

Day 3 - API Integration Report - Sinp Ecommerce

1. API Selection and Setup:

- The external API used for integration was <https://hackathon-apis.vercel.app/api/products>.
- Sanity Client was employed to handle data operations, ensuring smooth integration between the external API and the Sanity backend.

2. Data Fetching:

- The `importData` function made use of the `fetch` method in conjunction with Sanity Client to retrieve product data from the API.
- Fetched data was processed to align with the required structure for Sanity documents.

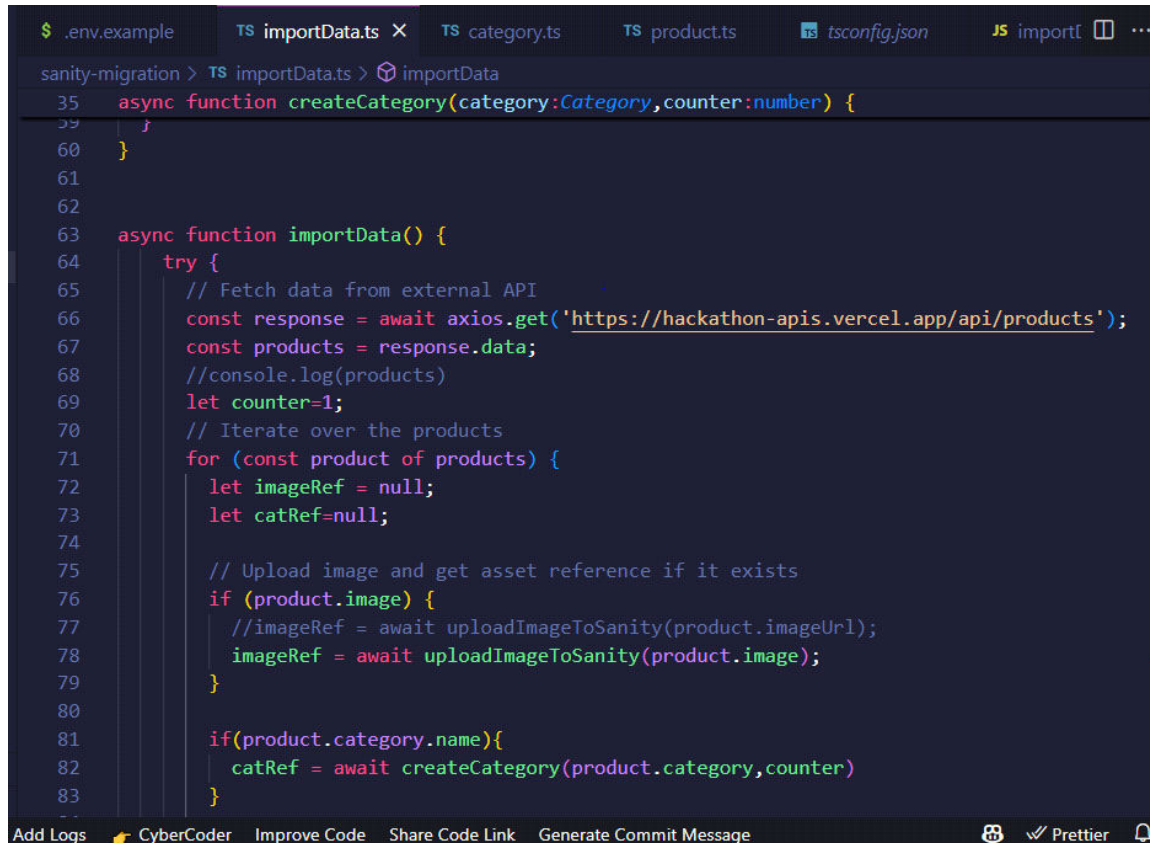
3. Data Mapping:

- Instead of a traditional `for` loop, `map` was used for processing each product.
- Optional fields like Dimensions were conditionally handled within the `map` callback.

