

API Integration Report – Avion_Ecommerce

API

<https://hackathon-apis.vercel.app/api/products>

Sanity Schema

```
my-app > src > sanity > schemasTypes > products > product > fields > fields > name
3 export const product = defineType({
4   title: "Product",
5   type: "document",
6   fields: [
7     defineField({
8       name: "category",
9       title: "Category",
10      type: "reference",
11      to: [{
12        type: "category"
13      }]
14    }),
15    defineField({
16      name: "name",
17      title: "Title",
18      validation: (rule) => rule.required(),
19      type: "string"
20    }),
21    defineField({
22      name: "slug",
23      title: "Slug",
24      validation: (rule) => rule.required(),
25      type: "slug"
26    }),
27    defineField({
28      name: "image",
29      type: "image",
30      validation: (rule) => rule.required(),
31      title: "Product Image"
32    }),
33    defineField({
34      name: "price",
35      type: "number",
36      validation: (rule) => rule.required(),
37      title: "Price",
38    }),
39    defineField({
40      name: "quantity",
41      title: "Quantity",
42      type: "number",
43      validation: (rule) => rule.min(0),
44    }),
45    defineField({
46      name: "tags",
47      type: "array",
48      title: "Tags",
49      of: [{
50        type: "string"
51      }]
52    })
53  ]
54 })
```

```
my-app > src > sanity > schemasTypes > products > product > fields > fields > name
7 fields: [
8   defineField({
9     name: "tags",
10    type: "array",
11    title: "Tags",
12    of: [{
13      type: "string"
14    }]
15  }),
16  defineField({
17    name: "description",
18    title: "Description",
19    type: "text",
20    description: "Detailed description of the product",
21  }),
22  defineField({
23    name: "features",
24    title: "Features",
25    type: "array",
26    of: [{ type: "string" }],
27    description: "List of key features of the product",
28  }),
29  defineField({
30    name: "dimensions",
31    title: "Dimensions",
32    type: "object",
33    fields: [
34      { name: "height", title: "Height", type: "string" },
35      { name: "width", title: "Width", type: "string" },
36      { name: "depth", title: "Depth", type: "string" },
37    ],
38    description: "Dimensions of the product",
39  })
40 ]
41 })
```

```
my-app > src > sanity > schemasTypes > category.ts > Category
1 import { defineType, defineField } from "sanity";
2
3 export const Category = defineType({
4   name: "category",
5   title: "Category",
6   type: "document",
7   fields: [
8     defineField({
9       name: "name",
10      title: "Name",
11      type: "string",
12      validation: (rule) => rule.required(),
13    }),
14    defineField({
15      name: "slug",
16      title: "Slug",
17      type: "slug",
18      validation: (rule) => rule.required(),
19      options: {
20        source: "name",
21      }
22    })
23  ]
24 })
```

Product.ts

Category.ts

Data Migration Script

```
my-app > scripts > importData.mjs > client
1 import { createClient } from '@sanity/client';
2 import axios from 'axios';
3 import dotenv from 'dotenv';
4 import { fileURLToPath } from 'url';
5 import path from 'path';
6 import slugify from 'slugify';
7
8 const __filename = fileURLToPath(import.meta.url);
9 const __dirname = path.dirname(__filename);
10 dotenv.config({ path: path.resolve(__dirname, '../.env.local') });
11
12 const client = createClient({
13   projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
14   dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
15   useCdn: false,
16   token: process.env.NEXT_PUBLIC_SANITY_AUTH_TOKEN,
17   apiVersion: '2025-01-25'
18 });
19
20 Tabnine | Edit | Test | Explain | Document
21 async function uploadImageToSanity(imageUrl) {
22   try {
23     console.log('Uploading image: ${imageUrl}');
24     const response = await axios.get(imageUrl, { responseType: 'arraybuffer' });
25     const buffer = Buffer.from(response.data);
26     const asset = await client.assets.upload('image', buffer, {
27       filename: imageUrl.split('/').pop()
28     });
29     console.log('Image uploaded successfully: ${asset.id}');
```

```
my-app > scripts > importData.mjs > client
36 async function createCategory(category, counter)
37 {
38   try {
39     const categoryExist = await client.fetch(
40       `[_type=="category" && slug.current=="${slug}[0]", { slug: category.slug }];
41     if (categoryExist) {
42       return categoryExist._id;
43     }
44     const catObj = {
45       _type: "category",
46       _id: `${category.slug}-${counter}`,
47       name: category.name,
48       slug: {
49         _type: 'slug',
50         current: slugify(category.name || 'default-category', {
51           lower: true, strict: true })
52       };
53     const response = await client.createOrReplace(catObj);
54     console.log('Category created successfully', response);
55     return response._id;
56   } catch (error) {
57     console.error('Failed to create category:', category.name, error);
58     return null;
59   }
60 }
61
62 Tabnine | Edit | Test | Explain | Document
63 async function importData() {
64   try {
65     console.log('Fetching products from API...');
```

```
my-app > scripts > importData.mjs > client
68 async function importData() {
69   try {
70     console.log('Fetching products from API...');
71     const response = await axios.get('https://hackathon-apis.vercel.app/api/products');
72     const products = response.data;
73     let counter = 1;
74
75     for (const product of products) {
76       console.log('Processing product: ${product.name}');
77       let imageRef = null;
78       let catRef = null;
79
80       if (product.image) {
81         imageRef = await uploadImageToSanity(product.image);
82       }
83       if (product.category?.name) {
84         catRef = await createCategory(product.category, counter);
85       }
86
87       const sanityProduct = {
88         _id: 'product-${counter}',
89         _type: 'product',
90         name: product.name,
91         slug: {
92           _type: 'slug',
93           current: slugify(product.name, { lower: true, strict: true })
94         },
95         price: product.price,
96         category: catRef ? { _type: 'reference',
```

DATA on Frontend

New Ceramic



Sleek Living
£ 300



Serene Seat
£ 350



Nordic Elegance
£ 280



TimberCraft
£ 320

[View collection](#)