Rustic Vase Set",
    "description": "A timeless design, with premium materials features as
one of our most popular and iconic pieces. The dandy chair is perfect for
any stylish living space with beech legs and lambskin leather
```

2. Adjusting the Schema

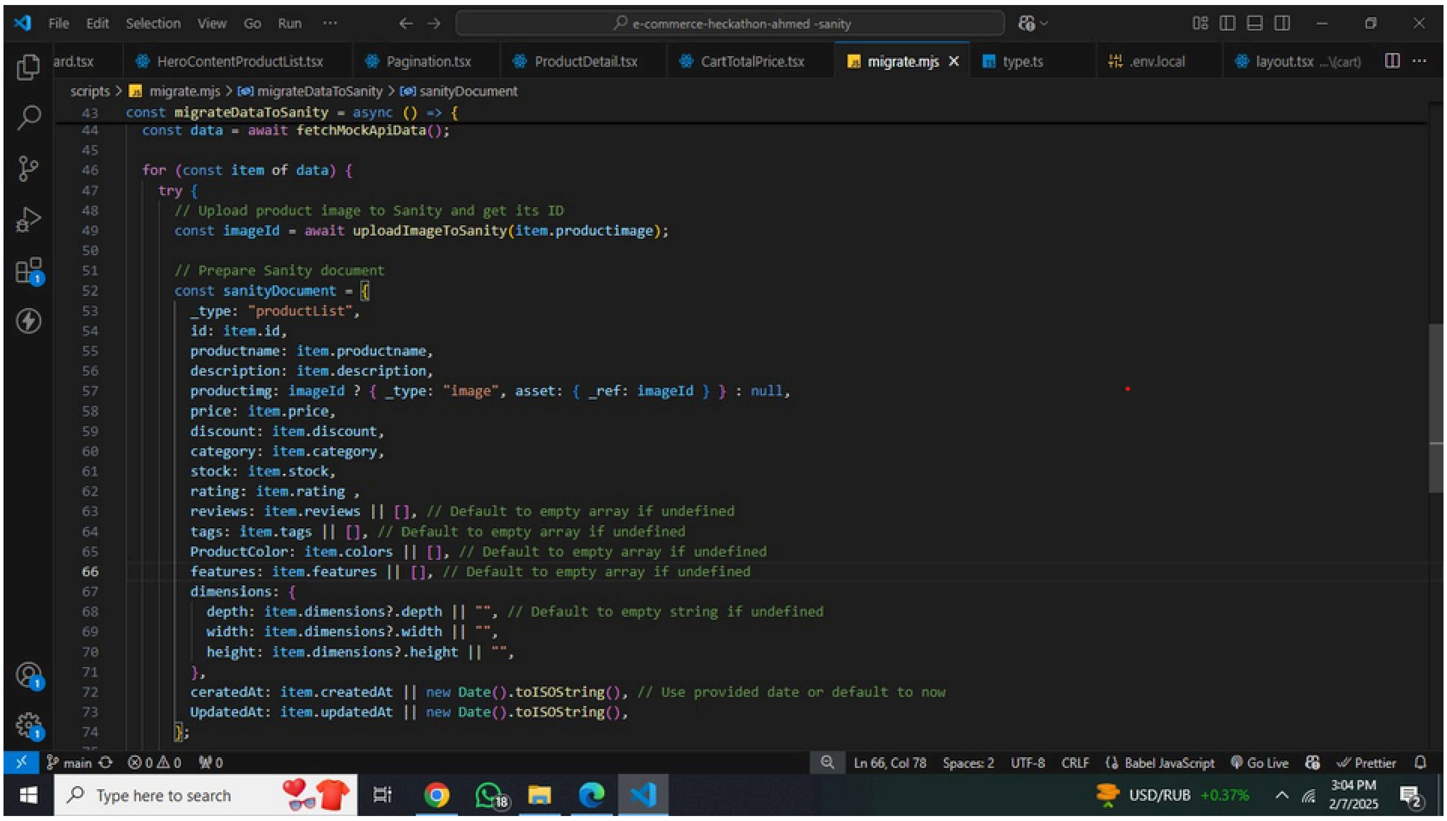
After setting up the Mock API, we need to adjust our schema to match the structure of the data returned by the API. The schema will include properties such as productName, productImage, and productPrice. To manage this, we need to install Sanity in our marketplace project and follow the provided instructions to configure and work with Sanity CMS.



