HACKATHON

Day 03

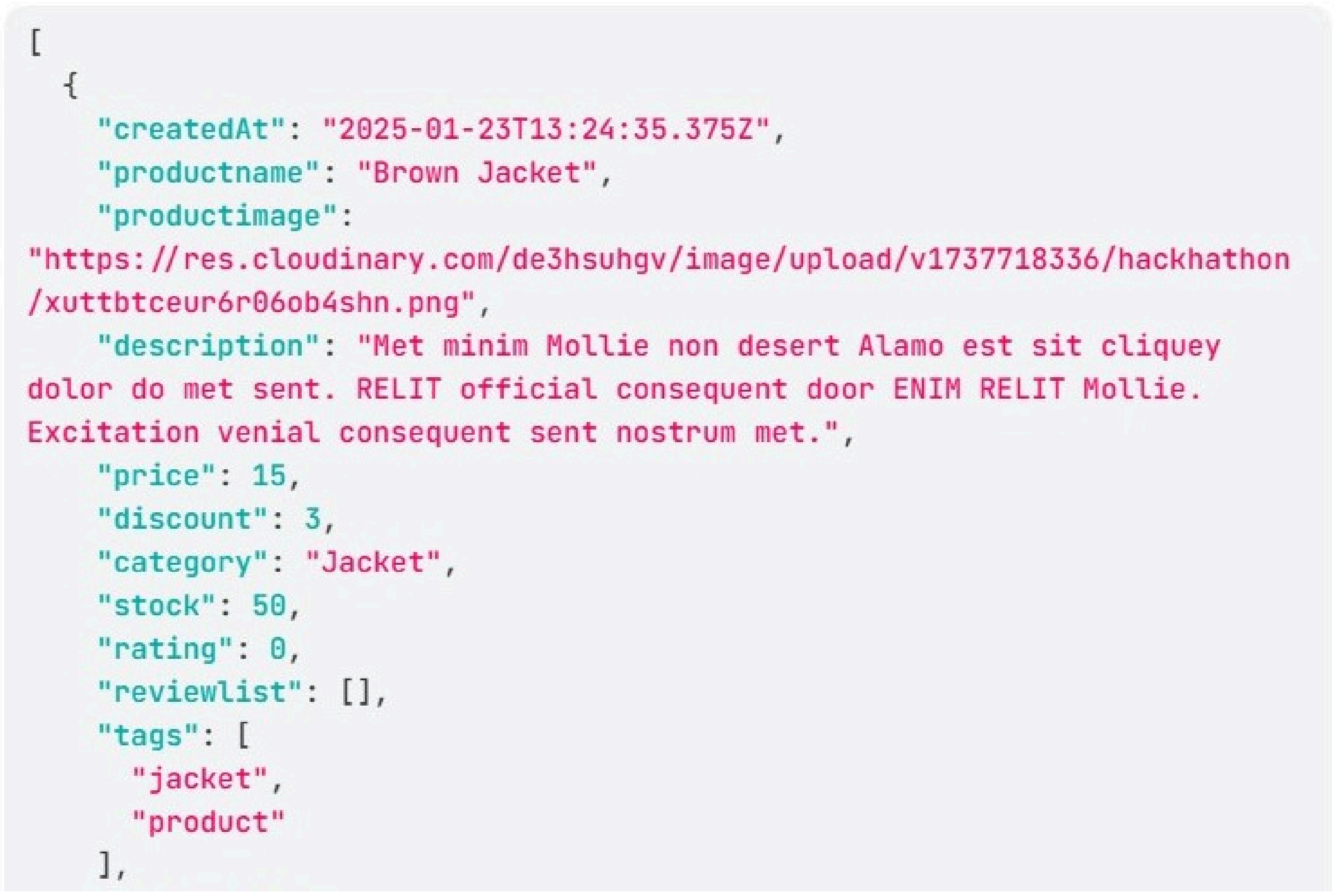
The goal for **Day 03** is to integrate a **Mock API** into your project and migrate the data to **Sanity CMS**.
In this phase, you'll learn how to set up the structure of the Mock API effectively. Additionally, you'll work on configuring the **product schema** to ensure smooth data migration from the Mock API to Sanity CMS.

