

"Day 3 - API Integration Report - [Avion]"

Step 01(API Understanding):

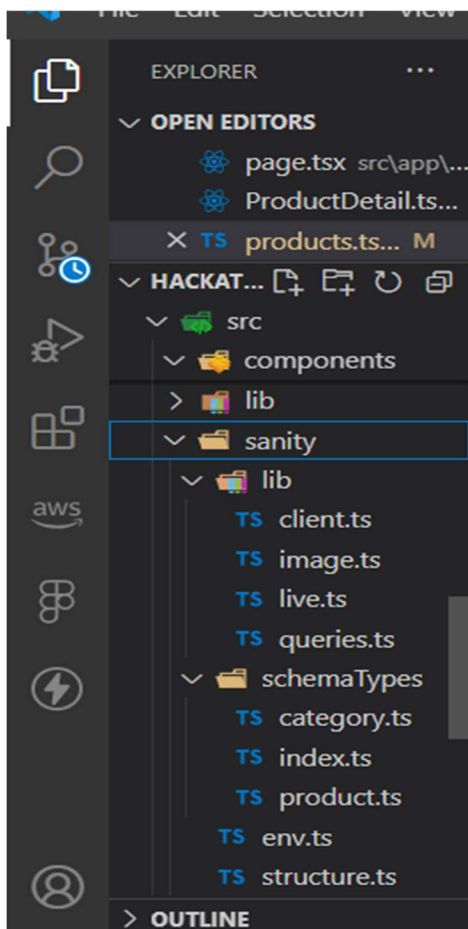
check API response understand it and test it.

Install sanity.io in next.js project.

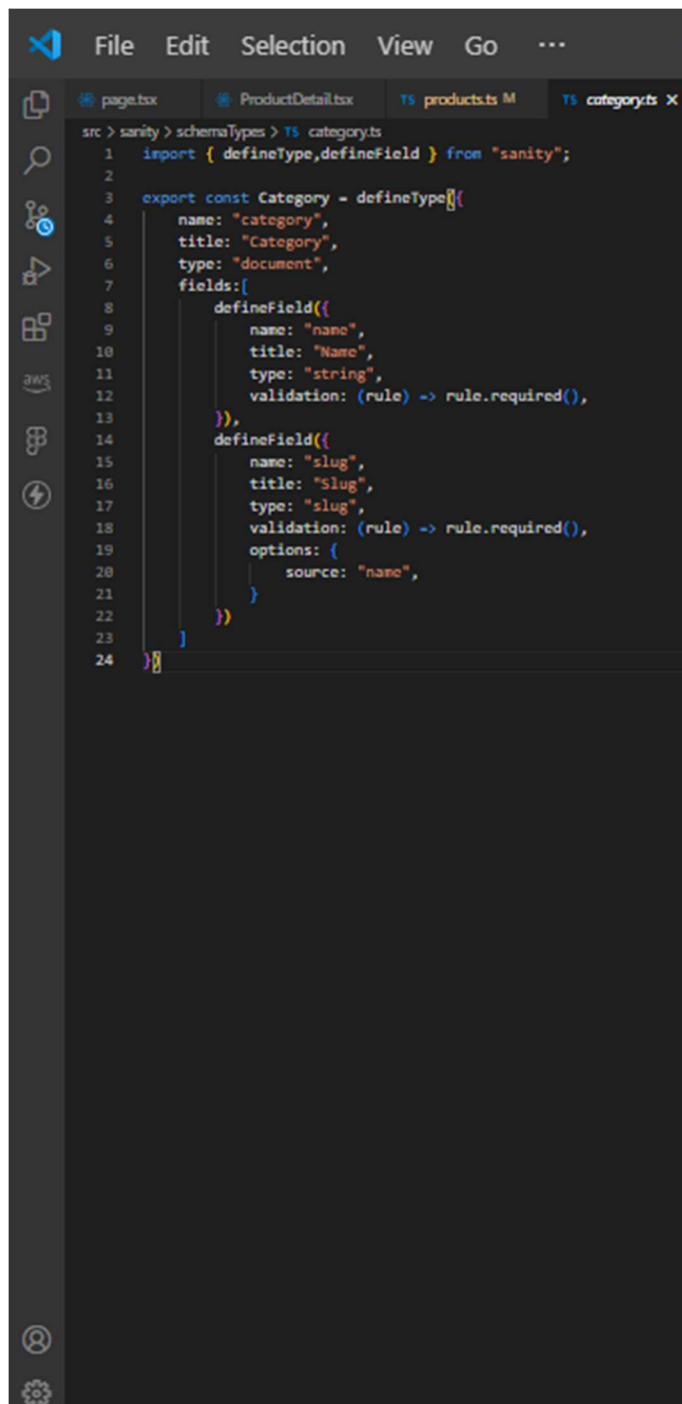
Step 02(Schema Validation):