4. Sanity Integration:

- Product images were uploaded to Sanity's Media Library using `client.assets.upload`.
- Each product was uploaded using the `client.createOrReplace` method to ensure unique entries.
- Categories were created using `client.createIfNotExists` to avoid duplications.

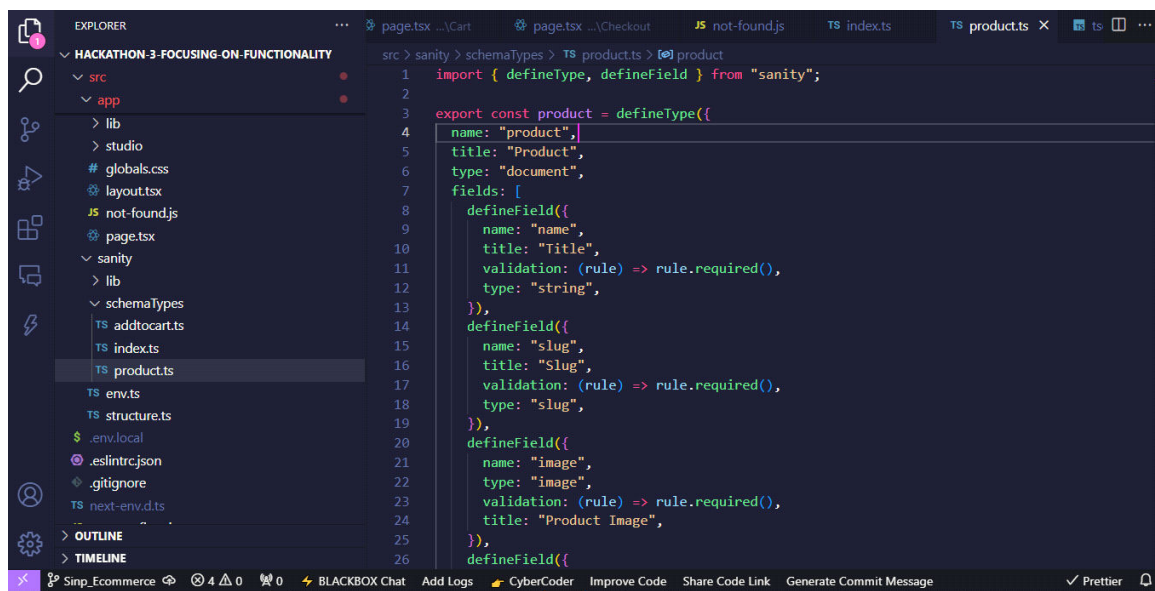
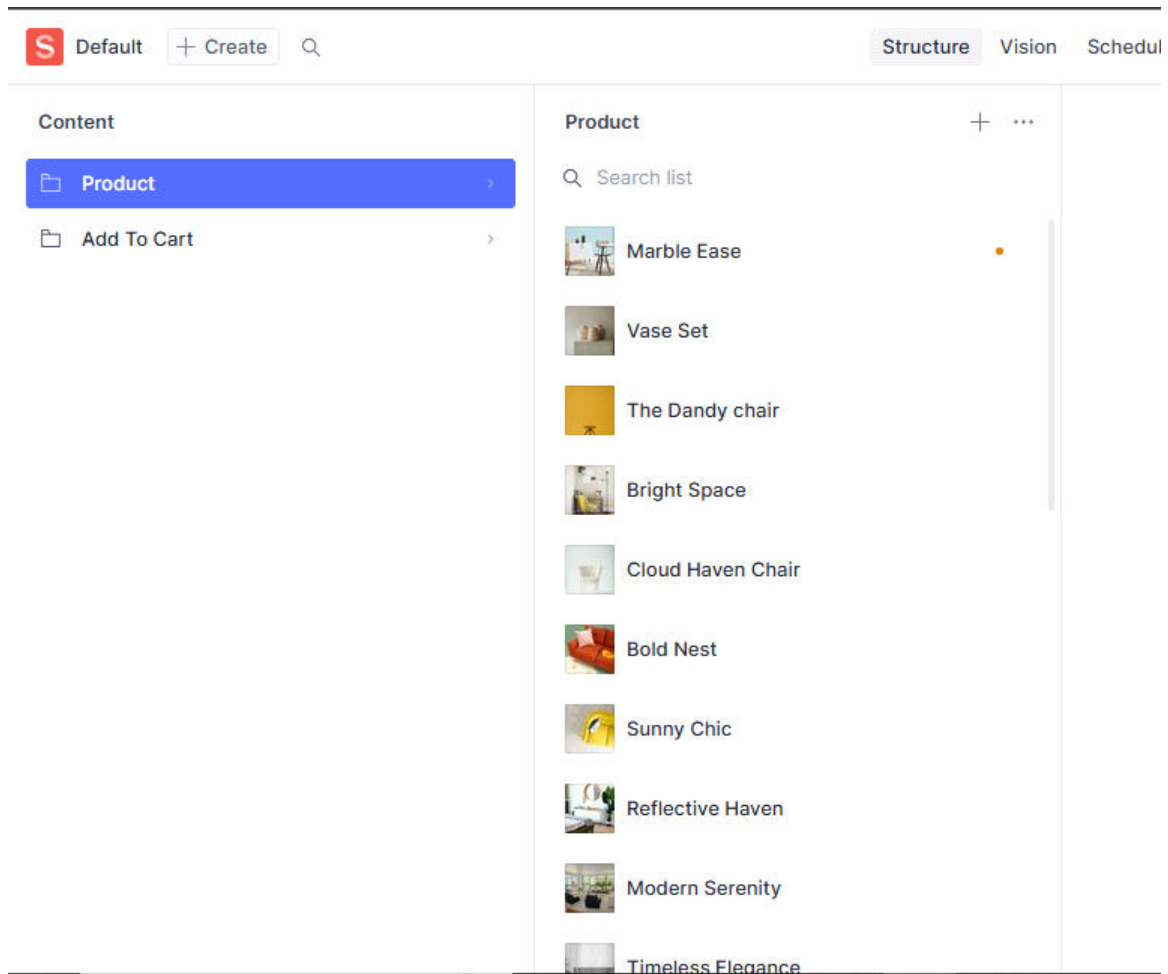
1. Screenshots of:



The screenshot shows a code editor with a dark theme. The top bar displays several tabs: `$.env.example`, `TS importData.ts` (active), `TS category.ts`, `TS product.ts`, `tsconfig.json`, and `JS import`. The main editor area shows the following TypeScript code:

```
sanity-migration > TS importData.ts > importData
35  async function createCategory(category:Category,counter:number) {
36    // ...
37  }
60  }
61
62
63  async function importData() {
64    try {
65      // Fetch data from external API
66      const response = await axios.get('https://hackathon-apis.vercel.app/api/products');
67      const products = response.data;
68      //console.log(products)
69      let counter=1;
70      // Iterate over the products
71      for (const product of products) {
72        let imageRef = null;
73        let catRef=null;
74
75        // Upload image and get asset reference if it exists
76        if (product.image) {
77          //imageRef = await uploadImageToSanity(product.imageUrl);
78          imageRef = await uploadImageToSanity(product.image);
79        }
80
81        if(product.category.name){
82          catRef = await createCategory(product.category,counter)
83        }
84      }
85    } catch (error) {
86      console.error('Error importing data:', error);
87    }
88  }
89  }
```

The bottom status bar includes the text "Add Logs" and several icons: a yellow lightning bolt (CyberCoder), "Improve Code", "Share Code Link", "Generate Commit Message", a bug icon, a checkmark (Prettier), and a bell icon.



```
File Edit Selection View Go ... hackathon-3-focusing-on-functionality TS product.ts x ts ...  
EXPLORER  
HACKATHON 3-FOCUSING-ON-FUNCTIONALITY  
src  
  app  
    lib  
    studio  
    # globals.css  
    layout.tsx  
    JS not-found.js  
    page.tsx  
  sanity  
    lib  
    schemaTypes  
      addtocart.ts  
      index.ts  
      product.ts  
      env.ts  
      structure.ts  
    .env.local  
    .eslintrc.json  
    .gitignore  
    next-env.d.ts  
  OUTLINE  
  TIMELINE  
src > sanity > schemaTypes > TS product.ts > [0] product  
3 export const product = defineType({  
7   fields: [  
26     },  
27     defineField({  
28       name: "price",  
29       type: "number",  
30       validation: (rule) => rule.required(),  
31       title: "Price",  
32     })),  
33     defineField({  
34       name: "quantity",  
35       title: "Quantity",  
36       type: "number",  
37       validation: (rule) => rule.min(0),  
38     })),  
39     defineField({  
40       name: "tags",  
41       type: "array",  
42       title: "Tags",  
43       of: [  
44         {  
45           type: "string",  
46         }],  
47     })),  
48     defineField({
```

```
File Edit Selection View Go ... hackathon-3-focusing-on-functionality TS product.ts x ts ...  
EXPLORER  
HACKATHON 3-FOCUSING-ON-FUNCTIONALITY  
src  
  app  
    lib  
    studio  
    # globals.css  
    layout.tsx  
    JS not-found.js  
    page.tsx  
  sanity  
    lib  
    schemaTypes  
      addtocart.ts  
      index.ts  
      product.ts  
      env.ts  
      structure.ts  
    .env.local  
    .eslintrc.json  
    .gitignore  
    next-env.d.ts  
  OUTLINE  
  TIMELINE  
src > sanity > schemaTypes > TS product.ts > [0] product  
7   fields: [  
47     },  
48     defineField({  
49       name: "description",  
50       title: "Description",  
51       type: "text",  
52       description: "Detailed description of the product",  
53     })),  
54     defineField({  
55       name: "features",  
56       title: "Features",  
57       type: "array",  
58       of: [{ type: "string" }],  
59       description: "List of key features of the product",  
60     })),  
61     defineField({  
62       name: "dimensions",  
63       title: "Dimensions",  
64       type: "object",  
65       fields: [  
66         { name: "height", title: "Height", type: "string" },  
67         { name: "width", title: "Width", type: "string" },  
68         { name: "depth", title: "Depth", type: "string" },  
69       ],  
70       description: "Dimensions of the product",  
71     })),  
72   ],  
73 }));
```

The screenshot shows the VS Code editor with the file explorer on the left displaying the project structure. The main editor area shows the code for the `ProductSlider` component in `page.tsx`. The code includes state management for `productList` and an interval for fetching products.

```
39
40 const ProductSlider = () => {
41
42
43   const prevRef = useRef<HTMLDivElement>(null);
44   const nextRef = useRef<HTMLDivElement>(null);
45
46   const [productList, setproductList] = useState<Product[]>([]);
47
48   useEffect(() => {
49     const intervalId = setInterval(async () => {
50       try {
51         const data = await fetchAllProducts();
52         if (data) {
53           setproductList(data);
54         }
55       } catch (error) {
56         console.error("Error fetching hero card data:", error);
57       }
58     }, 2000);
59
60     return () => clearInterval(intervalId);
61   }, []);
62
63   const handleAddToCart = async (...
64
65   );
66 }
```

The screenshot shows the VS Code editor with the file explorer on the left. The main editor area shows the code for the `fetchAllProducts` function in `ts apits`. The function uses GraphQL to fetch product data from a database.

```
8
9
10 export const fetchAllProducts = async (): Promise<any[]> => {
11   const query = `
12     query {
13       products {
14         _id,
15         _type,
16         _createdAt,
17         _updatedAt,
18         name,
19         slug {
20           current
21         },
22         description,
23         price,
24         quantity,
25         features,
26         dimensions {
27           width,
28           height,
29           depth,
30           _type
31         },
32         image {
33           asset->{
34             _id,
35             url
36           }
37         }
38       }
39     }
40   `;
41   try {
42     const products = await client.fetch(query);
43     return products;
44   } catch (error) {
45     console.error("Error fetching product data:", error);
46     throw new Error("Failed to fetch product data");
47   }
48 }
```

