

- **NAME : HAFSA SULTAN**
 - **ROLL NUM : 453493**
-

API Integration Report **(Stylestride)**

API Integration Process :

Overview :

This reports outlines the activities completed on Day 3 of the hackathon The main focus was on migrating to the market place using Sanity CMS ..The detailed are given below

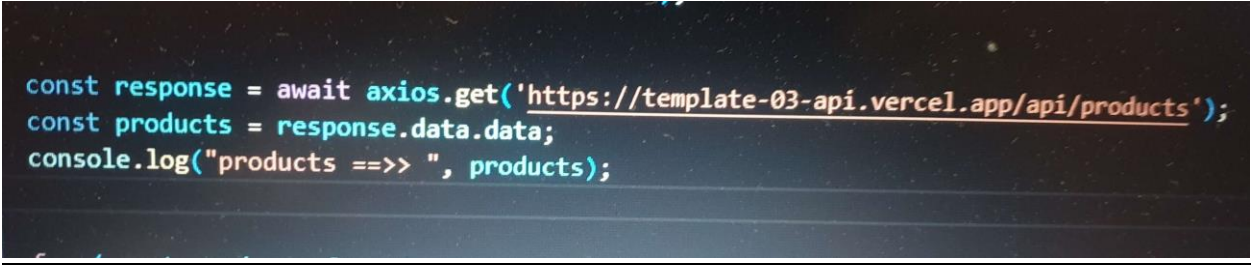
STEP # 1 :- API Endpoint Setup :

API Endpoint : 'Https://template-03-api.vercel.app/api/products'

Data Structure: The API provides produce data, including field like product Name 'price', "inventory", " fields category "colors" 'status', 'description", "image".

Library Used: Axios was used - Library used to fetch data from the API.

STEP # 2 :-Fetch Data From API :



```
const response = await axios.get('https://template-03-api.vercel.app/api/products');
const products = response.data.data;
console.log("products ==>> ", products);
```

The API call retrieves an array of product objects from the endpoint, ensuring proper serialization.

PRODUCT SCHEMA MODIFICATION :

1 Schema pics". “

```
.env.local  index.ts  page.tsx  importSanityData.mjs
src > sanity > schemaTypes > product.ts > productSchema
1  export const productSchema = {
2    name: 'product',
3    title: 'Product',
4    type: 'document',
5    fields: [
6      {
7        name: 'productName',
8        title: 'Product Name',
9        type: 'string',
10     },
11     {
12       name: 'category',
13       title: 'Category',
14       type: 'string',
15     },
16     {
17       name: 'price',
18       title: 'Price',
19       type: 'number',
20     },
21     {
22       name: 'inventory',
23       title: 'Inventory',
24       type: 'number',
25     },
26     {
27       name: 'colors',
```

```
env.local  index.ts  page.tsx  importSanityData.mjs  product.ts  fetch.ts  qu
sanity > schemaTypes > product.ts > productSchema
export const productSchema = {
  fields: [
    {
      name: 'colors',
      title: 'Colors',
      type: 'array',
      of: [{ type: 'string' }],
    },
    {
      name: 'status',
      title: 'Status',
      type: 'string',
    },
    {
      name: 'image',
      title: 'Image',
      type: 'image', // Using Sanity's image type for image field
      options: {
        hotspot: true,
      },
    },
    {
      name: 'description',
      title: 'Description',
      type: 'text',
    },
  ],
}
```

2 . “Fieldupdates”

Colors: Added as optional tancopadate array of strings to multiple color options.

Image : Configured as a sanity image field with hotspot enabled for better image cropping & scaling .

STEPS TAKEN :

1.Environmental Setup :

- utilized dotenv to load environment variables from env.local.
- Key environment variables include.
- NEXT_PUBLIC_SANITY_PROJECT_ID.
- NEXT_PUBLIC_SANITY_DATASET.
- SANITY_TOKEN.

2. Data Fetching:

- Retrieved product data from the API endpoint using Axios.
- Parsed and logged the data to confirm its structure and integrity.