Understand schema and check it

And go to sanity folder inside sanity folder go to schemaTypes folder make files of product.ts and category.ts and write or paste schemas



```
File Edit Selection View Go ...
pages ProductDetails products M products X
ur > tsdly > schemaType > ts product
1 import { defineType, defineField } from "sanity"
2
3 export const product = defineType({
4   name: "product",
5   title: "Product",
6   type: "document",
7   fields: [
8     defineField({
9       name: "category",
10      title: "Category",
11      type: "reference",
12      to: {
13        type: "category"
14      }
15    }),
16    defineField({
17      name: "name",
18      title: "Title",
19      validation: (rule) => rule.required(),
20      type: "string"
21    }),
22    defineField({
23      name: "slug",
24      title: "Slug",
25      validation: (rule) => rule.required(),
26      type: "slug"
27    }),
28    defineField({
29      name: "image",
30      type: "image",
31      validation: (rule) => rule.required(),
32      title: "Product Image"
33    }),
34    defineField({
35      name: "price",
36      type: "number",
37      validation: (rule) => rule.required(),
38      title: "Price"
39    }),
40    defineField({
41      name: "quantity",
42      title: "Quantity",
43      type: "number",
44      validation: (rule) => rule.min(0),
45    }),
46    defineField({
47      name: "tags",
48      type: "array",
49      title: "Tags",
50      of: {
51        type: "string"
52      }
53    }),
54    defineField({
55      name: "description",
56      title: "Description",
57      type: "text",
58      description: "Detailed description of the product",
59    }),
60    defineField({
61      name: "features",
62      title: "Features",
63      type: "array",
64      of: [{ type: "string" }],
65      description: "List of key features of the product",
66    }),
67    defineField({
68      name: "dimensions",
69      title: "Dimensions",
70      type: "object",
71      fields: [
72        { name: "height", title: "Height", type: "string" },
73        { name: "width", title: "Width", type: "string" },
74        { name: "depth", title: "Depth", type: "string" },
75      ],
76      description: "Dimensions of the product",
77    }),
78  ],
79 })
80
```



```
src > sanity > schemasTypes > TS category.ts
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 })
```