Key Sections to Focus On:

API Understanding:

To develop our marketplace, the first step is to identify the type of data we need. This data will be provided through an **API**, which we'll create and later migrate to **Sanity CMS**.

Now, you might wonder, *"Why create an API when we can directly add data through Sanity Studio?"*
Yes, for a small amount of data, that's possible. However, when dealing with larger datasets or multiple data bundles, we need an **external service** to efficiently develop the API. This API can then be integrated into our project and seamlessly migrated to Sanity CMS.

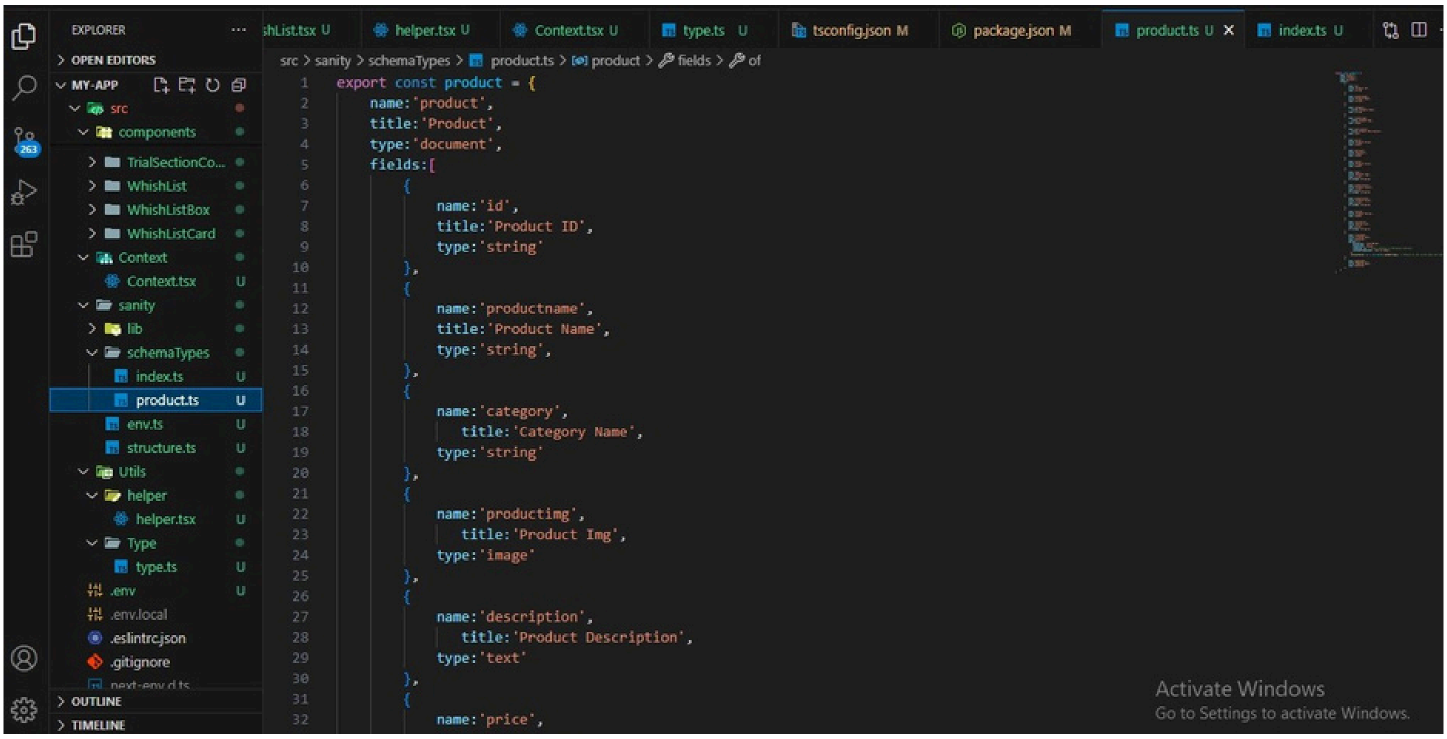


```
[
  {
    "createdAt": "2025-01-23T13:24:35.375Z",
    "productname": "Brown Jacket",
    "productimage":
"https://res.cloudinary.com/de3hsuhgv/image/upload/v1737718336/hackathon
/xuttbtceur6r06ob4shn.png",
    "description": "Met minim Mollie non desert Alamo est sit cliquey
dolor do met sent. RELIT official consequent door ENIM RELIT Mollie.
Excitation venial consequent sent nostrum met.",
    "price": 15,
    "discount": 3,
    "category": "Jacket",
    "stock": 50,
    "rating": 0,
    "reviewlist": [],
    "tags": [
      "jacket",
      "product"
    ],
```

