

NAME: RUBA HAROON
ROLL NO: 00441882
SLOT: FRIDAY 9 TO 12
TEACHER: SIR HAMZAH SYED

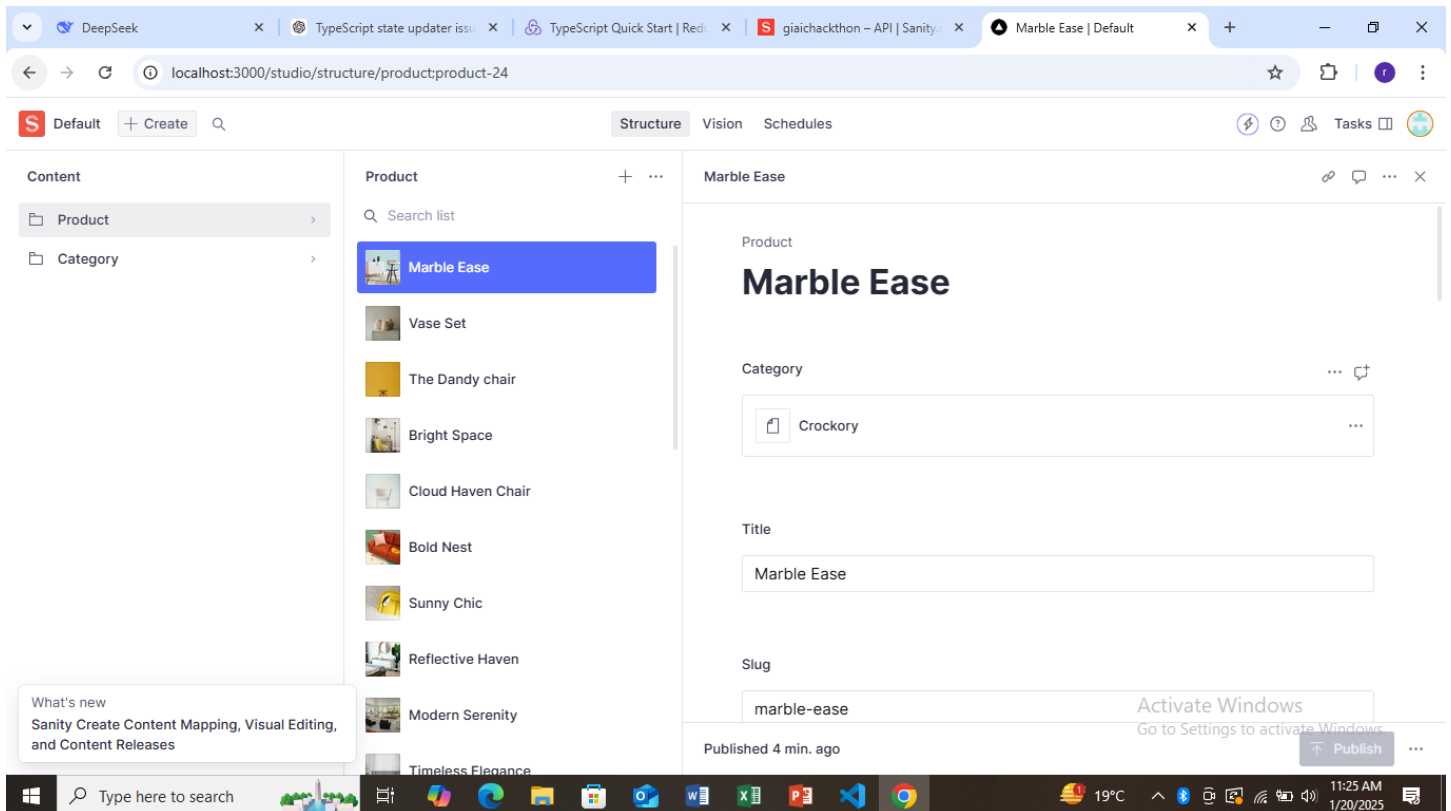
API Integration Report (AVION):

1) Reviewed API Documentation:

- I carefully read the provided API documentation for my assigned template to understand the available endpoint (/products).
- Identified the structure of the data are turned by the API, including fieldnames and data types.