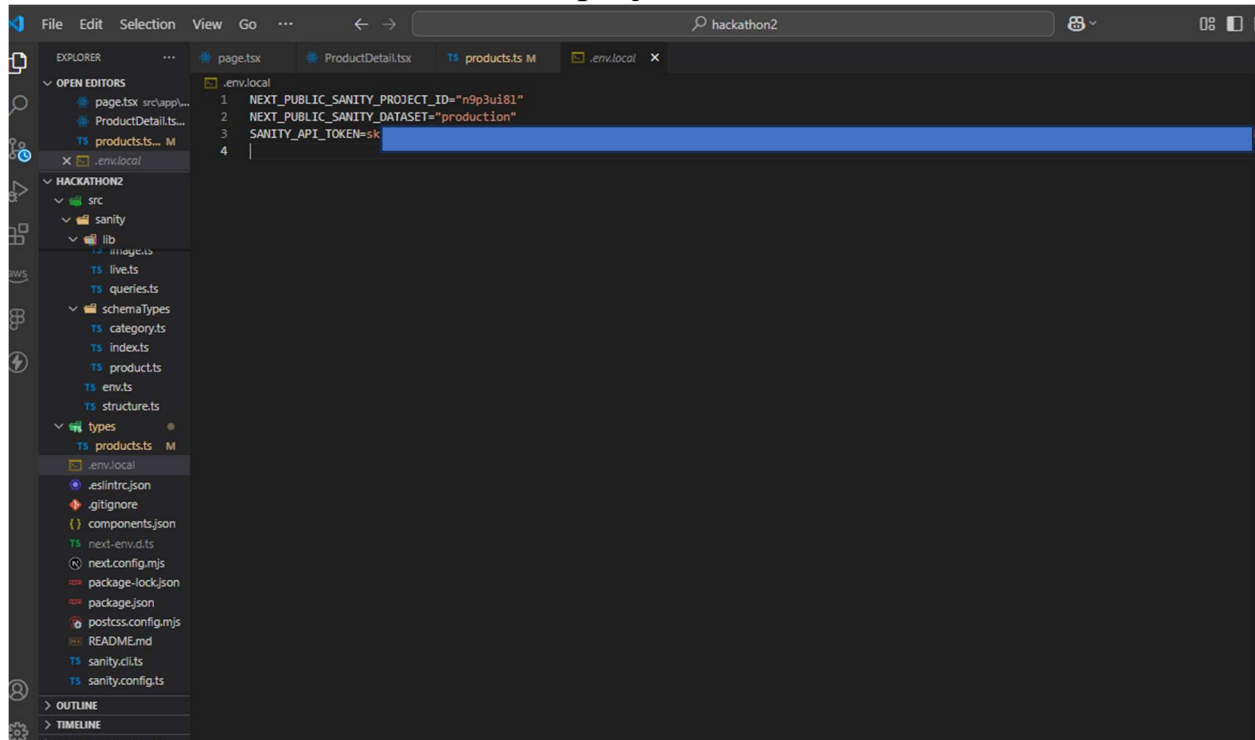
Step 03(Data Migration):

