

Day 3 – API Integration and Data Migration:

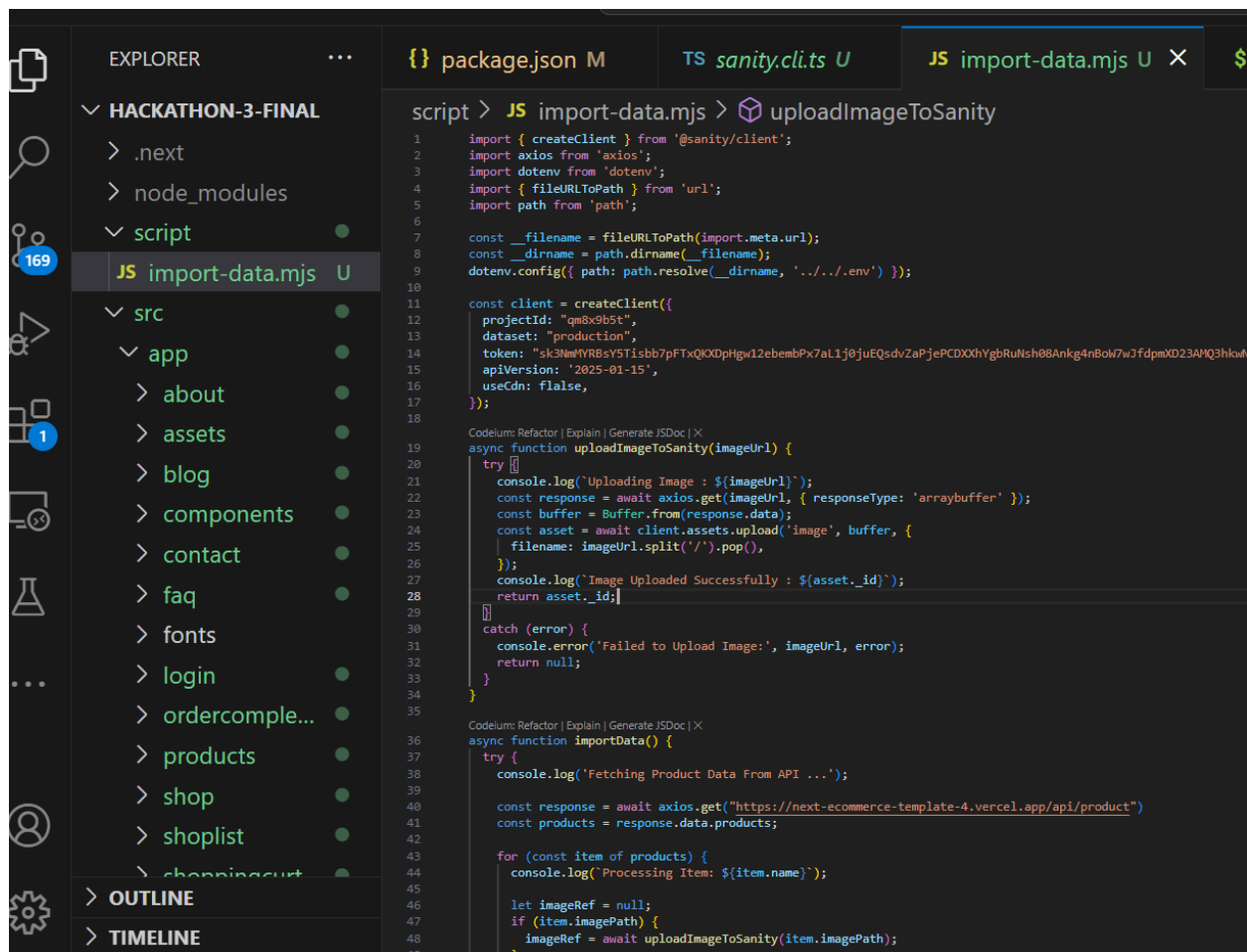
API Integration Process

1. Reviewed API Documentation

- Examined the provided API documentation to understand its structure and available endpoints.
- Identified the necessary endpoints for retrieving and managing data effectively.

2. Executed Data Migration Script

- Ran a custom script tailored to migrate data from the API into Sanity CMS.
- Confirmed the migration's success by verifying the populated fields in Sanity CMS.



