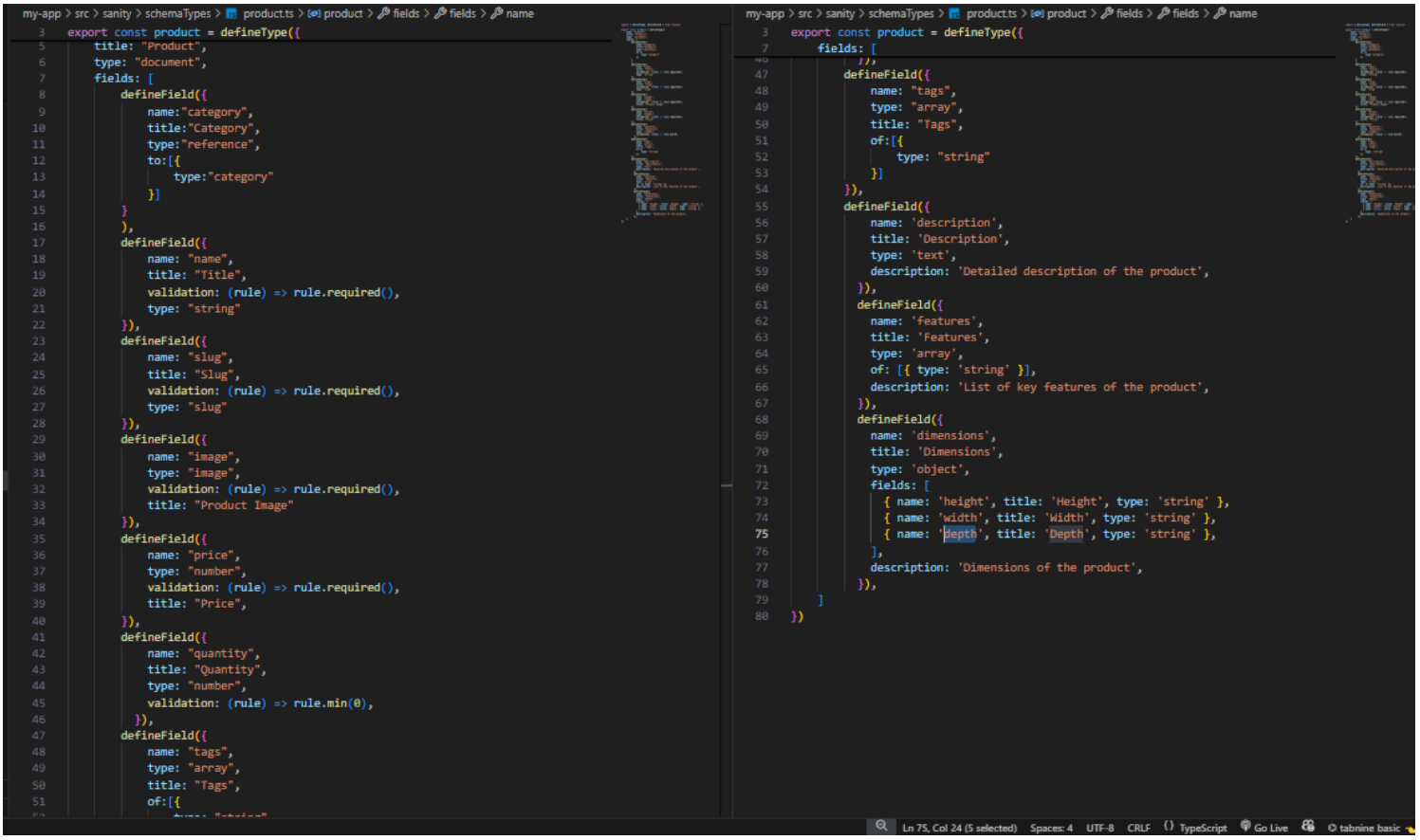
API Integration Report – Avion_Ecommerce

API

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

Sanity Schema



```
export const product = defineType({
    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:[{
```

```
export const product = defineType({
    fields: [
        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',
          }),
    ]
})
```