2) Set Up API Calls:

- I used Thunder client to test the API endpoint and ensure the data was being returned correctly. I created utility functions in my Next.js project to fetch data from the API.
- I used fetch to make GET requests to the API end points and stored the responses in variables. I logged the API responses in the console to verify the data structure.



3) Compared API Data with Sanity Schema:

- I reviewed the API data structure and compared it with the existing schema in Sanity CMS. I identified mismatches in field names and data types.
- I updated the Sanity schema to match the API data structure. For example:
 - ❖ API Field: `product_title` → SanityField: `name`
 - ❖ API Field: `price` → SanityField: `price` (with proper data type)

I added new fields in Sanity CMS to accommodate additional data from the API, because the API is not enough to complete my website products and their details.

```
import { defineType, defineField } from "sanity";

export const product = defineType({
  name: "product",
  title: "Product",
```

```
type: "document",
fields: [
  defineField({
    name: "category",
    title: "Category",
    type: "reference",
    to: [
      {
        type: "category",
      },
    ],
  }),
  defineField({
    name: "name",
    title: "Title",
    validation: (rule) => rule.required(),
    type: "string",
  }),
  defineField({
    name: "slug",
    title: "Slug",
    validation: (rule) => rule.required(),
    type: "slug",
  }),
  defineField({
    name: "image",
    type: "image",
    validation: (rule) => rule.required(),
    title: "Product Image",
  }),
  defineField({
    name: "price",
    type: "number",
    validation: (rule) => rule.required(),
    title: "Price",
  }),
  defineField({
    name: "quantity",
    title: "Quantity",
    type: "number",
    validation: (rule) => rule.min(0),
  }),
  defineField({
    name: "tags",
```

```

    type: "array",
    title: "Tags",
    of: [
      {
        type: "string",
      },
    ],
  )),
defineField({
  name: "description",
  title: "Description",
  type: "text",
  description: "Detailed description of the product",
}),
defineField({
  name: "features",
  title: "Features",
  type: "array",
  of: [{ type: "string" }],
  description: "List of key features of the product",
}),
defineField({
  name: "dimensions",
  title: "Dimensions",
  type: "object",
  fields: [
    { name: "height", title: "Height", type: "string" },
    { name: "width", title: "Width", type: "string" },
    { name: "depth", title: "Depth", type: "string" },
  ],
  description: "Dimensions of the product",
}),
defineField({
  name: "stock",
  title: "Stock",
  type: "number",
  validation: (rule) => rule.required(),
}),
defineField({
  name: "inStock",
  title: "In Stock",
  type: "boolean",
  validation: (rule) => rule.required(),
}),