Migrate API data to sanity

Make folder called scripts in next.js project and create a file called import data.mjs like this

```
File Edit Selection View Go ...  
import { useState } from 'react';  
import axios from 'axios';  
import { useHistory } from 'react-router-dom';  
import { useParams } from 'react-router-dom';  
import { useNavigate } from 'react-router-dom';  
  
const API_URL = 'http://localhost:3001/api';  
const API_HEADERS = {  
  'Content-Type': 'application/json',  
  'Authorization': 'Bearer ' + localStorage.getItem('token')  
};  
  
const useAuth = () => {  
  const token = localStorage.getItem('token');  
  const refreshToken = localStorage.getItem('refreshToken');  
  const login = async (email, password) => {  
    const response = await axios.post(`${API_URL}/login`, { email, password }, { headers: API_HEADERS });  
    if (response.status === 200) {  
      const { token, refreshToken } = response.data;  
      localStorage.setItem('token', token);  
      localStorage.setItem('refreshToken', refreshToken);  
      return { token, refreshToken };  
    }  
    return null;  
  };  
  const logout = () => {  
    localStorage.removeItem('token');  
    localStorage.removeItem('refreshToken');  
    return null;  
  };  
  const refreshToken = async () => {  
    const response = await axios.post(`${API_URL}/refresh-token`, { refreshToken }, { headers: API_HEADERS });  
    if (response.status === 200) {  
      const { token } = response.data;  
      localStorage.setItem('token', token);  
      return token;  
    }  
    return null;  
  };  
  return { login, logout, refreshToken };  
};  
  
const useProduct = () => {  
  const { id } = useParams();  
  const [product, setProduct] = useState(null);  
  const [loading, setLoading] = useState(true);  
  const [error, setError] = useState(null);  
  const fetchProduct = async () => {  
    try {  
      const response = await axios.get(`${API_URL}/products/${id}`, { headers: API_HEADERS });  
      if (response.status === 200) {  
        setProduct(response.data);  
      }  
    } catch (error) {  
      setError(error);  
    }  
  };  
  return { product, loading, error, fetchProduct };  
};  
  
const useCategory = () => {  
  const [categories, setCategories] = useState([]);  
  const [loading, setLoading] = useState(true);  
  const [error, setError] = useState(null);  
  const fetchCategories = async () => {  
    try {  
      const response = await axios.get(`${API_URL}/categories`, { headers: API_HEADERS });  
      if (response.status === 200) {  
        setCategories(response.data);  
      }  
    } catch (error) {  
      setError(error);  
    }  
  };  
  return { categories, loading, error, fetchCategories };  
};  
  
const useCreateProduct = () => {  
  const [product, setProduct] = useState({  
    name: '',  
    description: '',  
    price: 0,  
    category: '',  
    image: ''  
  });  
  const [loading, setLoading] = useState(false);  
  const [error, setError] = useState(null);  
  const createProduct = async () => {  
    try {  
      const response = await axios.post(`${API_URL}/products`, product, { headers: API_HEADERS });  
      if (response.status === 201) {  
        setProduct(response.data);  
      }  
    } catch (error) {  
      setError(error);  
    }  
  };  
  return { product, loading, error, createProduct };  
};  
  
const useUpdateProduct = () => {  
  const { id } = useParams();  
  const [product, setProduct] = useState({});  
  const [loading, setLoading] = useState(false);  
  const [error, setError] = useState(null);  
  const updateProduct = async (product) => {  
    try {  
      const response = await axios.put(`${API_URL}/products/${id}`, product, { headers: API_HEADERS });  
      if (response.status === 200) {  
        setProduct(response.data);  
      }  
    } catch (error) {  
      setError(error);  
    }  
  };  
  return { product, loading, error, updateProduct };  
};  
  
const useDeleteProduct = () => {  
  const { id } = useParams();  
  const [loading, setLoading] = useState(false);  
  const [error, setError] = useState(null);  
  const deleteProduct = async () => {  
    try {  
      const response = await axios.delete(`${API_URL}/products/${id}`, { headers: API_HEADERS });  
      if (response.status === 200) {  
        return true;  
      }  
    } catch (error) {  
      setError(error);  
    }  
  };  
  return { loading, error, deleteProduct };  
};  
  
const useImageUpload = () => {  
  const [image, setImage] = useState('');  
  const [loading, setLoading] = useState(false);  
  const [error, setError] = useState(null);  
  const uploadImage = async (image) => {  
    try {  
      const response = await axios.post(`${API_URL}/upload-image`, { image }, { headers: API_HEADERS });  
      if (response.status === 201) {  
        setImage(response.data);  
      }  
    } catch (error) {  
      setError(error);  
    }  
  };  
  return { image, loading, error, uploadImage };  
};  
  
const useProductForm = () => {  
  const { id } = useParams();  
  const { product, loading, error, fetchProduct } = useProduct();  
  const { categories, loading: catLoading, error: catError, fetchCategories } = useCategory();  
  const { product: newProduct, loading: newProductLoading, error: newProductError, createProduct } = useCreateProduct();  
  const { product: updateProduct, loading: updateProductLoading, error: updateProductError, updateProduct } = useUpdateProduct();  
  const { loading: deleteLoading, error: deleteError, deleteProduct } = useDeleteProduct();  
  const { image, loading: imageLoading, error: imageError, uploadImage } = useImageUpload();  
  return {  
    product, loading, error, fetchProduct,  
    categories, catLoading, catError, fetchCategories,  
    newProduct, newProductLoading, newProductError, createProduct,  
    updateProduct, updateProductLoading, updateProductError, updateProduct,  
    deleteLoading, deleteError, deleteProduct,  
    image, imageLoading, imageError, uploadImage  
  };  
};  
  
export default useProductForm;
```