```
export const ProductList={
    name:'productList',
    title:'Product List',
    type:'document',
    fields:[
        {
            name:'id',
            title:'product id',
            type:'string'
        },
        {
            name:'productname',
            title:'Product Name',
            type:'string'
        },
        {
            name:'description',
            title:'Product Description',
            type:'text'
        },
        {
            name:'productimg',
            title:'Product Img',
            type:'image'
        },
```

3. Migrating API Data into Sanity

Once the API is set up, the next step is to migrate the data to Sanity. This requires configuring the Mock API and Sanity keys in the .env.local file. For fetching the Mock API, we use axios, and for migrating the data into Sanity Studio, we utilize Sanity's migration methods. We create a migrate.mjs file for this operation and place it in the scripts folder of the project.


File Edit Selection View Go Run
e-commerce-heckathon-ahmed -sanity
ard.tsx
HeroContentProductList.tsx
Pagination.tsx
ProductDetail.tsx
CartTotalPrice.tsx
migrate.mjs
type.ts
.env.local
layout.tsx ...\(cart)
scripts > migrate.mjs > migrateDataToSanity > sanityDocument
main
Ln 66, Col 78 Spaces: 2 UTF-8 CRLF Babel JavaScript Go Live Prettier
Type here to search
USD/RUB +0.37%
3:04 PM
2/7/2025

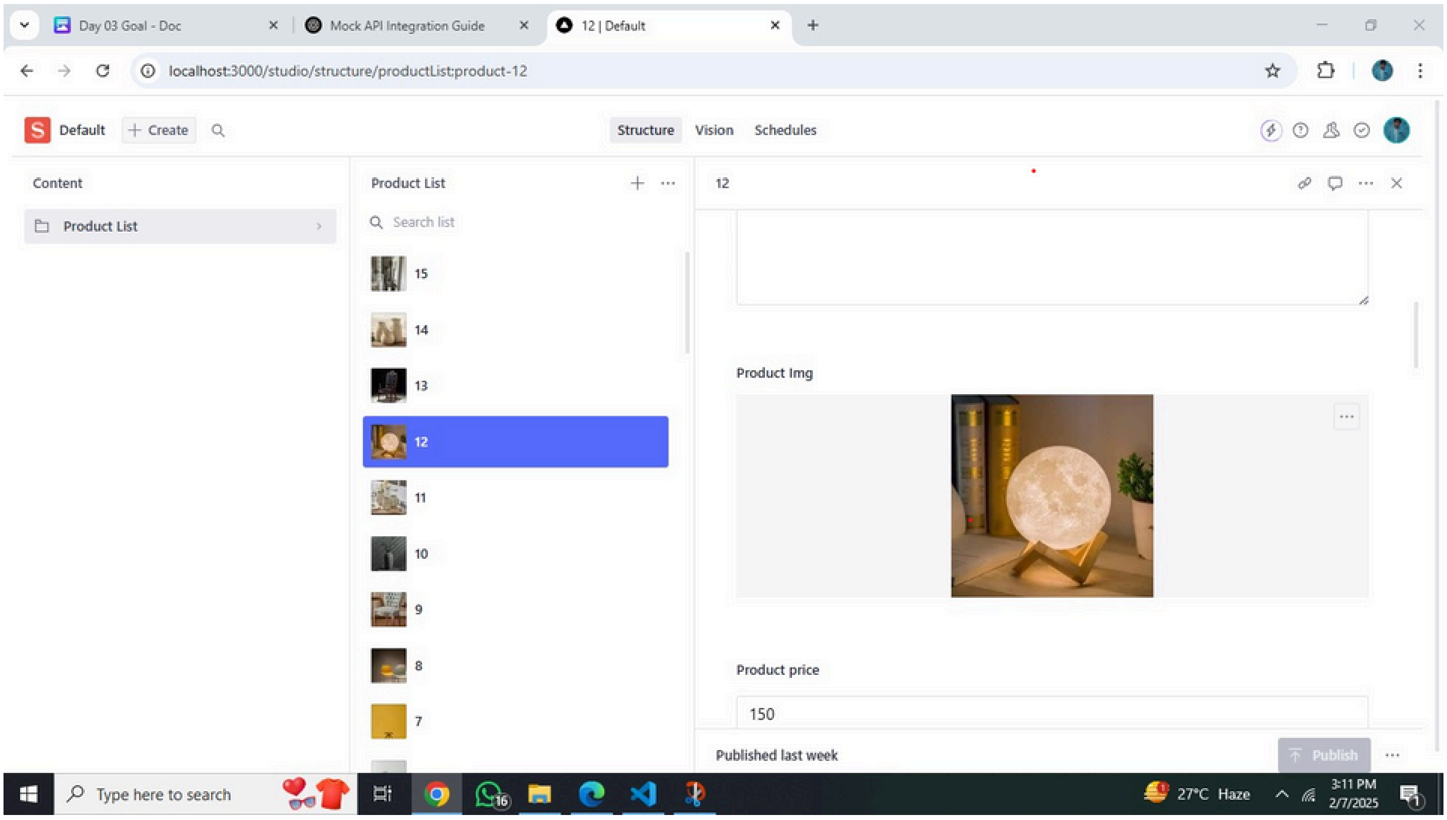
```
const migrateDataToSanity = async () => {
  const data = await fetchMockApiData();

  for (const item of data) {
    try {
      // Upload product image to Sanity and get its ID
      const imageId = await uploadImageToSanity(item.productimage);