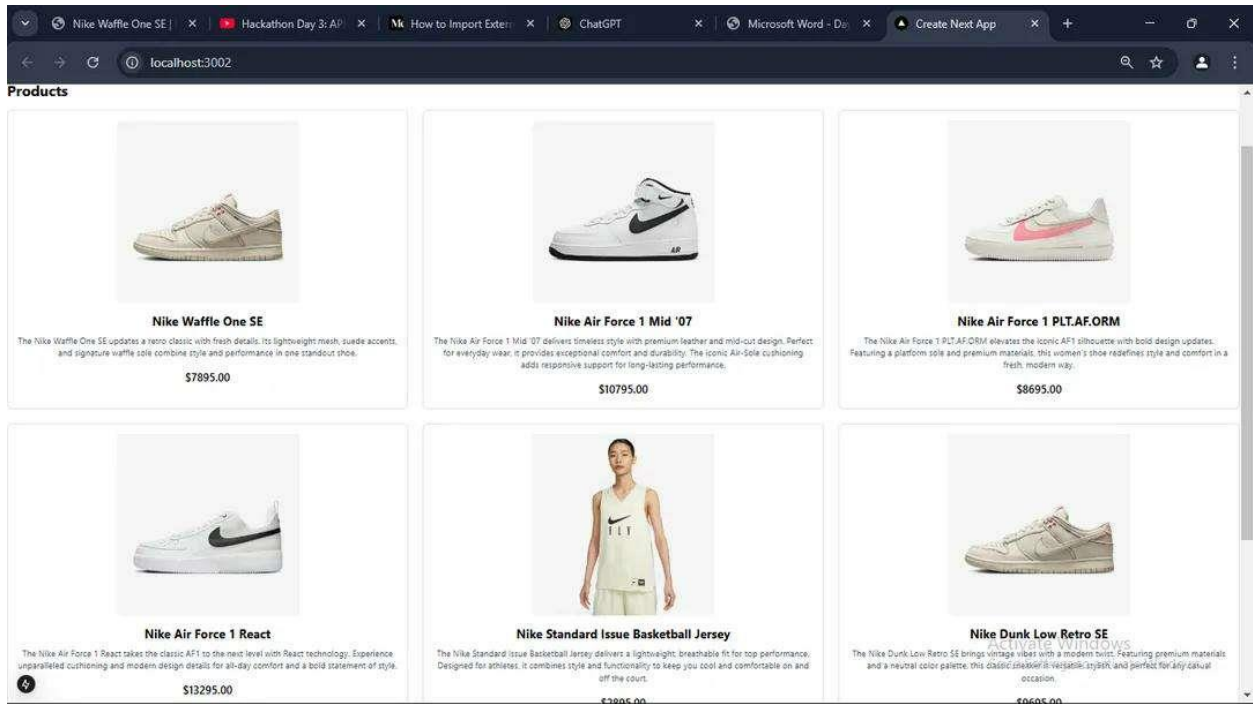
3. Document Creation:

- Created Sanity documents for each product by combining API data and uploaded image references.