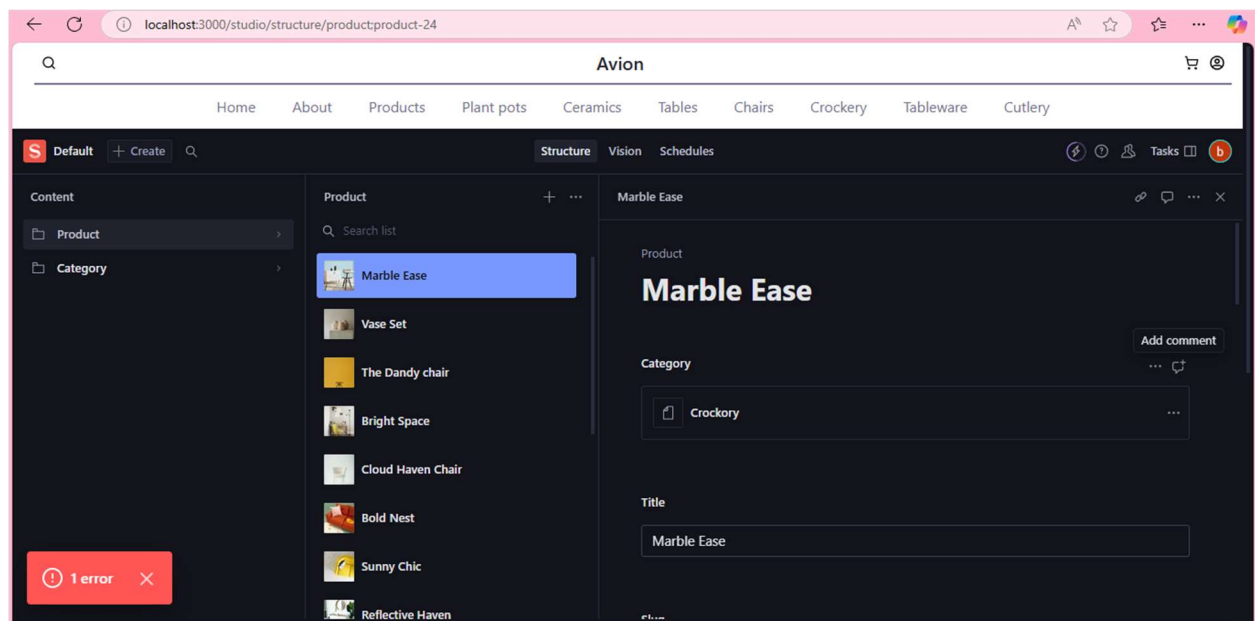
And then got to sanity of current project and see id, dataset and generate token and write in .env.local file of project like this

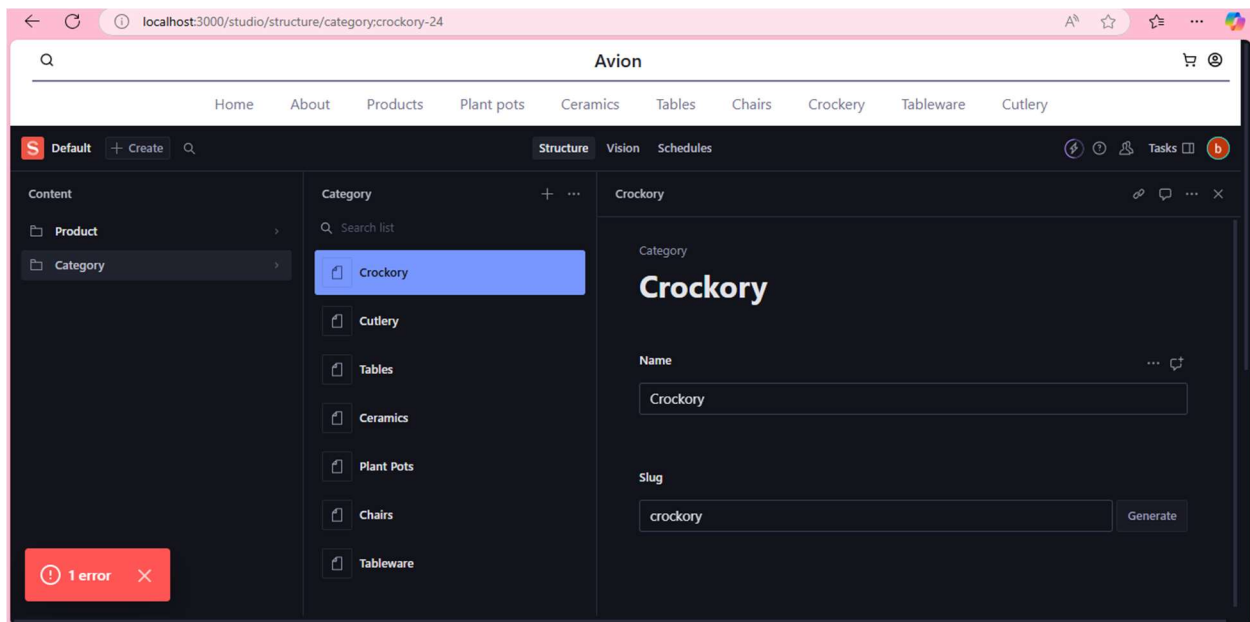


```
1 NEXT_PUBLIC_SANITY_PROJECT_ID="n9p3ui81"  
2 NEXT_PUBLIC_SANITY_DATASET="production"  
3 SANITY_API_TOKEN=sk  
4
```