The screenshot shows a VS Code editor with a dark theme. The Explorer sidebar on the left shows a project named 'HACKATHON-3-FINAL' with a file tree including '.next', 'node_modules', 'script', and 'src'. The 'script' folder is expanded, showing 'import-data.mjs'. The main editor area displays the content of 'import-data.mjs', which is a JavaScript script for migrating data from an API to Sanity CMS. The script includes imports for 'createClient', 'axios', 'dotenv', 'fileURLToPath', and 'path'. It defines a 'client' object with project details and an 'uploadImageToSanity' function that takes an 'imageUrl' and uploads it to Sanity. It also includes an 'importData' function that fetches product data from an API and processes each item by uploading its image to Sanity.

```
script > JS import-data.mjs > uploadImageToSanity
1  import { createClient } from '@sanity/client';
2  import axios from 'axios';
3  import dotenv from 'dotenv';
4  import { fileURLToPath } from 'url';
5  import path from 'path';
6
7  const __filename = fileURLToPath(import.meta.url);
8  const __dirname = path.dirname(__filename);
9  dotenv.config({ path: path.resolve(__dirname, '../../.env') });
10
11  const client = createClient({
12    projectId: "qm8x9b5t",
13    dataset: "production",
14    token: "sk3MmYR8sY5Tisbb7pFTxQKXDpHgw12ebembPx7aL1j0juEQsdvZaPjePCDXXhYgbRuNsh08Ankg4n8ok7w3fdpmXD23AMQ3hkw",
15    apiVersion: '2025-01-15',
16    useCdn: false,
17  });
18
19  Codium: Refactor | Explain | Generate JSDoc | X
20  async function uploadImageToSanity(imageUrl) {
21    try {
22      console.log('Uploading Image : ${imageUrl}');
23      const response = await axios.get(imageUrl, { responseType: 'arraybuffer' });
24      const buffer = Buffer.from(response.data);
25      const asset = await client.assets.upload('image', buffer, {
26        filename: imageUrl.split('/').pop(),
27      });
28      console.log('Image Uploaded Successfully : ${asset.id}');
29      return asset.id;
30    } catch (error) {
31      console.error('Failed to Upload Image:', imageUrl, error);
32      return null;
33    }
34  }
35
36  Codium: Refactor | Explain | Generate JSDoc | X
37  async function importData() {
38    try {
39      console.log('Fetching Product Data From API ...');
40
41      const response = await axios.get("https://next-ecommerce-template-4.vercel.app/api/product");
42      const products = response.data.products;
43
44      for (const item of products) {
45        console.log("Processing Item: ${item.name}");
46
47        let imageRef = null;
48        if (item.imagePath) {
49          imageRef = await uploadImageToSanity(item.imagePath);
50        }
51      }
52    } catch (error) {
53      console.error('Error importing data:', error);
54    }
55  }
```

next-ecommerce-template-4

> .next

> node_modules

> public

> script

JS import-data.mjs

> src

> app

> about

> assets

> blog

> components

> contact

> faq

> fonts

> login

> ordercompleted

> products

page.tsx

> shop

OUTLINE

TIMELINE

script > JS import-data.mjs > uploadImageToSanity

35

Codeium: Refactor | Explain | Generate JSDoc | X

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

async function importData() {

try {

console.log('Fetching Product Data From API ...');

const response = await axios.get("https://next-ecommerce-template-4.vercel.app/api/product");

const products = response.data.products;

for (const item of products) {

console.log(`Processing Item: \${item.name}`);

let imageRef = null;

if (item.imagePath) {

imageRef = await uploadImageToSanity(item.imagePath);