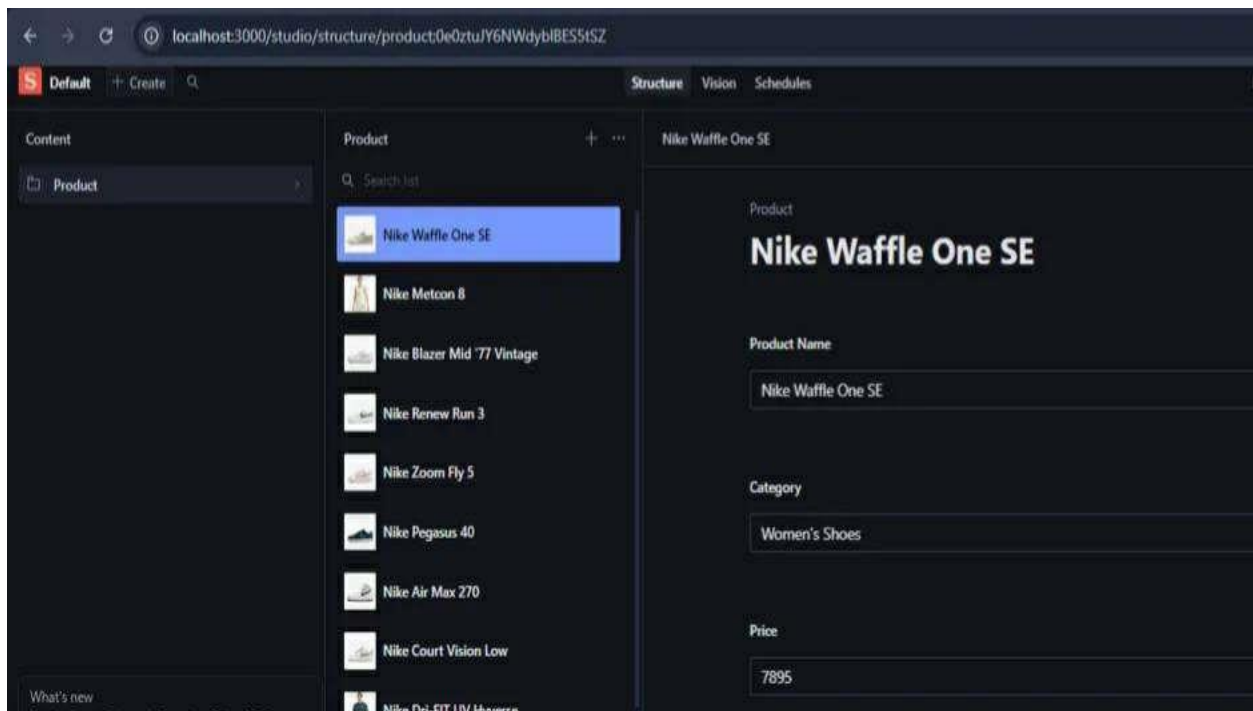
```
const sanityProduct = {
  _type: 'product',
  productName: product.productName,
  category: product.category,
  price: product.price,
  inventory: product.inventory,
  colors: product.colors || [],
  status: product.status,
  description: product.description,
  image: imageRef ? {
    _type: 'image',
    asset: {
      _type: 'reference',
      _ref: imageRef,
    },
  },
  } : undefined,
};
await client.create(sanityProduct);
```

SCREEN SHOTS :

FRONTED DISPLAY :



POPULATED SANITY CMS FIELDS:



MIGRATION STEPS AND TOOL USED :

- Back up Data
- Update Schema
- Deploy Updated Schema
- Inspect Data
- Migrate Data
- Test Locally
- Validate Date
- Deploy to Production

TOOLS USED :

1. Sanity CLI

- For exporting and importing datasets and deploying schema updates.

2. Sanity Studio

- To update and test schema changes locally before deploying to production.

3. Sanity Dashboard

- To monitor and manage the live dataset and environment.