Schema Adjustment:

After setting up the API, the next step is to **adjust the schema** according to the API structure. This involves defining properties like **productName**, **productImage**, and **productPrice** to ensure the data is organized correctly.

To achieve this, we need to **install Sanity** into our marketplace project and follow the necessary instructions to configure and work with Sanity CMS effectively.



```
export const product = {
    name:'product',
    title:'Product',
    type:'document',
    fields:[
        {
            name:'id',
            title:'Product ID',
            type:'string'
        },
        {
            name:'productname',
            title:'Product Name',
            type:'string',
        },
        {
            name:'category',
              title:'Category Name',
            type:'string'
        },
        {
            name:'productimg',
              title:'Product Img',
            type:'image'
        },
        {
            name:'description',
              title:'Product Description',
            type:'text'
        },
        {
            name:'price',
```

Migrating API to Sanity:

The next step is to **migrate the API** into **Sanity**. To perform this operation, follow these key steps:

1. Configure Environment Variables:

- Add your **Mock API** and **Sanity keys** to the .env.local file, and also in the .env file if needed. This setup ensures secure data migration into Sanity.

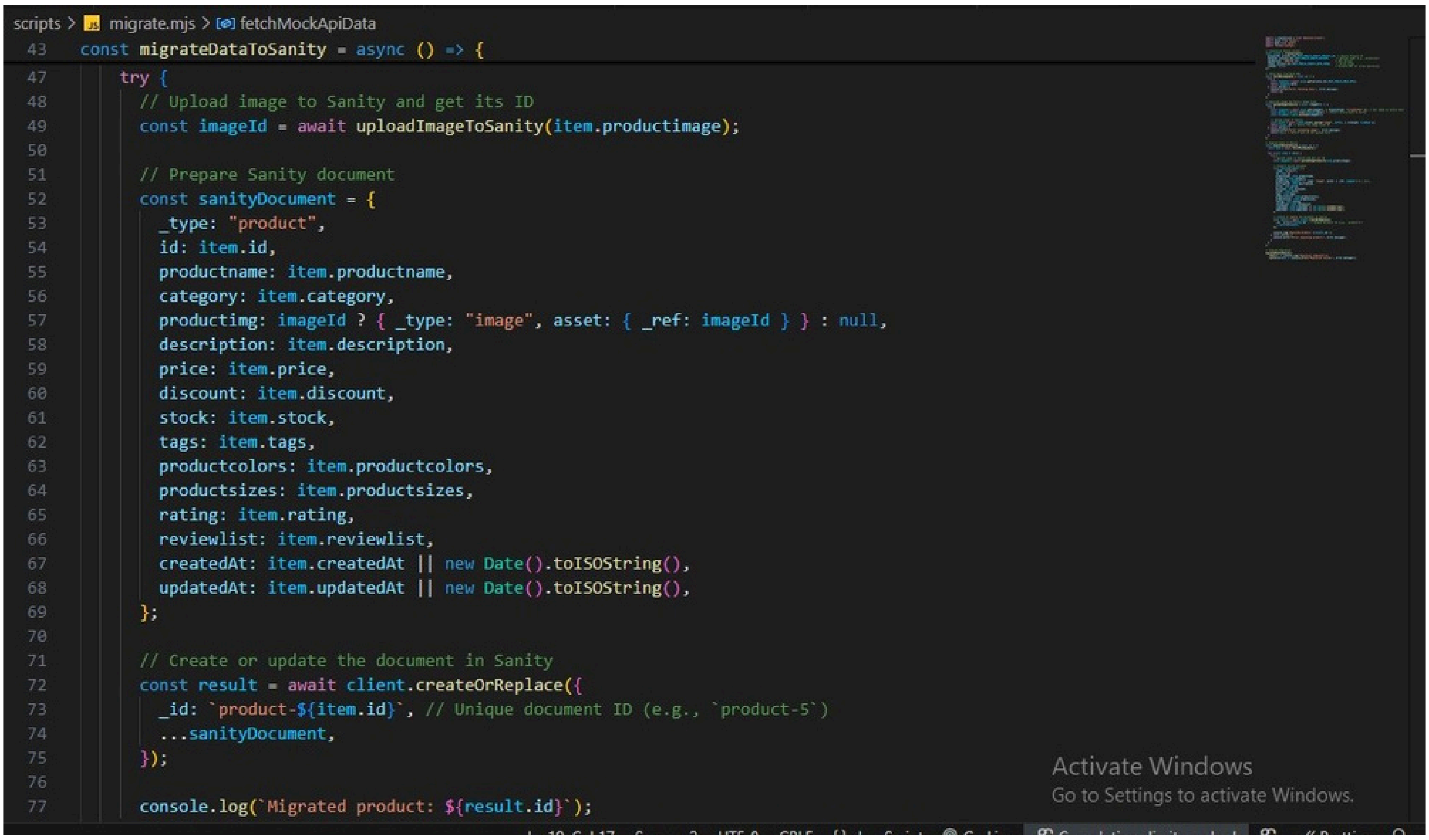
2. Fetching Data:

- Use **Axios** to fetch data from the Mock API.
- Utilize specific **Sanity methods** to handle the data migration into **Sanity Studio**.

3. Migration Script:

- Create a file named migrate.mjs to manage the entire migration process.
- Place this file inside the **scripts folder** within your project for better organization.

By following these steps, you'll be able to smoothly migrate data from the Mock API into Sanity CMS.


scripts > migrate.mjs > fetchMockApiData

```
const migrateDataToSanity = async () => {
    try {
      // Upload image to Sanity and get its ID
      const imageId = await uploadImageToSanity(item.productimage);

      // Prepare Sanity document
      const sanityDocument = {
        _type: "product",
        id: item.id,
        productname: item.productname,
        category: item.category,
        productimg: imageId ? { _type: "image", asset: { _ref: imageId } } : null,
        description: item.description,
        price: item.price,
        discount: item.discount,
        stock: item.stock,
        tags: item.tags,
        productcolors: item.productcolors,
        productsizes: item.productsizes,
        rating: item.rating,
        reviewlist: item.reviewlist,
        createdAt: item.createdAt || new Date().toISOString(),
        updatedAt: item.updatedAt || new Date().toISOString(),
      };

      // Create or update the document in Sanity
      const result = await client.createOrReplace({
        _id: `product-${item.id}`, // Unique document ID (e.g., `product-5`)
        ...sanityDocument,
      });

      console.log(`Migrated product: ${result.id}`);
```

Activate Windows
Go to Settings to activate Windows.