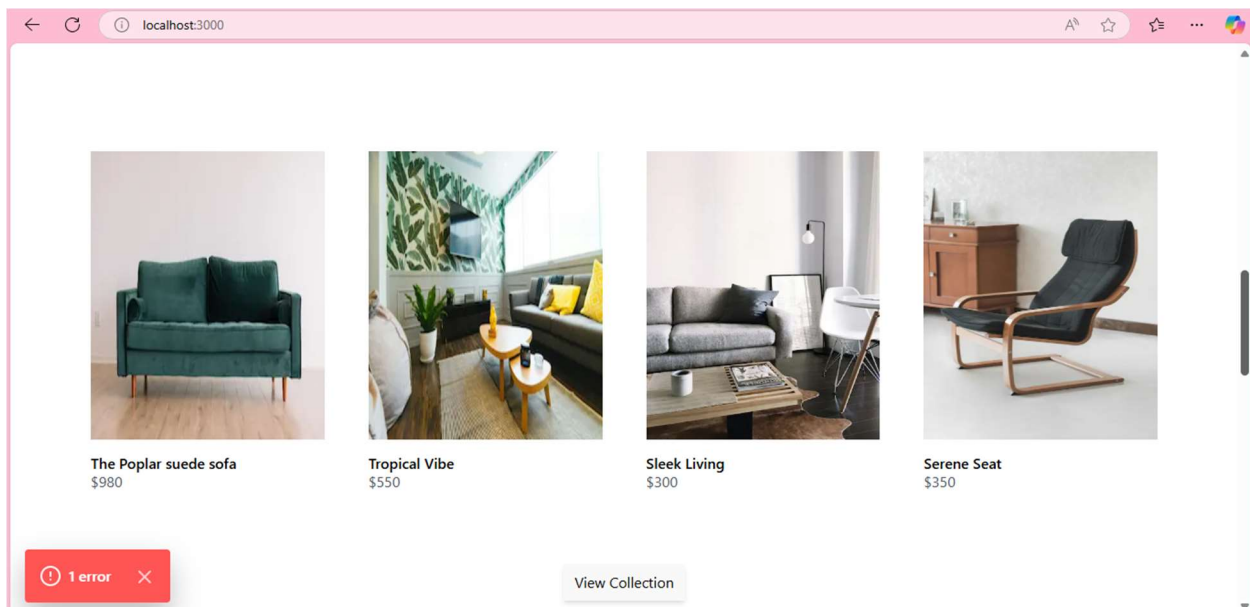
The screenshot shows a VS Code editor with the Explorer sidebar on the left. The file explorer shows a project structure for 'HACKATHON2' with folders like 'src', 'sanity', and 'lib'. The 'sanity' folder is expanded, showing files like 'live.ts', 'queries.ts', 'schemaTypes', 'category.ts', 'indexes', 'products.ts', 'env.ts', 'structure.ts', 'types', and 'products.ts M'. The 'products.ts M' file is selected, and its content is visible in the editor. The content of the file is as follows:

And make changes in package.json file in scripts add it "data": "node scripts/importdata.mjs" goto terminal and run command “npm run data” after running this command data is migrated to sanity check result is like this





Step 04(API Integration in Next.js):



Day 3 Checklist:

Self-Validation Checklist:

API Understanding: • ✓

Schema Validation: • ✓

Data Migration: • ✓

API Integration in Next.js: • ✓

Submission Preparation: • ✓