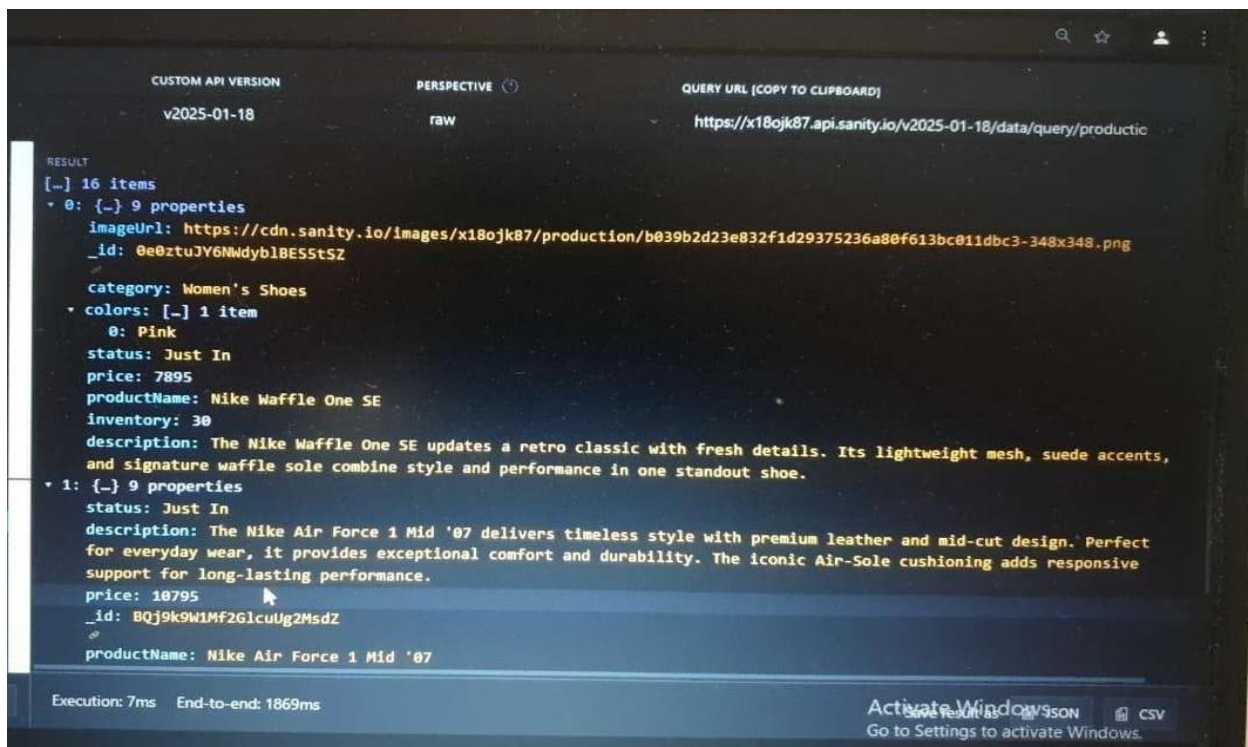
4. GROQ Queries

- For querying and inspecting existing documents to find missing or invalid data.

5. Custom Scripts (Optional)

- For large-scale migrations, use JavaScript and the Sanity Client to automate updates to multiple documents.

DATA FETCHED :