}

const sanityItem = {

_type: 'product',

name: item.name,

category: item.category || null,

price: item.price,

description: item.description || '',

discountPercentage: item.discountPercentage || 0,

stockLevel: item.stockLevel || 0,

isFeaturedProduct: item.isFeaturedProduct,

image: imageRef

? {

_type: 'image',

asset: {

_type: 'reference',

_ref: imageRef,

},

}

: undefined,

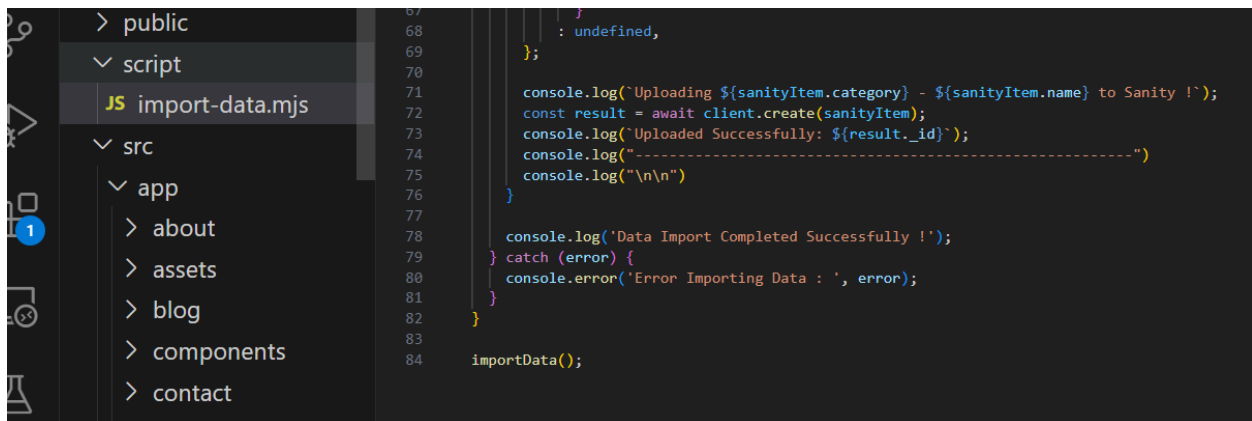
};

console.log(`Uploading \${sanityItem.category} - \${sanityItem.name} to Sanity !`);

const result = await client.create(sanityItem);

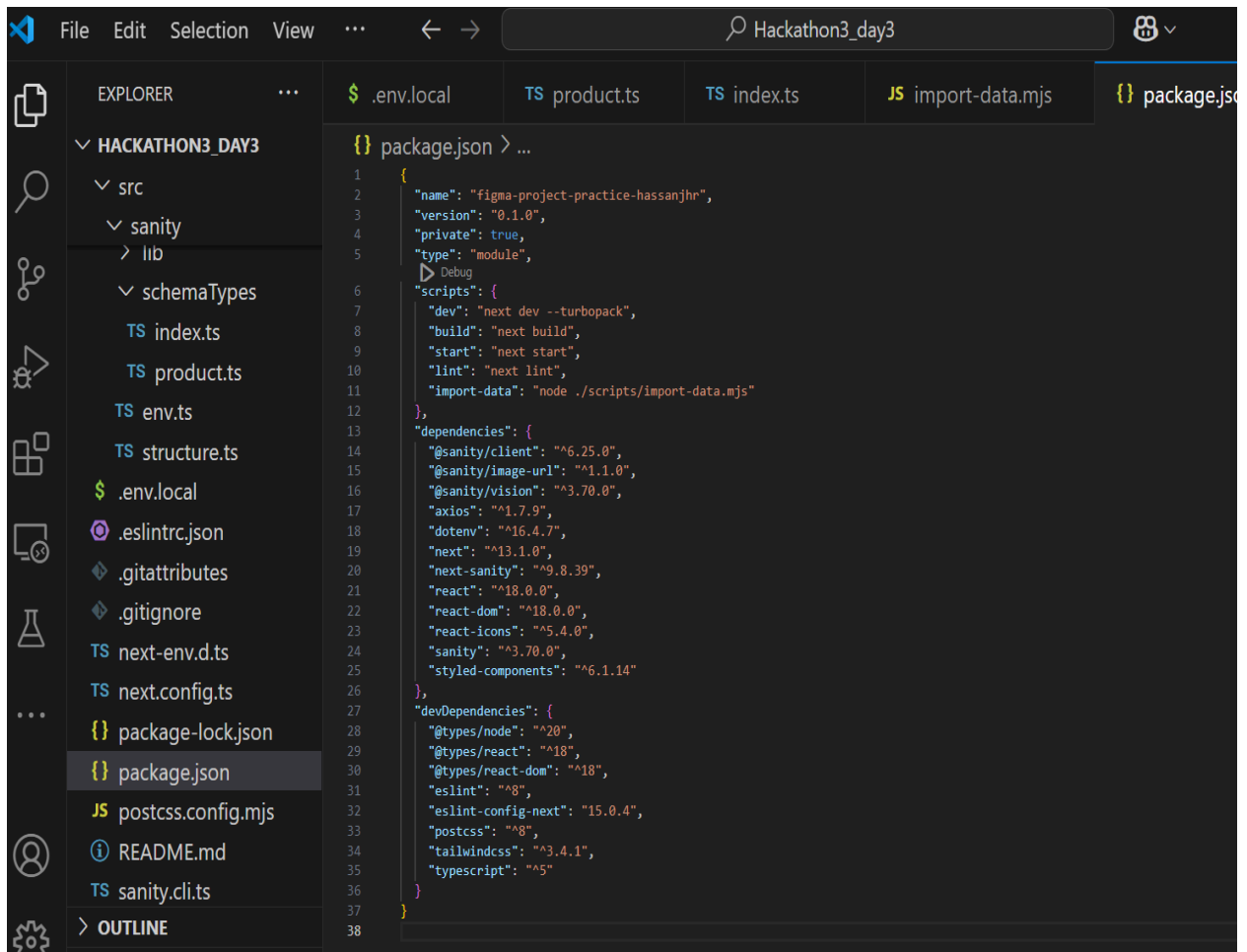
console.log(`Uploaded Successfully: \${result._id}`);

