

**IQRA EHSAN 00454450**

Document Title: "Day 3 - API Integration Report--General ECommerce -  
[Frurniture Website]"

### **1.API integration process**

I carefully read the provided API documenttation to understand the available endpoint(/products)

First installed sanity in project and then used manual API-Document to update website.

### **2:Adjustments made to schemas**

I defined a clear folder structure in wesite to make API integration in our website.

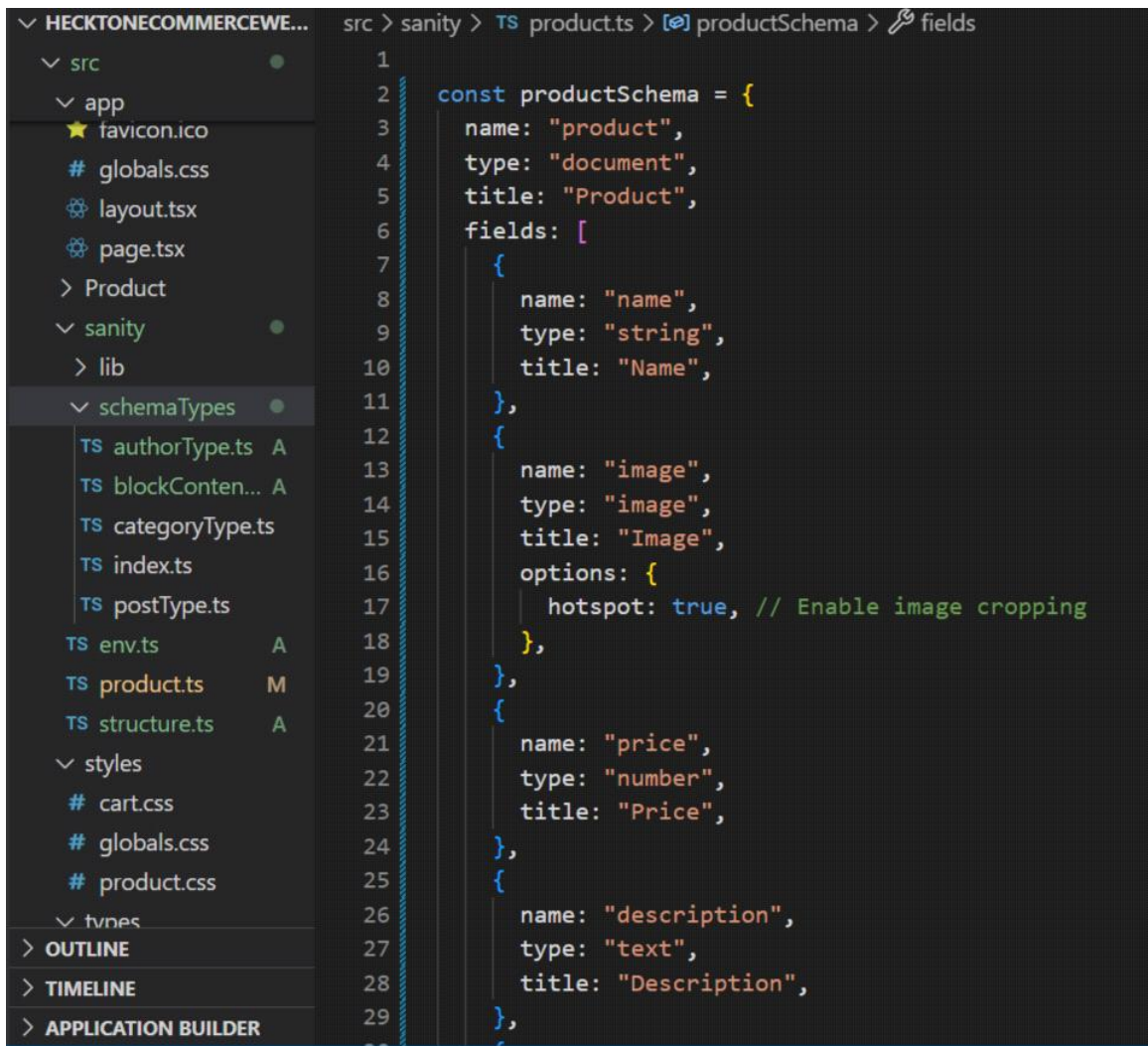
### **3.Migration steps and tools used.**

As it was template zero so i understand code deeply and created importSanityData.mjs file and here used ecommerce-business API-Integration setup such as API token and Project Id.