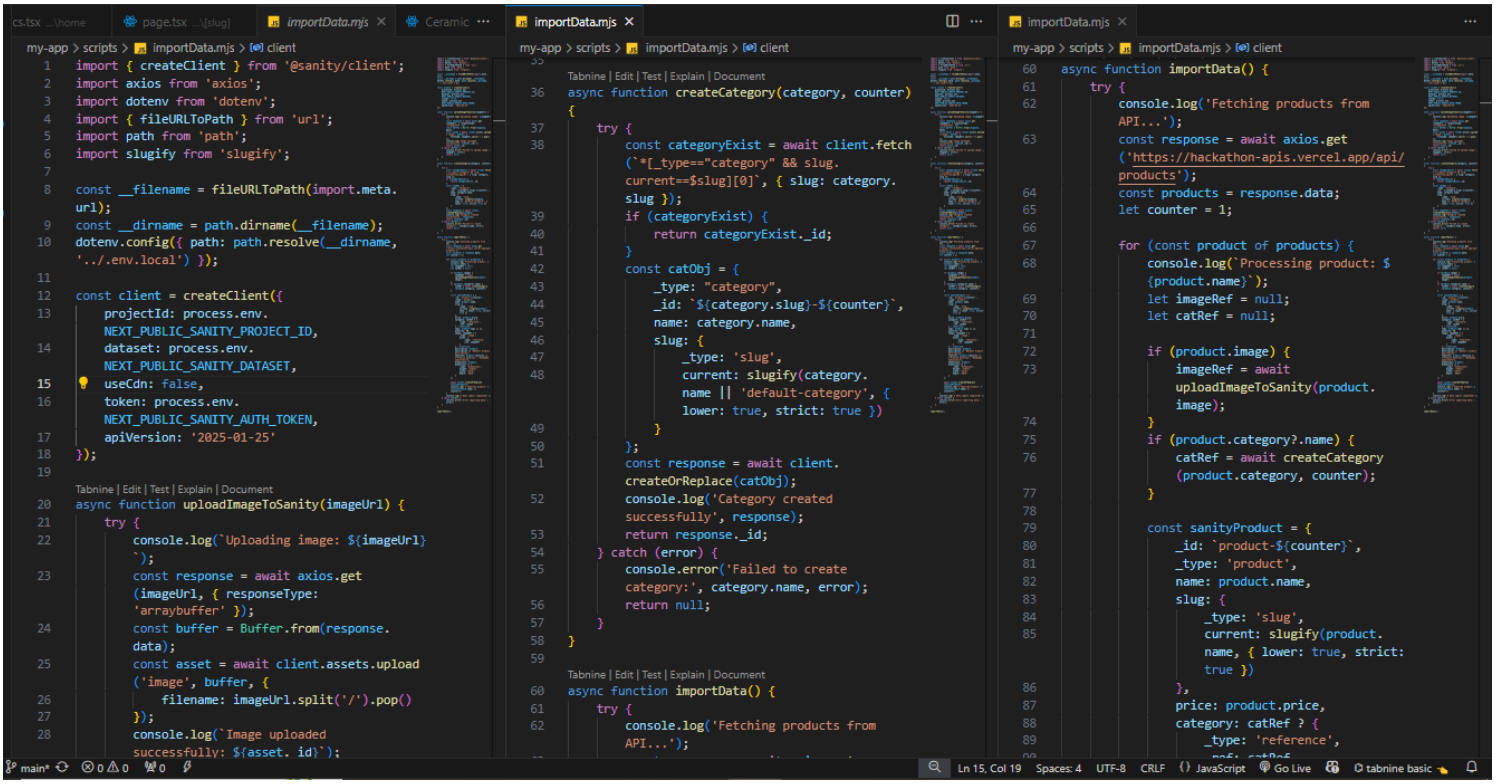
Product.ts

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

export const Category = defineType({
    name: "category",
    title: "Category",
    type: "document",
    fields:[
        defineField({
            name: "name",
            title: "Name",
            type: "string",
            validation: (rule) => rule.required(),
        }),
        defineField({
            name: "slug",
            title: "Slug",
            type: "slug",
            validation: (rule) => rule.required(),
            options: {
                source: "name",
            }
        })
    ]
})
```

Category.ts

Data Migration Script



```
import { createClient } from '@sanity/client';
import axios from 'axios';
import dotenv from 'dotenv';
import { fileURLToPath } from 'url';
import path from 'path';
import slugify from 'slugify';

const __filename = fileURLToPath(import.meta.
url);
const __dirname = path.dirname(__filename);
dotenv.config({ path: path.resolve(__dirname,
'../.env.local') });

const client = createClient({
    projectId: process.env.
    NEXT_PUBLIC_SANITY_PROJECT_ID,
    dataset: process.env.
    NEXT_PUBLIC_SANITY_DATASET,
    useCdn: false,
    token: process.env.
    NEXT_PUBLIC_SANITY_AUTH_TOKEN,
    apiVersion: '2025-01-25'
});

Tabnine | Edit | Test | Explain | Document
async function uploadImageToSanity(imageUrl) {
    try {
        console.log(`Uploading image: ${imageUrl}
        `);
        const response = await axios.get
        (imageUrl, { responseType:
        'arraybuffer' });
        const buffer = Buffer.from(response.
        data);
        const asset = await client.assets.upload
        ('image', buffer, {
            filename: imageUrl.split('/').pop()
        });
        console.log(`Image uploaded
        successfully: ${asset._id}`);
```

```
Tabnine | Edit | Test | Explain | Document
async function createCategory(category, counter)
{
    try {
        const categoryExist = await client.fetch
        (`*[_type=="category" && slug.
        current==$slug][0]`, { slug: category.
        slug });
        if (categoryExist) {
            return categoryExist._id;
        }
        const catObj = {
            _type: "category",
            _id: `${category.slug}-${counter}`,
            name: category.name,
            slug: {
                _type: 'slug',
                current: slugify(category.
                name || 'default-category', {
                lower: true, strict: true })
            }
        };
        const response = await client.
        createOrReplace(catObj);
        console.log('Category created
        successfully', response);
        return response._id;
    } catch (error) {
        console.error('Failed to create
        category:', category.name, error);
        return null;
    }
}

Tabnine | Edit | Test | Explain | Document
async function importData() {
    try {
        console.log('Fetching products from
        API...');
```

```
async function importData() {
    try {
        console.log('Fetching products from
        API...');
        const response = await axios.get
        ('https://hackathon-apis.vercel.app/api/
        products');
        const products = response.data;
        let counter = 1;

        for (const product of products) {
            console.log(`Processing product: $
            {product.name}`);
            let imageRef = null;
            let catRef = null;

            if (product.image) {
                imageRef = await
                uploadImageToSanity(product.
                image);
            }
            if (product.category?.name) {
                catRef = await createCategory
                (product.category, counter);
            }

            const sanityProduct = {
                _id: `product-${counter}`,
                _type: 'product',
                name: product.name,
                slug: {
                    _type: 'slug',
                    current: slugify(product.
                    name, { lower: true, strict:
                    true })
                },
                price: product.price,
                category: catRef ? {
                    _type: 'reference',
```

DATA on Frontend