Hackathon Day 3 Task Documentation

Introduction

On Day 3 of the hackathon, the focus was on fetching data from an external API, importing it into Sanity CMS, and displaying the products on the frontend using Next.js. The products were categorized into **latest products** and **featured products** for better presentation. The template provided was "Template 4," which included furniture products like chairs and sofas.

This document outlines the step-by-step process followed to complete the task.

Steps

1. Fetching Data from the External API

The first step was to fetch product data from the external API:
`https://next-ecommerce-template-4.vercel.app/api/product`

Code to Fetch Data:

```
import axios from 'axios';

Tabnine | Edit | Test | Explain | Document | Pieces: Comment | Pieces: Explain
async function fetchProductData() {
  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;

    console.log('Data Fetched Successfully:', products);
    return products;
  } catch (error) {
    console.error('Error Fetching Data:', error);
  }
}

fetchProductData();
```

Result: This fetched an array of product objects, each containing details like `id`, `name`, `imagePath`, `price`, `description`, `discountPercentage`, `isFeaturedProduct`, `stockLevel`, and `category`.

2. Creating Sanity Schema

The next step was to define a schema in Sanity for storing the product data.

Product Schema in Sanity:

import-data.mjs M | chair.json M | product.ts 6, M

src > sanity > schemaTypes > product.ts > product > fields > title

```python
export const product = {
  name: "product",
  type: "document",
  title: "Product",
  fields: [
    {
      name: "name",
      type: "string",
      title: "Name",
      Pieces: Comment | Pieces: Explain
      validation: (Rule: any) => Rule.required().error("Name is required"),
    },
    {
      name: "image",
      type: "image",
      title: "Image",
      options: {
        hotspot: true,
      },
      description: "Upload an image of the product.",
    },
    {
      name: "price",
      type: "string",
      title: "Price",
      Pieces: Comment | Pieces: Explain
      validation: (Rule: any) => Rule.required().error("Price is required"),
    },
    {
      name: "description",
      type: "text",
      title: "Description",
      Pieces: Comment | Pieces: Explain
      validation: (Rule: any) =>
        Rule.max(150).warning("Keep the description under 150 characters."),
```

import-data.mjs M | chair.json M | product.ts 6, M

src > sanity > schemaTypes > product.ts > product > fields > title

```
1   export const product = {
5     fields: [
        Pieces: Comment | Pieces: Explain
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",
        Pieces: Comment | Pieces: Explain
38        validation: (Rule: any) =>
39          Rule.min(0).max(100).warning("Discount must be between 0 and 100."),
40      },
41      {
42        name: "isFeaturedProduct",
43        type: "boolean",
44        title: "Is FeatuDred Product",
45      },
46      {
47        name: "stockLevel",
48        type: "number",
49        title: "Stock Level",
        Pieces: Comment | Pieces: Explain
50        validation: (Rule: any) =>
51          Rule.min(0).error("Stock level must be a positive number."),
52      },
53      {
54        name: "category",
55        type: "string",
56        title: "Category",
57        options: {
58          list: [
59            { title: "Chair", value: "Chair" },
60            { title: "Sofa", value: "Sofa" },
```

```
import-data.mjs M    chair.json M    product.ts 6, M ×
src > sanity > schemaTypes > product.ts > product > fields > title
1   export const product = {
5     fields: [
56        title: "Category",
57        options: {
58          list: [
59            { title: "Chair", value: "Chair" },
60            { title: "Sofa", value: "Sofa" },
61          ],
62        },
          Pieces: Comment | Pieces: Explain
63        validation: (Rule: any) => Rule.required().error("Category is required"),
64      },
65    ],
66  };
67
```

Purpose: The schema ensured proper structuring and validation of the data being imported.

---

3. Importing Data into Sanity

Using Sanity's client and API token, the fetched data was imported into Sanity after some processing (e.g., uploading images).

Code for Importing Data:

import-data.mjs M × | chair.json M | product.ts 6, M

src > scripts > import-data.mjs > ...

```
98
99  import { createClient } from '@sanity/client';
100 import axios from 'axios';
101 import dotenv from 'dotenv';
102 import { fileURLToPath } from 'url';
103 import path from 'path';
104
105 const __filename = fileURLToPath(import.meta.url);
106 const __dirname = path.dirname(__filename);
107 dotenv.config({ path: path.resolve(__dirname, '../../.env') });
108
109 const client = createClient({
110   projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
111   dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
112   token: process.env.SANITY_API_TOKEN,
113   apiVersion: '2025-01-15',
114   useCdn: false,
115 });
116
    Tabnine | Edit | Test | Explain | Document | Pieces: Comment | Pieces: Explain
117 async function uploadImageToSanity(imageUrl) {
118   try {
119     console.log(`Uploading Image : ${imageUrl}`);
120     const response = await axios.get(imageUrl, { responseType: 'arraybuffer' });
121     const buffer = Buffer.from(response.data);
122     const asset = await client.assets.upload('image', buffer, {
123       filename: imageUrl.split('/').pop(),
124     });
125     console.log(`Image Uploaded Successfully : ${asset._id}`);
126     return asset._id;
127   } catch (error) {
128     console.error('Failed to Upload Image:', imageUrl, error);
129     return null;
130   }
131 }
```

