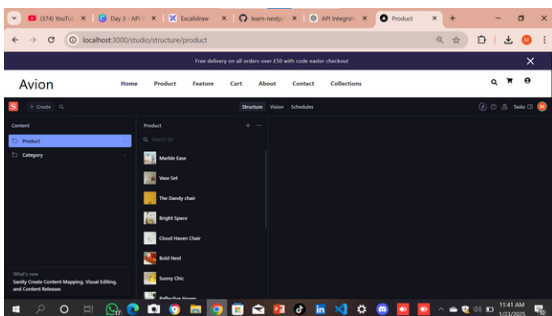


# Day 3 – API Integration Report – Avion

## API Integration Process

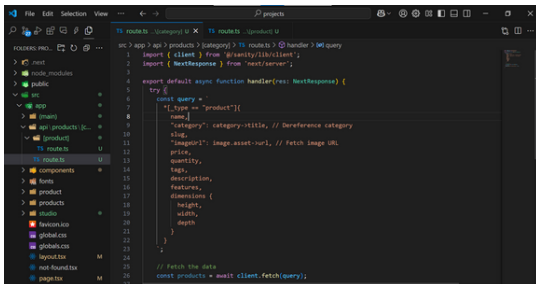
### API Data Migration to Sanity

- Developed a migration script to fetch data from the external API.
- Used Axios to handle HTTP requests and responses.
- Transformed the fetched data into a format compatible with Sanity schemas.
- Populated the Sanity CMS database using the Sanity client library.



## Fetching Data in Next.js

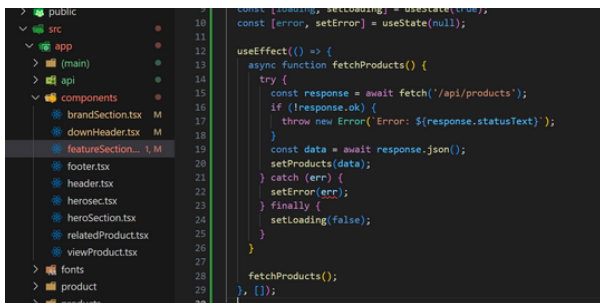
- Created API routes in Next.js to query the Sanity CMS using GROQ queries.
- Utilized the `@sanity/client` library for connecting to the Sanity API.
- Served the queried data to frontend components through these



```
1 import { client } from '@sanity/lib/client';
2 import { NextResponse } from 'next/server';
3
4 export default async function handler(req: NextResponse) {
5   try {
6     const query = `
7       *[_type == "product"]{
8         "category": category-title, // dereference category
9         slug,
10        "imageurl": image.asset-url, // fetch image url
11        price,
12        quantity,
13        tags,
14        description,
15        features,
16        dimensions {
17          height,
18          width,
19          depth
20        }
21      }
22    `;
23
24    // Fetch the data
25    const products = await client.fetch(query);
26  } catch (error) {
27    // Handle error
28  }
29 }
```

## Frontend Component Integration

- Implemented `useEffect` hooks to call the Next.js API routes and fetch data asynchronously.
- Managed state using React's `useState` for rendering the fetched data dynamically in the UI.



```
10 const [loading, setloading] = useState(true);
11 const [error, setError] = useState(null);
12
13 useEffect(() => {
14   async function fetchProducts() {
15     try {
16       const response = await fetch('/api/products');
17       if (!response.ok) {
18         throw new Error('Error: ${response.statusText}');
19       }
20       const data = await response.json();
21       setProducts(data);
22     } catch (err) {
23       setError(err);
24     } finally {
25       setloading(false);
26     }
27   }
28
29   fetchProducts();
30 }, []);
```

## Adjustments Made to Schemas

- Schema Customizations:
  - Added new fields to existing schemas to match the structure of the external API data.
  - Modified validation rules to ensure data integrity.
  - Adjusted references between schemas to support relational data from the API.
- Example: Updated schema for a "Product" document:

```
1  import { defineType, defineField } from "sanity"
2
3  export const product = defineType({
4    name: "product",
5    title: "Product",
6    type: "document",
7    fields: [
8      defineField({
9        name: "category",
10       title: "Category",
11       type: "reference",
12       to: [{
13         type: "category"
14       }]
15     }),
16     defineField({
17       name: "name",
18       title: "Title",
19       validation: (rule) => rule.required(),
20       type: "string"
21     }),
22     defineField({
23       name: "slug",
24       title: "Slug",
25       validation: (rule) => rule.required(),
26       type: "slug"
27     }),
28     defineField({
29       name: "image",
30       type: "image",
31       validation: (rule) => rule.required(),
32       title: "Product Image"
33     }),
34     defineField({
35       name: "price",
36       type: "number",
37       validation: (rule) => rule.required(),
38       title: "Price",
39     })
40   ]
41 })
```

```

41     defineField({
42       name: "quantity",
43       title: "Quantity",
44       type: "number",
45       validation: (rule) => rule.min(0),
46     }),
47     defineField({
48       name: "tags",
49       type: "array",
50       title: "Tags",
51       of: [{
52         type: "string"
53       }]
54     }),
55     defineField({
56       name: 'description',
57       title: 'Description',
58       type: 'text',
59       description: 'Detailed description of the product',
60     }),
61     defineField({
62       name: 'features',
63       title: 'Features',
64       type: 'array',
65       of: [{ type: 'string' }],
66       description: 'List of key features of the product',
67     }),
68     defineField({
69       name: 'dimensions',
70       title: 'Dimensions',
71       type: 'object',
72       fields: [
73         { name: 'height', title: 'Height', type: 'string' },
74         { name: 'width', title: 'Width', type: 'string' },
75         { name: 'depth', title: 'Depth', type: 'string' },
76       ],
77       description: 'Dimensions of the product',
78     }),
79   ]
80 })

```