      // Prepare Sanity document
      const sanityDocument = {
        _type: "productList",
        id: item.id,
        productname: item.productname,
        description: item.description,
        productimg: imageId ? { _type: "image", asset: { _ref: imageId } } : null,
        price: item.price,
        discount: item.discount,
        category: item.category,
        stock: item.stock,
        rating: item.rating ,
        reviews: item.reviews || [], // Default to empty array if undefined
        tags: item.tags || [], // Default to empty array if undefined
        ProductColor: item.colors || [], // Default to empty array if undefined
        features: item.features || [], // Default to empty array if undefined
        dimensions: {
          depth: item.dimensions?.depth || "", // Default to empty string if undefined
          width: item.dimensions?.width || "",
          height: item.dimensions?.height || "",
        },
        ceratedAt: item.createdAt || new Date().toISOString(), // Use provided date or default to now
        UpdatedAt: item.updatedAt || new Date().toISOString(),
      };
```

4. Migration Completed

Once the migration is complete, the Sanity Studio will display all the product data and details from the Mock API, confirming that the migration was successful. This allows us to easily manage and update the marketplace data through Sanity.



Product Img

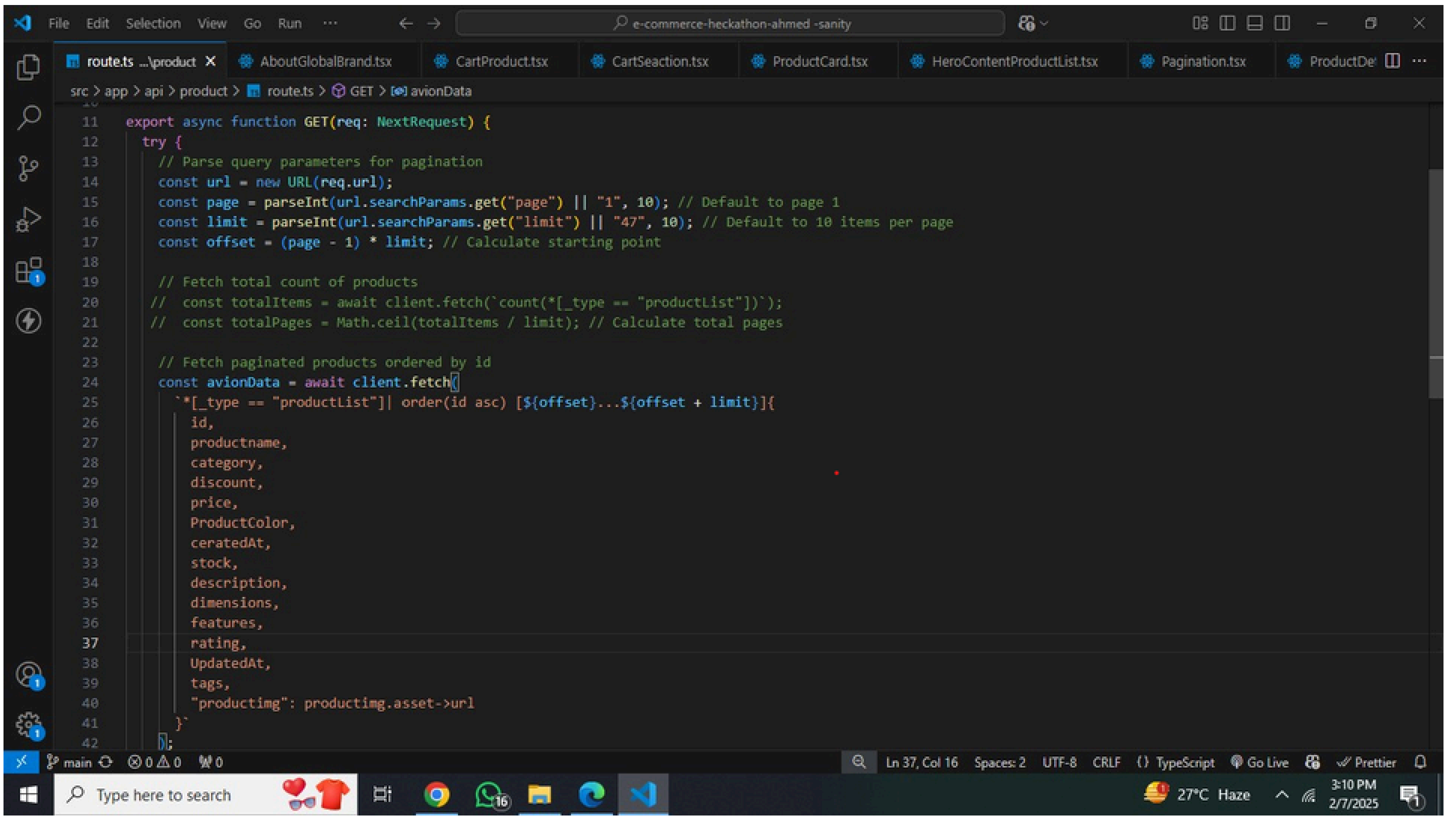
Product price

150

Published last week

5. Integrating Data in Next.js

