

Day 3 - API Integration Report - [e-commerce-furniture-website-furino]

API integration process.

by setting up my environment variables ,create `.env.local` project root directory. Then, added

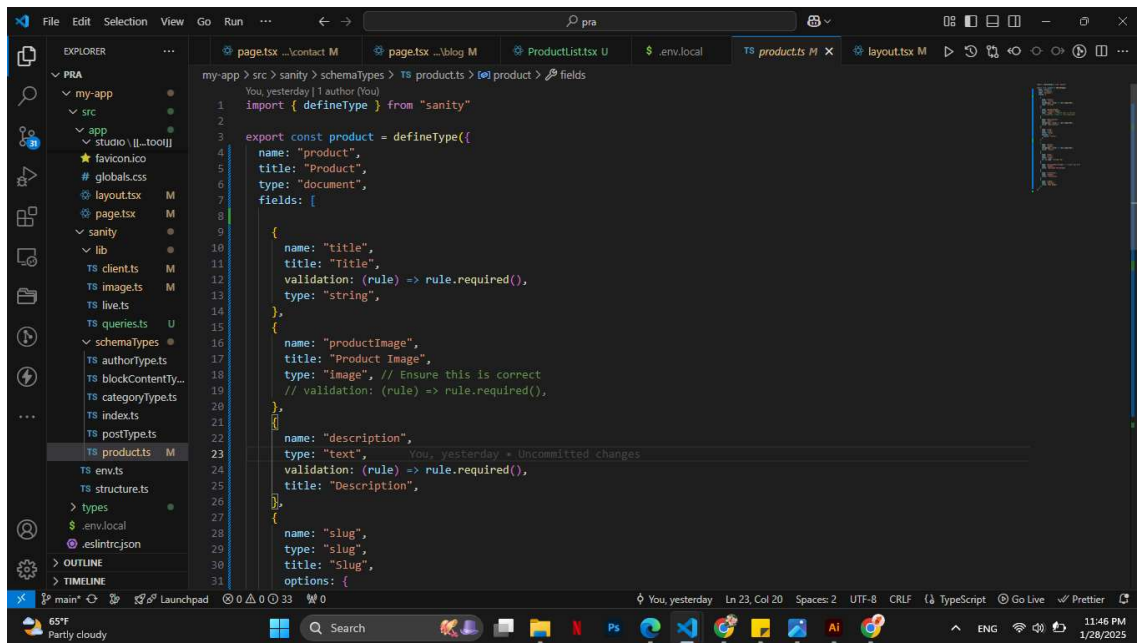
```
NEXT_PUBLIC_SANITY_PROJECT_ID=your_project_id
```

```
NEXT_PUBLIC_SANITY_DATASET=production
```

```
SANITY_API_TOKEN=your_sanity_token
```

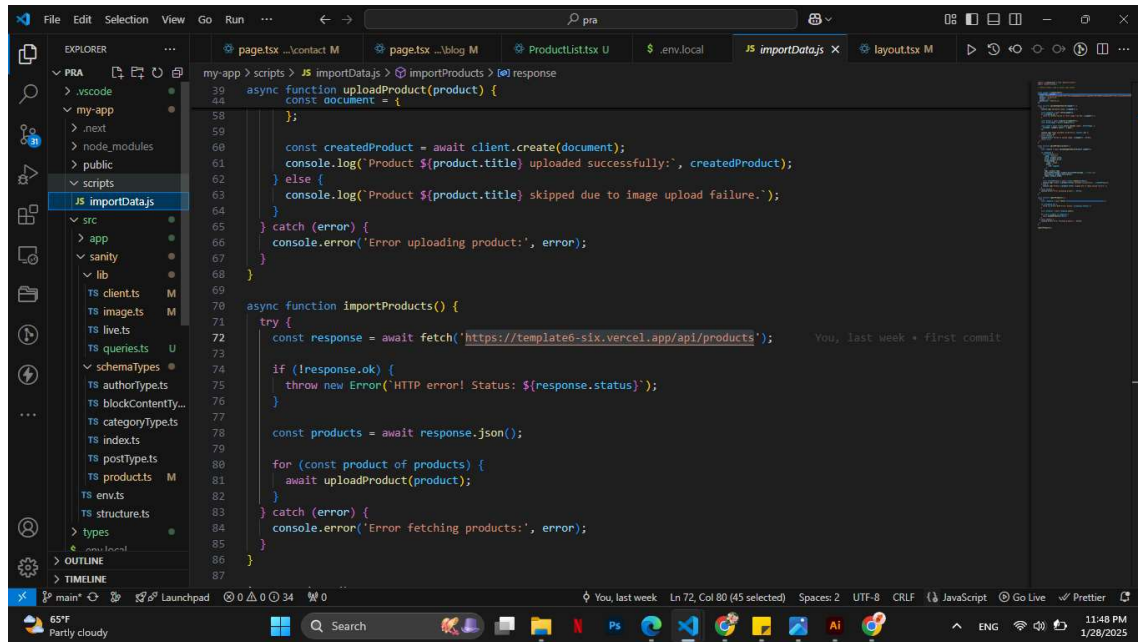
These

I created a sanity project and took the provided id ,token from it and added it to my env.local file then created a schema

A screenshot of a Visual Studio Code editor window. The Explorer sidebar on the left shows a project structure with folders like 'my-app', 'src', 'app', 'sanity', and 'schemas'. The main editor area displays a TypeScript file named 'products.ts' with a Sanity schema definition. The schema defines a 'product' type with fields for 'title', 'productImage', 'description', and 'slug'. The status bar at the bottom indicates the file is unsaved and shows the current line and column (Ln 23, Col 20).

```
1 import { defineType } from "sanity"
2
3 export const product = defineType({
4   name: "product",
5   title: "Product",
6   type: "document",
7   fields: {
8     {
9       name: "title",
10      title: "Title",
11      validation: (rule) => rule.required(),
12      type: "string",
13    },
14    {
15      name: "productImage",
16      title: "Product Image",
17      type: "image", // Ensure this is correct
18      // validation: (rule) => rule.required(),
19    },
20    {
21      name: "description",
22      type: "text",
23      validation: (rule) => rule.required(),
24      title: "Description",
25    },
26    {
27      name: "slug",
28      type: "slug",
29      title: "Slug",
30      options: {
31
```

Then created folder in root directory



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays a file tree for a project named 'my-app'. The tree includes folders like '.vscode', 'my-app', 'node_modules', 'public', 'scripts', 'src', 'app', 'sanity', 'lib', and 'types'. The 'scripts' folder is expanded, showing a file named 'importData.js'. The main editor area displays the content of 'importData.js', which contains two asynchronous functions: 'uploadProduct' and 'importProducts'. The 'uploadProduct' function takes a 'product' object and attempts to create a new product in a database. The 'importProducts' function fetches a list of products from an external API and then calls 'uploadProduct' for each item. The status bar at the bottom indicates the current file is 'main.ts', the editor is in 'main' mode, and the file encoding is 'UTF-8'.

```
my-app > scripts > JS importData.js > importProducts > response
39  async function uploadProduct(product) {
40    const document = {
58    };
59
60    const createdProduct = await client.create(document);
61    console.log('Product ${product.title} uploaded successfully:', createdProduct);
62  } else {
63    console.log('Product ${product.title} skipped due to image upload failure.');
```

Then finally run this

`npm run import-data`