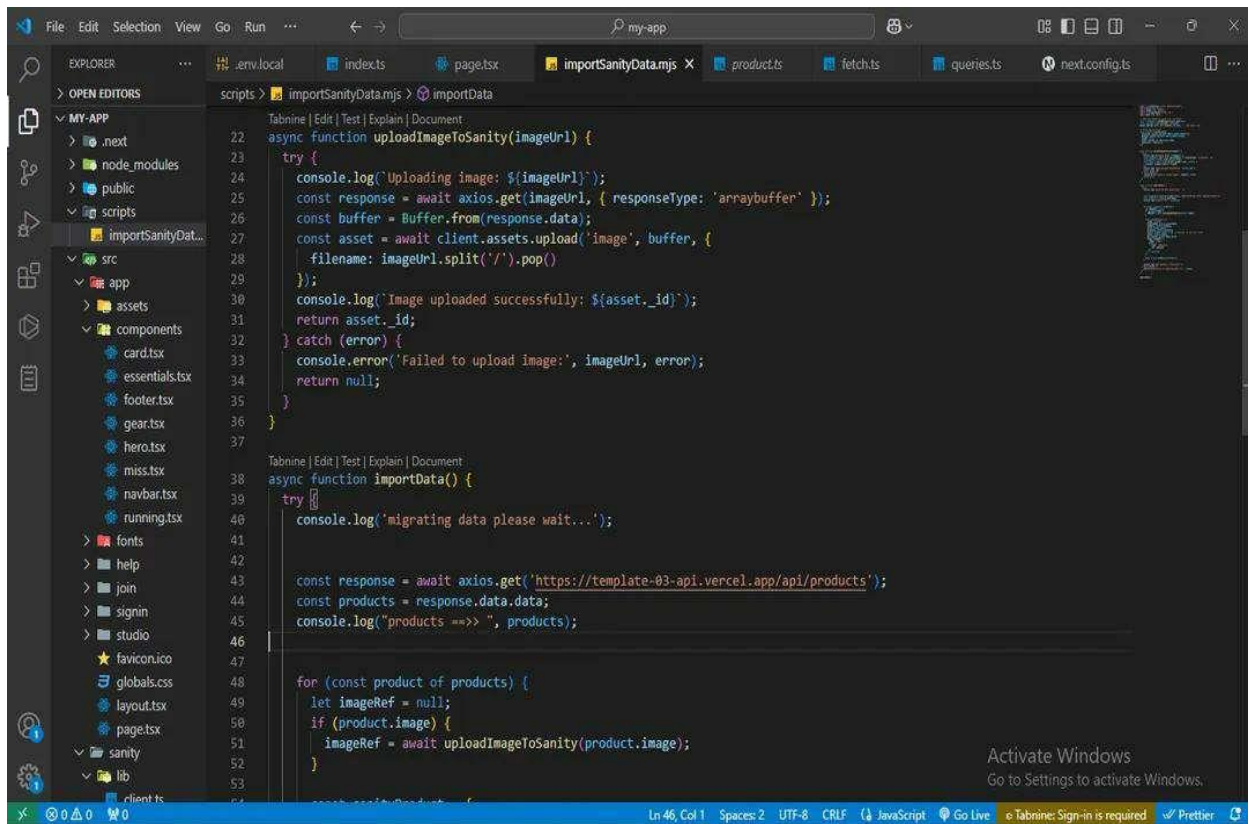


The screenshot shows a web browser interface with a dark theme. At the top, there are tabs for 'CUSTOM API VERSION' (v2025-01-18), 'PERSPECTIVE' (raw), and 'QUERY URL (COPY TO CLIPBOARD)' (https://x18ojk87.api.sanity.io/v2025-01-18/data/query/productic). The main content area displays the 'RESULT' of a query, showing two items. Item 0 is a product with properties like imageUrl, _id, category, colors, status, price, productName, inventory, and description. Item 1 is another product with similar properties. At the bottom, there is a status bar showing 'Execution: 7ms End-to-end: 1869ms' and an 'Activate Windows' watermark.

```
RESULT
[ ] 16 items
  0: { } 9 properties
    imageUrl: https://cdn.sanity.io/images/x18ojk87/production/b039b2d23e832f1d29375236a80f613bc011dbc3-348x348.png
    _id: 0e0ztuJY6NWdyblBESStSZ
    category: Women's Shoes
    colors: [ ] 1 item
      0: Pink
    status: Just In
    price: 7895
    productName: Nike Waffle One SE
    inventory: 30
    description: The Nike Waffle One SE updates a retro classic with fresh details. Its lightweight mesh, suede accents, and signature waffle sole combine style and performance in one standout shoe.
  1: { } 9 properties
    status: Just In
    description: The Nike Air Force 1 Mid '07 delivers timeless style with premium leather and mid-cut design. Perfect for everyday wear, it provides exceptional comfort and durability. The iconic Air-Sole cushioning adds responsive support for long-lasting performance.
    price: 10795
    _id: BQj9k9W1Mf2GlcUg2MsdZ
    productName: Nike Air Force 1 Mid '07

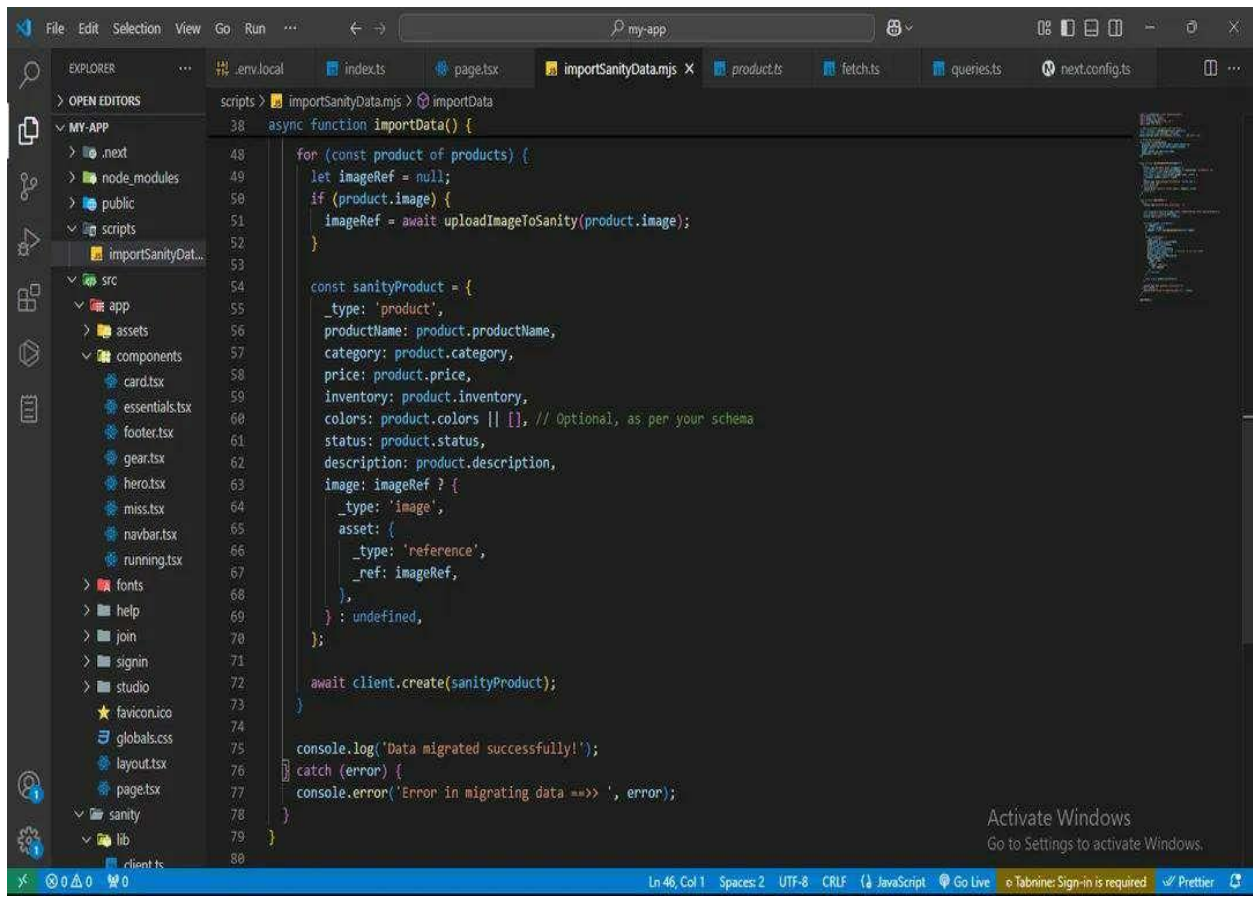
Execution: 7ms End-to-end: 1869ms
```