With the API data successfully migrated into Sanity, the next step is to integrate it into our Next.js project. We use the GROQ query language to fetch the data, leveraging the client.fetch() method from Sanity. This allows us to retrieve and display the product information dynamically in our marketplace.



```
export async function GET(req: NextRequest) {
  try {
    // Parse query parameters for pagination
    const url = new URL(req.url);
    const page = parseInt(url.searchParams.get("page") || "1", 10); // Default to page 1
    const limit = parseInt(url.searchParams.get("limit") || "47", 10); // Default to 10 items per page
    const offset = (page - 1) * limit; // Calculate starting point

    // Fetch total count of products
  //  const totalItems = await client.fetch(`count(*[_type == "productList"])`);
  //  const totalPages = Math.ceil(totalItems / limit); // Calculate total pages

    // Fetch paginated products ordered by id
    const avionData = await client.fetch(
      `*[_type == "productList"]| order(id asc) [${offset}...${offset + limit}]{
        id,
        productname,
        category,
        discount,
        price,
        ProductColor,
        ceratedAt,
        stock,
        description,
        dimensions,
        features,
        rating,
        UpdatedAt,
        tags,
        "productimg": productimg.asset->url
      }`
    );
```

6. Challenges Faced

Throughout the process, I encountered a few challenges, especially with the data migration into Sanity. Initially, I underestimated the complexity of the migration process, assuming it would be short and simple. Additionally, handling image data posed some difficulties, but after troubleshooting, everything was successfully integrated.