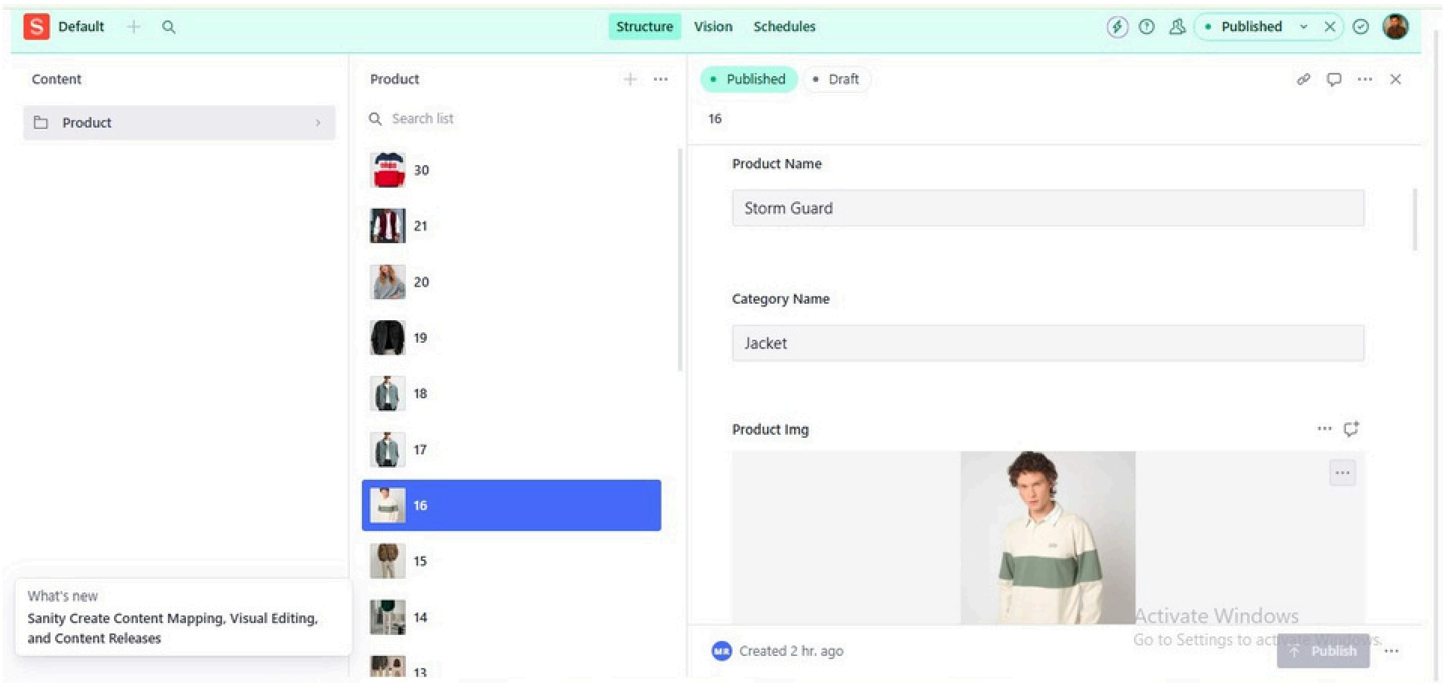
Running the Migration Script:

To run the migration script, use the following command in your terminal:

```
npm run migrate
```

Migration Completed:

Once the **Mock API** data has been successfully migrated to **Sanity Studio**, you'll be able to see all the products along with their details exactly as defined in the Mock API. This confirms that the **migration process** has been completed successfully.