By using VS code editor and documentaion performed further task.

Screenshots of:   ▪ API calls.

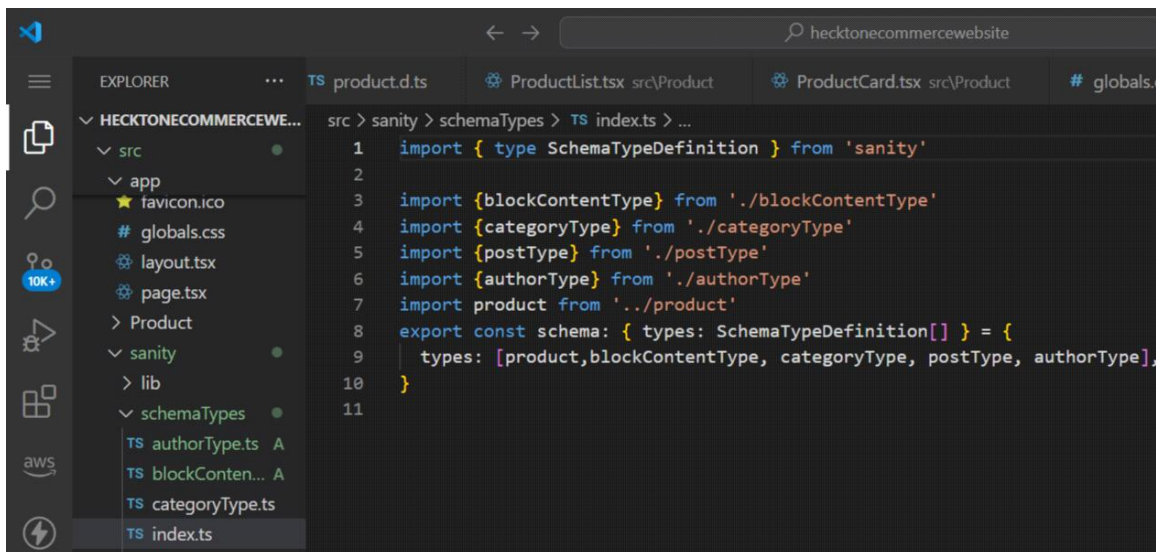
Created product.tsx file. and define fields in project schema.



The screenshot shows a VS Code editor with the file explorer on the left and the editor window on the right. The file explorer shows the project structure for 'HECKTONECOMMERCEWE...', with the 'sanity' folder expanded to show 'schemaTypes'. The 'product.ts' file is selected. The editor window shows the 'productSchema' definition in 'product.ts'.

```
1
2
3  const productSchema = {
4    name: "product",
5    type: "document",
6    title: "Product",
7    fields: [
8      {
9        name: "name",
10       type: "string",
11       title: "Name",
12     },
13     {
14       name: "image",
15       type: "image",
16       title: "Image",
17       options: {
18         hotspot: true, // Enable image cropping
19       },
20     },
21     {
22       name: "price",
23       type: "number",
24       title: "Price",
25     },
26     {
27       name: "description",
28       type: "text",
29       title: "Description",
30     },
31   ],
32 }
```