```

```

defineField({
  name: "price_id",
  title: "Stripe Price ID",
  type: "string",
}),
],

preview: {
  select: {
    title: "name",
    media: "image",
    subtitle: "price",
    inStock: "inStock",
    stock: "stock",
  },
  prepare(selection) {
    const { title, subtitle, media, inStock, stock } = selection;
    return {
      title,
      subtitle: `${subtitle} | ${inStock ? `In Stock (${stock})` : "Out of Stock"}`,
      media,
    };
  },
},
});

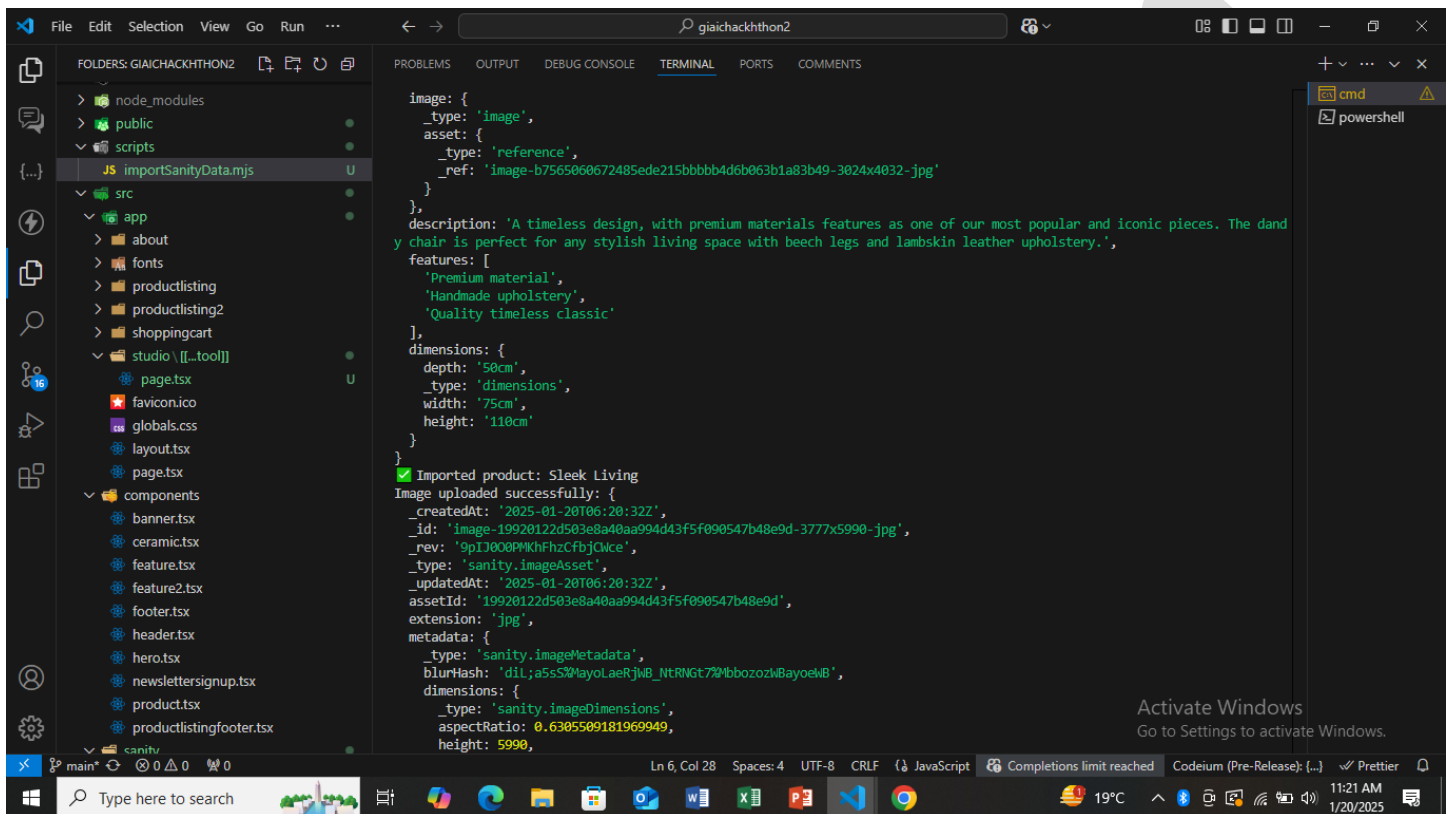
```

4) To migrate data from API to Sanity CMS:

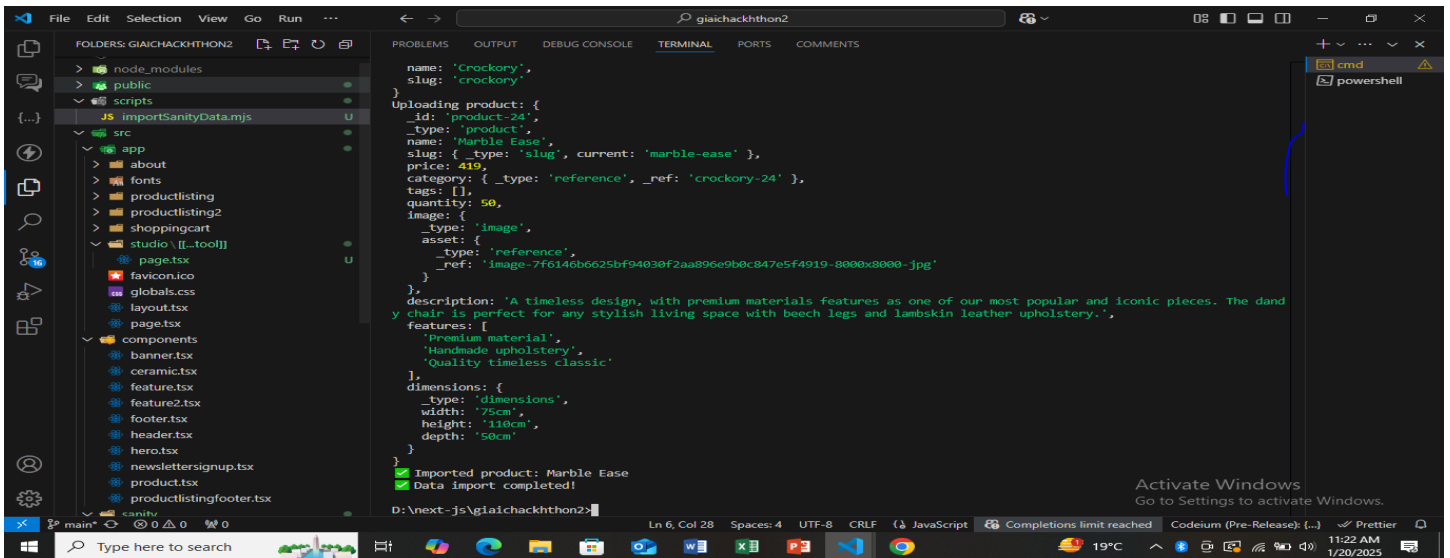
I followed these steps:

- I decided to use the provided API to fetch data and write a script to import it into Sanity CMS.
- I created a script Folder and then I created a migration (.mjs) file to fetch data from the API and transform it into the format required by Sanity CMS.
- I used the Sanity client library to upload the data to the CMS. I ran the migration script to import product data, categories, and other relevant information into Sanity CMS.
- I verified the imported data by checking the Sanity dashboard and ensuring all fields were correctly populated.

In this project, I successfully integrated the provided API into my Next.js frontend and migrated data in to Sanity CMS. I adjusted the schema to match the API data structure and ensured the data was accurately displayed in the frontend. This exercise helped me gain practical experience in API integration, data migration, and schema validation, which are essential skills for building scalable marketplaces.