console.log("-----")



```
67 |         }
68 |         : undefined,
69 |     };
70 |
71 |     console.log(`Uploading ${sanityItem.category} - ${sanityItem.name} to Sanity !`);
72 |     const result = await client.create(sanityItem);
73 |     console.log(`Uploaded Successfully: ${result._id}`);
74 |     console.log("-----")
75 |     console.log("\n\n")
76 | }
77 |
78 | console.log("Data Import Completed Successfully !");
79 | } catch (error) {
80 |     console.error('Error Importing Data : ', error);
81 | }
82 |
83 |
84 | importData();
```

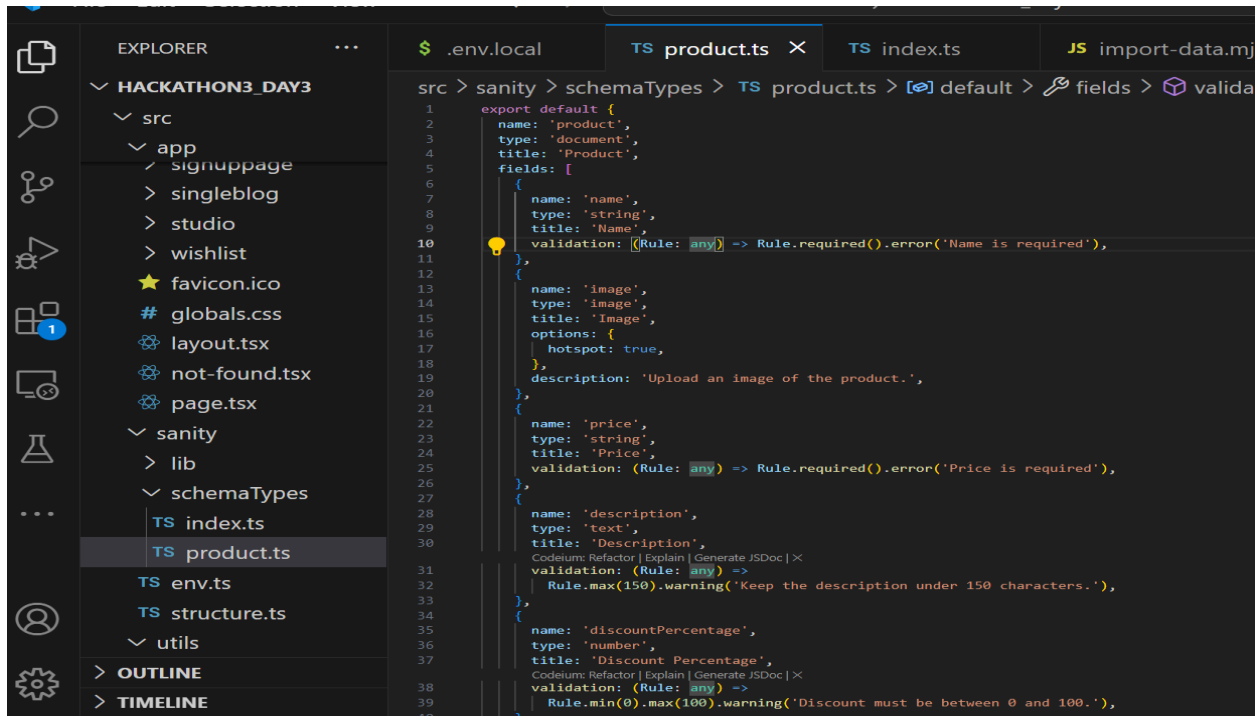
Changes in package.json



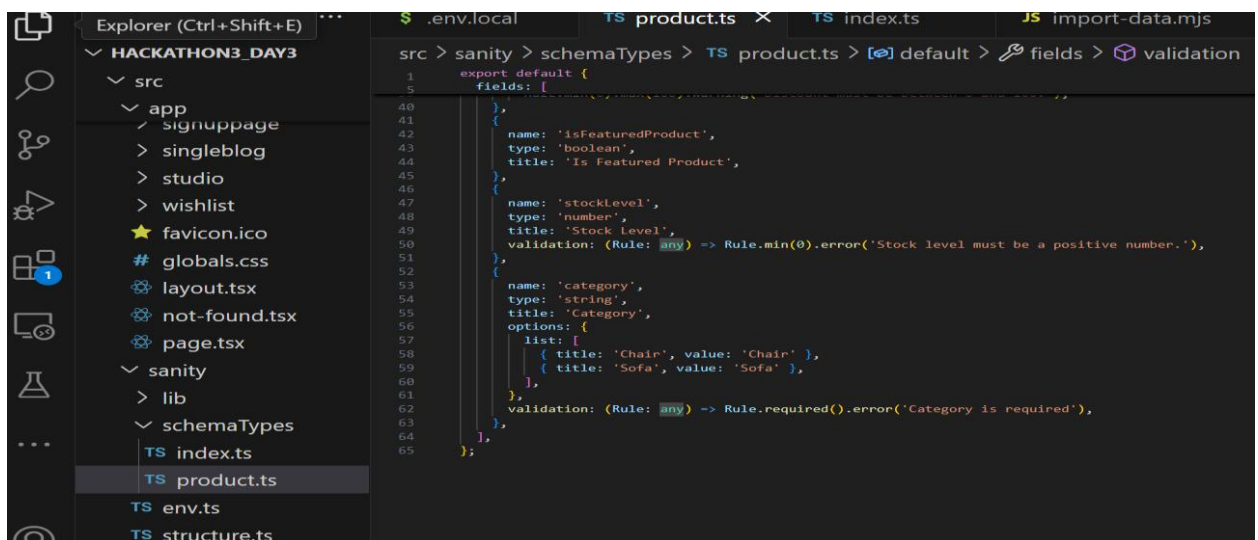
```
1 {
2   "name": "figma-project-practice-hassanjhr",
3   "version": "0.1.0",
4   "private": true,
5   "type": "module",
6   "scripts": {
7     "dev": "next dev --turbo",
8     "build": "next build",
9     "start": "next start",
10    "lint": "next lint",
11    "import-data": "node ./scripts/import-data.mjs"
12  },
13  "dependencies": {
14    "@sanity/client": "^6.25.0",
15    "@sanity/image-url": "^1.1.0",
16    "@sanity/vision": "^3.70.0",
17    "axios": "^1.7.9",
18    "dotenv": "^16.4.7",
19    "next": "^13.1.0",
20    "next-sanity": "^9.8.39",
21    "react": "^18.0.0",
22    "react-dom": "^18.0.0",
23    "react-icons": "^5.4.0",
24    "sanity": "^3.70.0",
25    "styled-components": "^6.1.14"
26  },
27  "devDependencies": {
28    "@types/node": "^20",
29    "@types/react": "^18",
30    "@types/react-dom": "^18",
31    "eslint": "^8",
32    "eslint-config-next": "15.0.4",
33    "postcss": "^8",
34    "tailwindcss": "^3.4.1",
35    "typescript": "^5"
36  }
37 }
38
```

Product Schemas

- Made necessary modifications to the default schemas to align with the requirements of the migrated data.
- Updated the schema structure to ensure seamless compatibility with the API data format.