File Edit Selection View ··· import-next-ecommerce-template-4

import-data.mjs M | chair.json M | product.ts 6, M

src > scripts > import-data.mjs > ...

```
117 async function uploadImageToSanity(imageUrl) {
131 }
132
    Tabnine | Edit | Test | Explain | Document | Pieces: Comment | Pieces: Explain
133 async function importData() {
134   try {
135     console.log('Fetching Product Data From API ...');
136
137     const response = await axios.get("https://next-ecommerce-template-4.vercel.app/api/product");
138     const products = response.data.products;
139
140     for (const item of products) {
141       console.log(`Processing Item: ${item.name}`);
142
143       let imageRef = null;
144       if (item.imagePath) {
145         imageRef = await uploadImageToSanity(item.imagePath);
146       }
147
148       const sanityItem = {
149         _type: 'product',
150         name: item.name,
151         category: item.category || null,
152         price: item.price,
153         description: item.description || '',
154         discountPercentage: item.discountPercentage || 0,
155         stockLevel: item.stockLevel || 0,
156         isFeaturedProduct: item.isFeaturedProduct,
157         image: imageRef
158           ? {
159               _type: 'image',
160               asset: {
161                 _type: 'reference',
162                 _ref: imageRef,
```

```
import-data.mjs M ×   chair.json M   product.ts 6, M
src > scripts > import-data.mjs > ...
133  async function importData() {
148        const sanityItem = {
149          _type: 'product',
150          name: item.name,
151          category: item.category || null,
152          price: item.price,
153          description: item.description || '',
154          discountPercentage: item.discountPercentage || 0,
155          stockLevel: item.stockLevel || 0,
156          isFeaturedProduct: item.isFeaturedProduct,
157          image: imageRef
158            ? {
159                _type: 'image',
160                asset: {
161                  _type: 'reference',
162                  _ref: imageRef,
163                },
164              }
165            : undefined,
166        };
167
168        console.log(`Uploading ${sanityItem.category} - ${sanityItem.name} to Sanity!`);
169        const result = await client.create(sanityItem);
170        console.log(`Uploaded Successfully: ${result._id}`);
171      }
172
173      console.log('Data Import Completed Successfully!');
174    } catch (error) {
175      console.error('Error Importing Data : ', error);
176    }
177  }
178
179  importData();
180
```

---

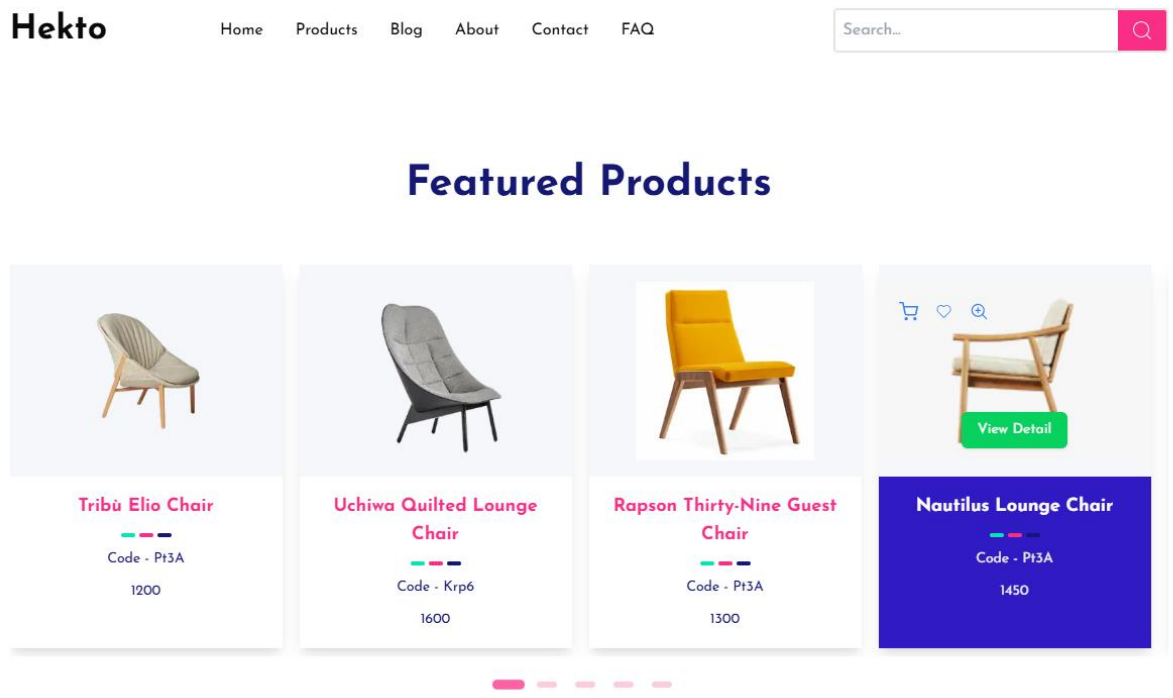
4. Displaying Data on the Frontend (Next.js)

On the frontend, the data was displayed in a simple and organized layout using Next.js. The **App Router** was used for routing.