CODE SNIPPETS FOR MIGRATION SCRIPTS :



The screenshot shows a code editor with a dark theme. The left sidebar shows a file explorer with a project structure including 'MY-APP', 'node_modules', 'public', 'scripts', 'src', 'app', 'assets', 'components', 'card.tsx', 'essentials.tsx', 'footer.tsx', 'gear.tsx', 'hero.tsx', 'miss.tsx', 'navbar.tsx', 'running.tsx', 'fonts', 'help', 'join', 'signin', 'studio', 'faviconico', 'globals.css', 'layout.tsx', 'page.tsx', 'sanity', 'lib', and 'client.ts'. The main editor area shows two code snippets. The first snippet is a function 'uploadImageToSanity' that takes an 'imageUrl' and uploads it to Sanity. The second snippet is a function 'importData' that fetches product data from a template API and uploads it to Sanity. At the bottom, there is a status bar showing 'Ln 46, Col 1 Spaces: 2 UTF-8 CRLF JavaScript Go Live e Tabnine: Sign-in is required Prettier' and an 'Activate Windows' watermark.

```
scripts > importSanityData.mjs > importData
Tabnine | Edit | Test | Explain | Document
22 async function uploadImageToSanity(imageUrl) {
23   try {
24     console.log('Uploading image: ${imageUrl}');
25     const response = await axios.get(imageUrl, { responseType: 'arraybuffer' });
26     const buffer = Buffer.from(response.data);
27     const asset = await client.assets.upload('image', buffer, {
28       filename: imageUrl.split('/').pop()
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 async function importData() {
39   try {
40     console.log('migrating data please wait...');
41
42
43     const response = await axios.get('https://template-03-api.vercel.app/api/products');
44     const products = response.data.data;
45     console.log('products ==> ', products);
46
47
48     for (const product of products) {
49       let imageRef = null;
50       if (product.image) {
51         imageRef = await uploadImageToSanity(product.image);
52       }
53     }
54   }
55 }
```

-----THE END-----