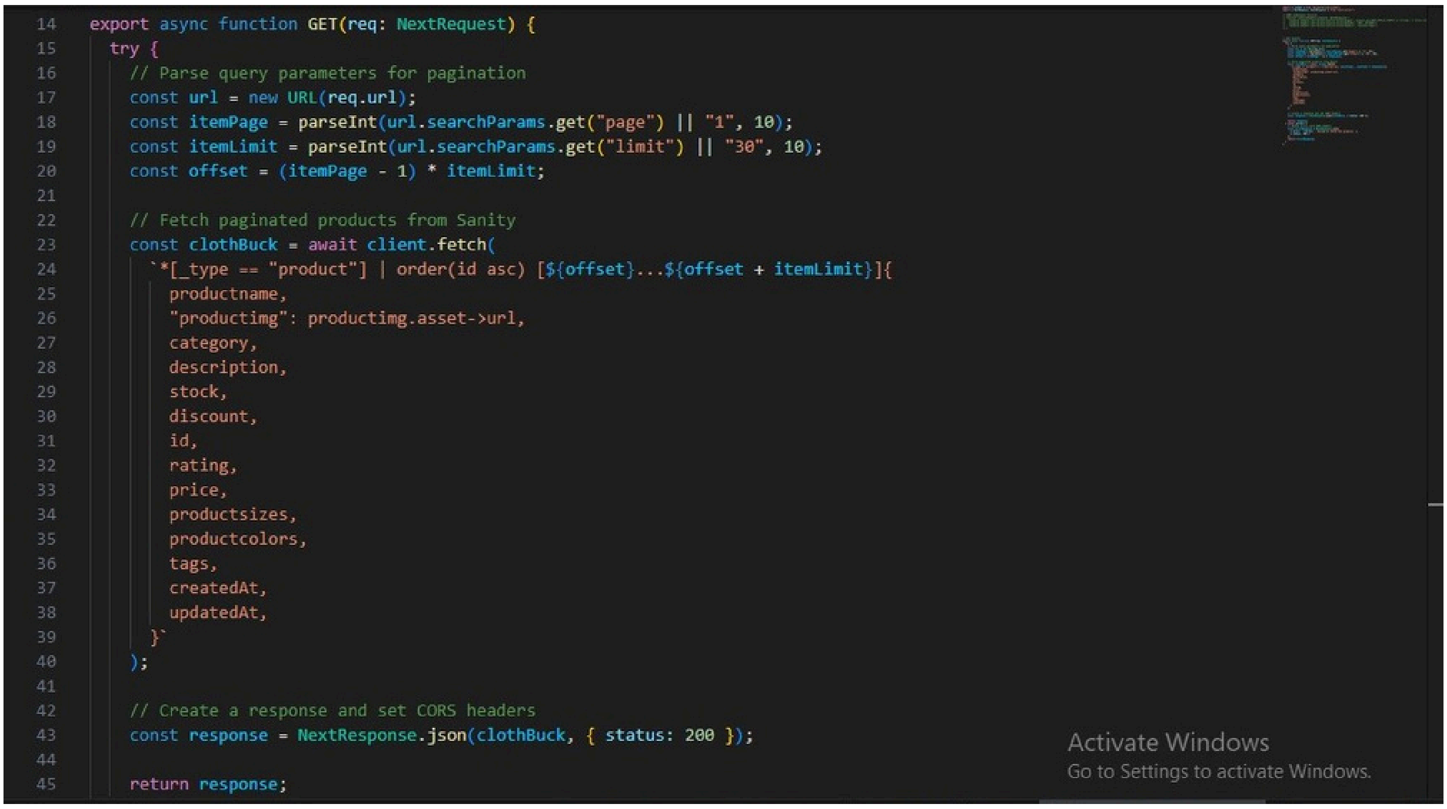


Integrate in Next.js:

After the API data is successfully migrated to **Sanity Studio**, the next step is to integrate the Sanity data into **Next.js**.

We do this by writing queries using **GROQ** and fetching the data with the client.fetch() method from Sanity.

Here, I have integrated the data using this method.



```
export async function GET(req: NextRequest) {
  try {
    // Parse query parameters for pagination
    const url = new URL(req.url);
    const itemPage = parseInt(url.searchParams.get("page") || "1", 10);
    const itemLimit = parseInt(url.searchParams.get("limit") || "30", 10);
    const offset = (itemPage - 1) * itemLimit;

    // Fetch paginated products from Sanity
    const clothBuck = await client.fetch(
      `*[_type == "product"] | order(id asc) [${offset}...${offset + itemLimit}]{
        productname,
        "productimg": productimg.asset->url,
        category,
        description,
        stock,
        discount,
        id,
        rating,
        price,
        productsizes,
        productcolors,
        tags,
        createdAt,
        updatedAt,
      }`
    );

    // Create a response and set CORS headers
    const response = NextResponse.json(clothBuck, { status: 200 });

    return response;
```

Challenges Faced:

I faced some challenges because I didn’t know how to migrate data to **Sanity**. I thought the migration process would be short, but it wasn’t as simple as I expected.

I also faced issues while handling images, but after some effort, it worked successfully.