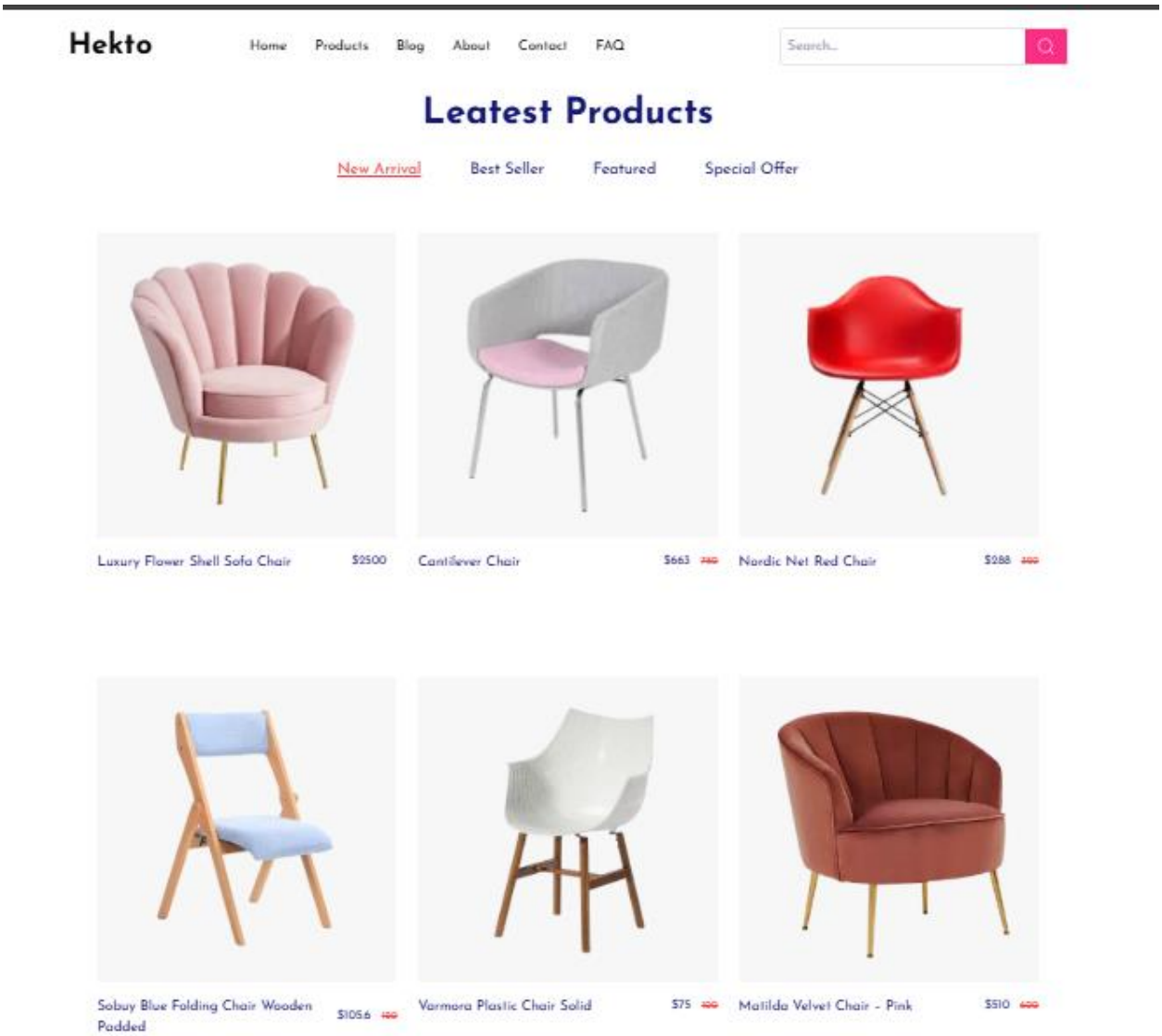
Steps:

1. **Fetch Data**: Use `client.fetch` to fetch the data from Sanity.
2. **Organize Products**: Separate them into **latest products** and **featured products**.
3. **Render the Products**: Display the products dynamically on the page.

Featured Products Image:



Latest Products Image:



Conclusion

The Day 3 task successfully demonstrated fetching data from an external API, structuring and importing it into Sanity, and dynamically displaying it on a Next.js frontend. This process involved integrating multiple tools and technologies, ensuring smooth data flow from source to display.