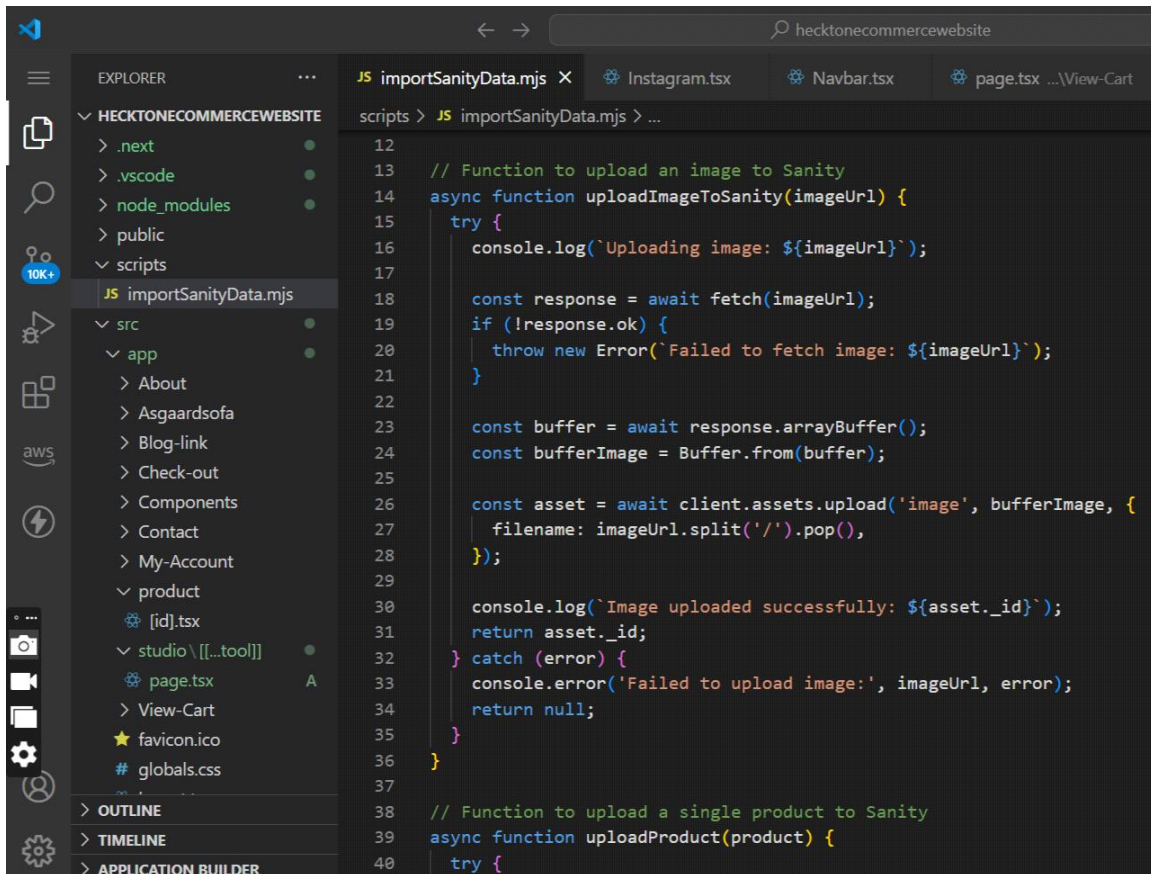
imported a product.ts file in index.ts



The screenshot shows a VS Code editor with the file explorer on the left and the editor window on the right. The file explorer shows the project structure for 'HECKTONECOMMERCEWE...', with the 'sanity' folder expanded to show 'schemaTypes'. The 'index.ts' file is selected. The editor window shows the 'index.ts' file content.

```
1  import { type SchemaTypeDefinition } from 'sanity'
2
3  import {blockContentType} from './blockContentType'
4  import {categoryType} from './categoryType'
5  import {postType} from './postType'
6  import {authorType} from './authorType'
7  import product from './product'
8  export const schema: { types: SchemaTypeDefinition[] } = {
9    types: [product, blockContentType, categoryType, postType, authorType],
10  }
11
```

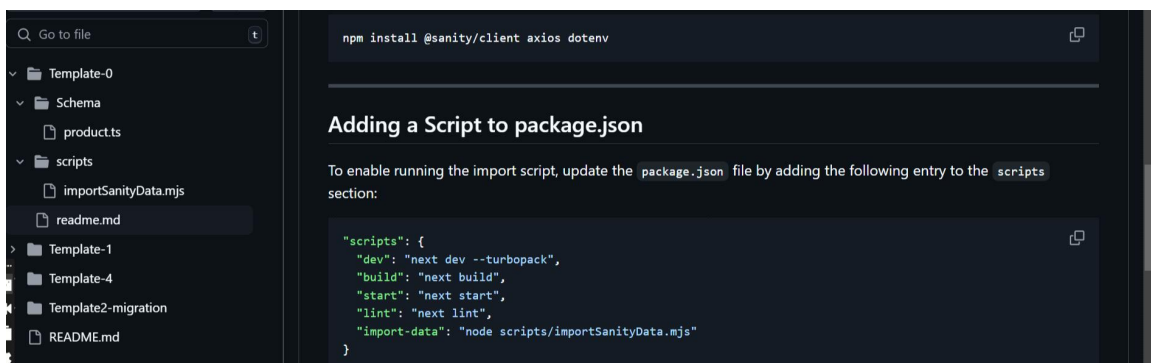
Created a importSanityData.mj file in scripts folder.