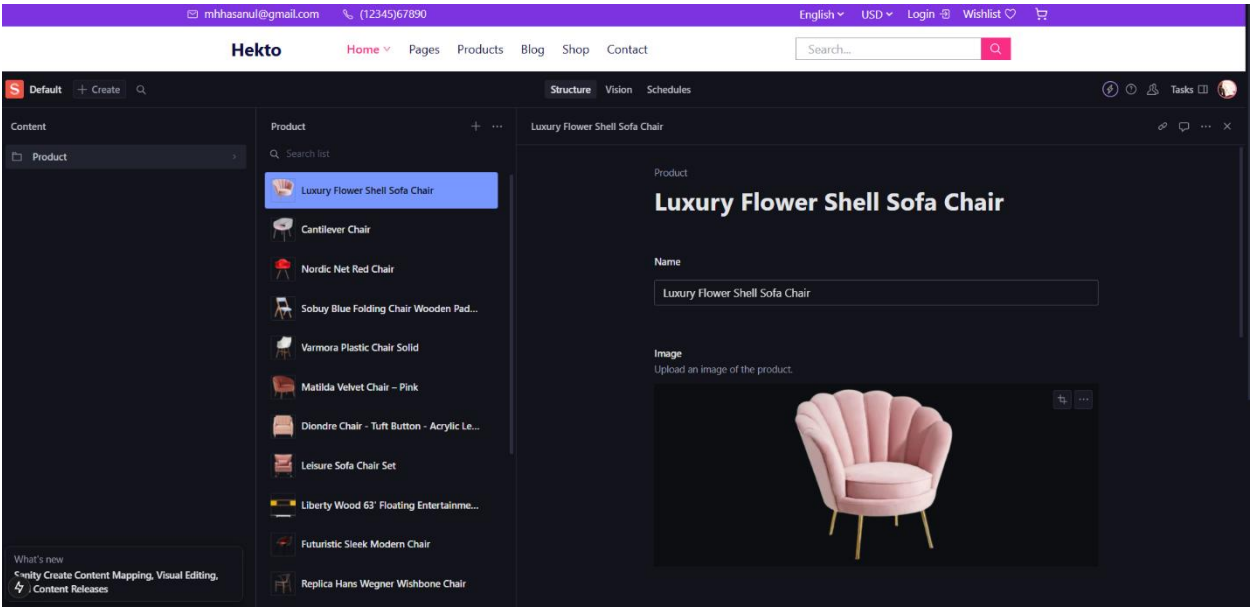
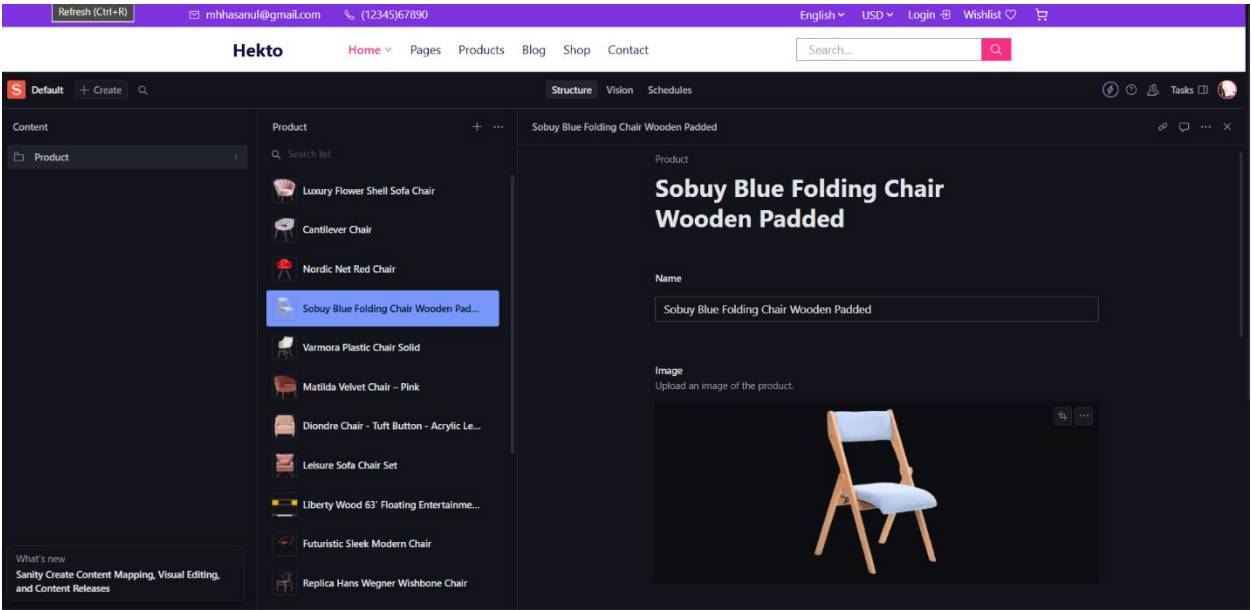


```
1 export default {
2   name: 'product',
3   type: 'document',
4   title: 'Product',
5   fields: [
6     {
7       name: 'name',
8       type: 'string',
9       title: 'Name',
10      validation: (Rule: any) => Rule.required().error('Name is required'),
11    },
12    {
13      name: 'image',
14      type: 'image',
15      title: 'Image',
16      options: {
17        hotspot: true,
18      },
19      description: 'Upload an image of the product.',
20    },
21    {
22      name: 'price',
23      type: 'string',
24      title: 'Price',
25      validation: (Rule: any) => Rule.required().error('Price is required'),
26    },
27    {
28      name: 'description',
29      type: 'text',
30      title: 'Description',
31      validation: (Rule: any) =>
32        Rule.max(150).warning('Keep the description under 150 characters.'),
33    },
34    {
35      name: 'discountPercentage',
36      type: 'number',
37      title: 'Discount Percentage',
38      validation: (Rule: any) =>
39        Rule.min(0).max(100).warning('Discount must be between 0 and 100.'),
40    }
41  ]
42 }
```



```
1 export default {
2   fields: [
3     {
4       name: 'isFeaturedProduct',
5       type: 'boolean',
6       title: 'Is Featured Product',
7     },
8     {
9       name: 'stockLevel',
10      type: 'number',
11      title: 'Stock level',
12      validation: (Rule: any) => Rule.min(0).error('Stock level must be a positive number.'),
13    },
14    {
15      name: 'category',
16      type: 'string',
17      title: 'Category',
18      options: {
19        list: [
20          { title: 'Chair', value: 'Chair' },
21          { title: 'Sofa', value: 'Sofa' },
22        ],
23      },
24      validation: (Rule: any) => Rule.required().error('Category is required'),
25    }
26  ],
27 }
```