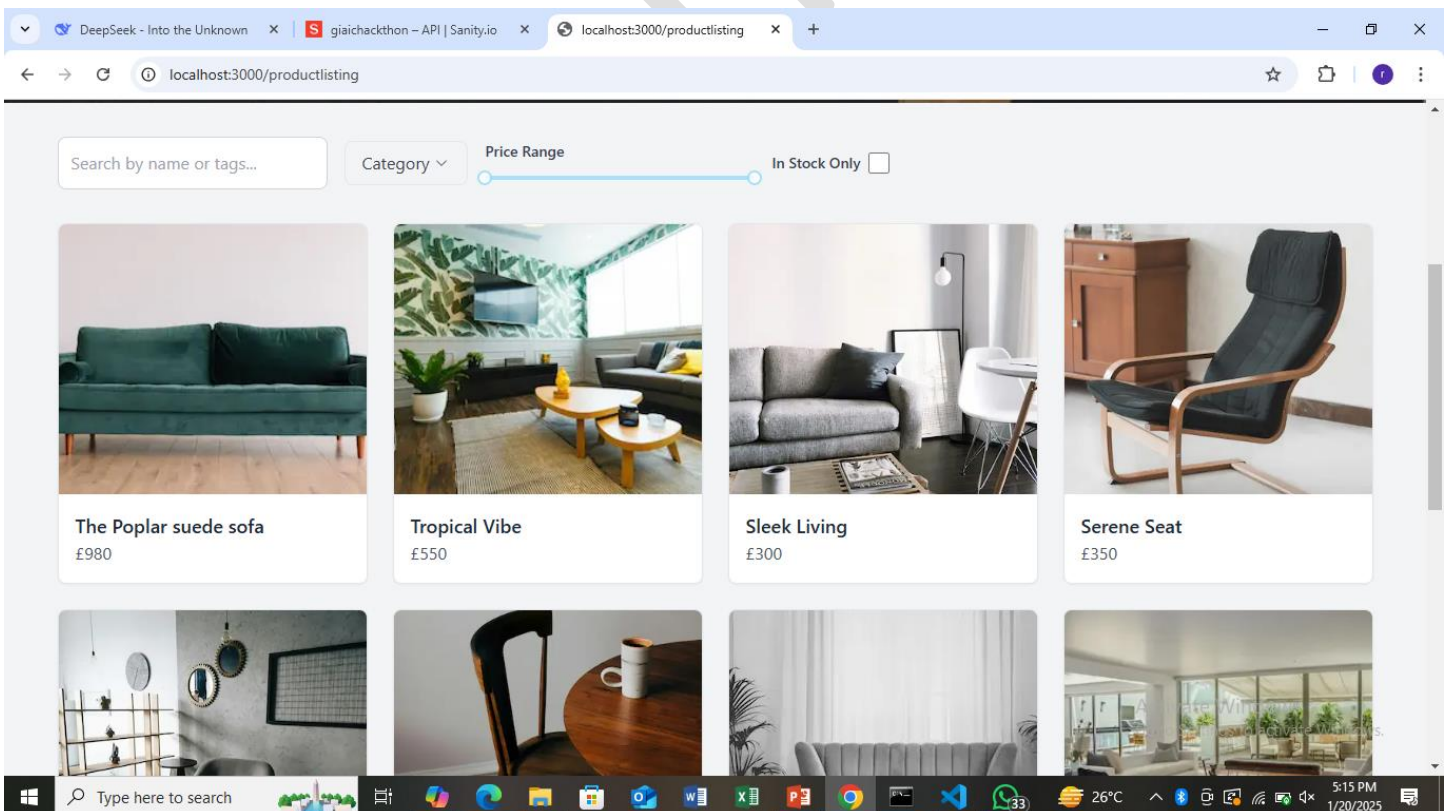
```
image: {
  _type: 'image',
  asset: {
    _type: 'reference',
    _ref: 'image-b7565060672485ede215bbbbb4d6b063b1a83b49-3024x4032-jpg'
  },
  description: 'A timeless design, with premium materials features as one of our most popular and iconic pieces. The dandy chair is perfect for any stylish living space with beech legs and lambskin leather upholstery.',
  features: [
    'Premium material',
    'Handmade upholstery',
    'Quality timeless classic'
  ],
  dimensions: {
    depth: '50cm',
    _type: 'dimensions',
    width: '75cm',
    height: '110cm'
  }
}
✓ Imported product: Sleek Living
Image uploaded successfully: {
  _createdAt: '2025-01-20T06:20:32Z',
  _id: 'image-19920122d503e8a40aa994d43f5f090547b48e9d-3777x5990-jpg',
  _rev: '9pIJ000PMKhFhzCfbjCWce',
  _type: 'sanity.imageAsset',
  _updatedAt: '2025-01-20T06:20:32Z',
  assetId: '19920122d503e8a40aa994d43f5f090547b48e9d',
  extension: 'jpg',
  metadata: {
    _type: 'sanity.imageMetadata',
    blurHash: 'dIL;a5sS7M4yoLaeRjMB_NtRNGt774bbozoziMBayoeMB',
    dimensions: {
      _type: 'sanity.imageDimensions',
      aspectRatio: 0.6305509181969949,
      height: 5990,
```



DATA SUCCESSFULLY IMPORTED ON SANITY:

Data successfully gets imported.

DATA FETCH ON FRONT END:



Submission Report:

Task	Status
API Understanding	✓
Schema Validation	✓
Data Migration	✓
API Integration in Next JS	✓
Submission Preparation	✓