The screenshot shows the Visual Studio Code editor with the file explorer on the left. The project is named 'HECKTONECOMMERCEWEBSITE'. The 'scripts' folder is expanded, showing 'importSanityData.mjs'. The main editor displays the content of this file, which includes two asynchronous functions: 'uploadImageToSanity' and 'uploadProduct'. The 'uploadImageToSanity' function uses 'fetch' to get an image, converts it to a buffer, and uploads it to Sanity. The 'uploadProduct' function is partially visible at the bottom.

```
12
13 // Function to upload an image to Sanity
14 async function uploadImageToSanity(imageUrl) {
15   try {
16     console.log(`Uploading image: ${imageUrl}`);
17
18     const response = await fetch(imageUrl);
19     if (!response.ok) {
20       throw new Error(`Failed to fetch image: ${imageUrl}`);
21     }
22
23     const buffer = await response.arrayBuffer();
24     const bufferImage = Buffer.from(buffer);
25
26     const asset = await client.assets.upload('image', bufferImage, {
27       filename: imageUrl.split('/').pop(),
28     });
29
30     console.log(`Image uploaded successfully: ${asset._id}`);
31     return asset._id;
32   } catch (error) {
33     console.error('Failed to upload image:', imageUrl, error);
34     return null;
35   }
36 }
37
38 // Function to upload a single product to Sanity
39 async function uploadProduct(product) {
40   try {
```

Added import-data in pake.json



The screenshot shows the Visual Studio Code editor with the file explorer on the left. The 'scripts' folder is expanded, showing 'importSanityData.mjs'. The main editor displays the content of the 'package.json' file, specifically the 'scripts' section. The 'import-data' script has been added to the list of scripts.

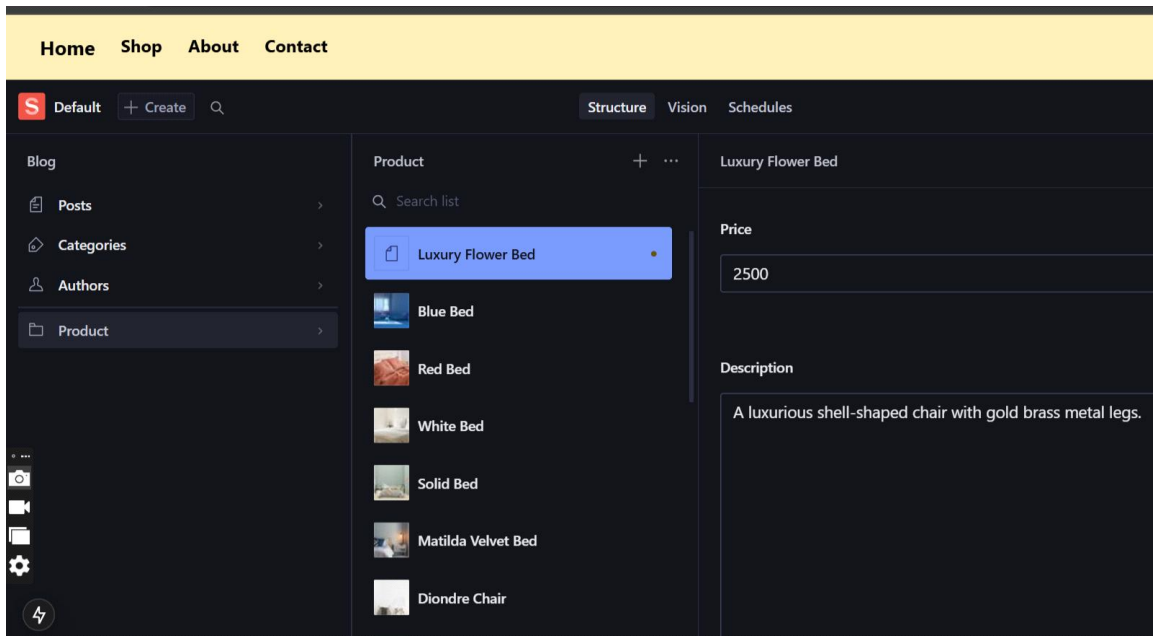
```
npm install @sanity/client axios dotenv

Adding a Script to package.json

To enable running the import script, update the package.json file by adding the following entry to the scripts section:

{
  "scripts": {
    "dev": "next dev --turbo",
    "build": "next build",
    "start": "next start",
    "lint": "next lint",
    "import-data": "node scripts/importSanityData.mjs"
  }
}
```

■ Data successfully displayed in the frontend.



■ **Populated Sanity CMS fields.**

name,price,category,description

**Day 3 Checklist: Self-Validation Checklist:**

API Understanding: 🕒 ✓

Schema Validation: 🕒 ✓

Data Migration: 🕒 ✓

API Integration in Next.js: 🕒 ✓

Submission Preparation: 🕒 ✓