Migrated Data in Sanity:



GROQ Query Implementation:

The screenshot displays the Hekto application interface, which is a tool for managing and querying data. The top navigation bar includes the Hekto logo, a search bar, and links for Home, Pages, Products, Blog, Shop, and Contact. The main interface is divided into several sections:

- Dataset:** A dropdown menu showing 'production'.
- API Version:** A dropdown menu showing 'Other'.
- Custom API Version:** A dropdown menu showing 'v2025-01-20'.
- Perspective:** A dropdown menu showing 'raw'.
- Query URL:** A text field containing the URL: `https://0ouSayap.api.sanity.io/v2025-01-20/data/query/production?query=%5Btype%3Dproduct%5D`.

The main content area is split into two panels:

- Query:** A text editor showing the GROQ query: `*[type==product]`.
- Result:** A JSON output showing the results of the query. The first item is a product with the following properties: `{ "isNew": true, "discountPercentage": 10, "tags": ["item"], "price": 781, "imageUrl": { "_type": "image", "asset": { "_ref": "image-e20620a261d3a68c851f873301bbf6baf94e855-223x229-png", "_type": "reference", "_rev": "k16uv2mlcNemf7PCABSVTD", "_type": "product", "description": "A luxurious armchair with velvet fabric and golden metal legs.", "_createdAt": "2025-01-20T20:41:46Z", "_id": "576ef73c-a3fb-4ade-916a-0f85078edcfd", "title": "Stylish Golden Metal Legs Mint Blue Fabric Velvet Sofa Leisure Armchair", "_updatedAt": "2025-01-20T20:52:26Z" }, "price": 850, "imageUrl": { "_type": "image", "asset": { "_ref": "image-a777fa2eccfb4d5c3e931f7297cb4abdbdb42310-247x244-png", "_type": "reference", "_rev": "703b333e-200f-476b-800f-6d9-01c044e6" } } }`.

At the bottom left, there is a 'What's new' section with the text: 'Sanity Create Content Mapping, Visual Editing, and Content Releases'.