Day 3 - API Integration Report – Every Thing

1. API Integration Process:

Api integration is a step that is used for my market place in sense of connecting backend and frontend, it helps to connect external service for my marketplace. It is used to seamless data flow, gives dynamic features like products listing, order tracking e.t.c. we can also use third-party Apis.

Steps:

1. I used the provided Api in document of my template and fetch it by the Api end point of "/Products".

2. I copy the script file which is given in the repo and create folder of script out the SRC directory and past it in the folder, after setting up folder I run the commad "node scripts/data-migration.mjs" then after some error the data is uploaded.

3. First error I face is the axios I installed it, second error is the Api token error w=the token I add in .env file is saved with the name "Sanity_Token" but in migrated-data file it is written as Sanity_Api_Tocken.

2. Adjustments Made to Schemas:

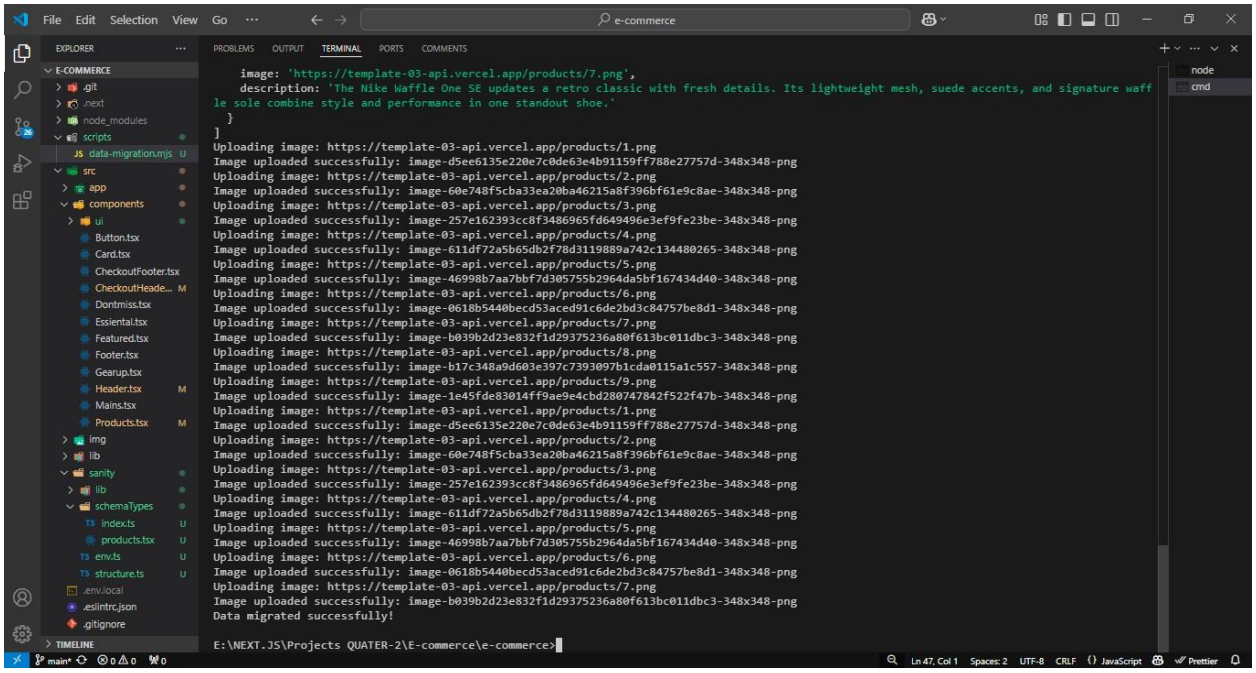
The scheme I made on day 2 is different so I do some changes, the schema of day2 is missing some properties "status" and "color", I face some different naming issues. But still error is not solved so I copy paste the schema that is provided.

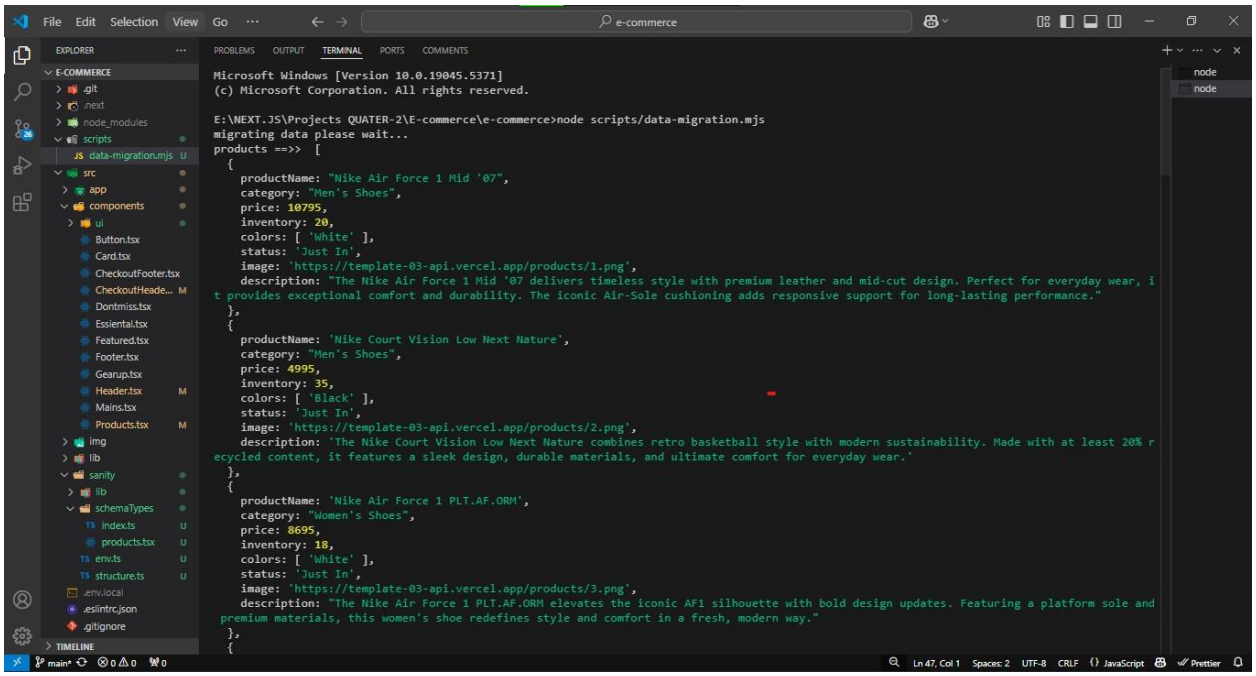
3. Migration Steps and Tools Used:

In migration data, tool I used is Sanity Client to upload data, I copy the script you provided in repo in which the Api is first fetch then upload data to sanity, validating the data is uploaded or not by sanity studio in web browser.

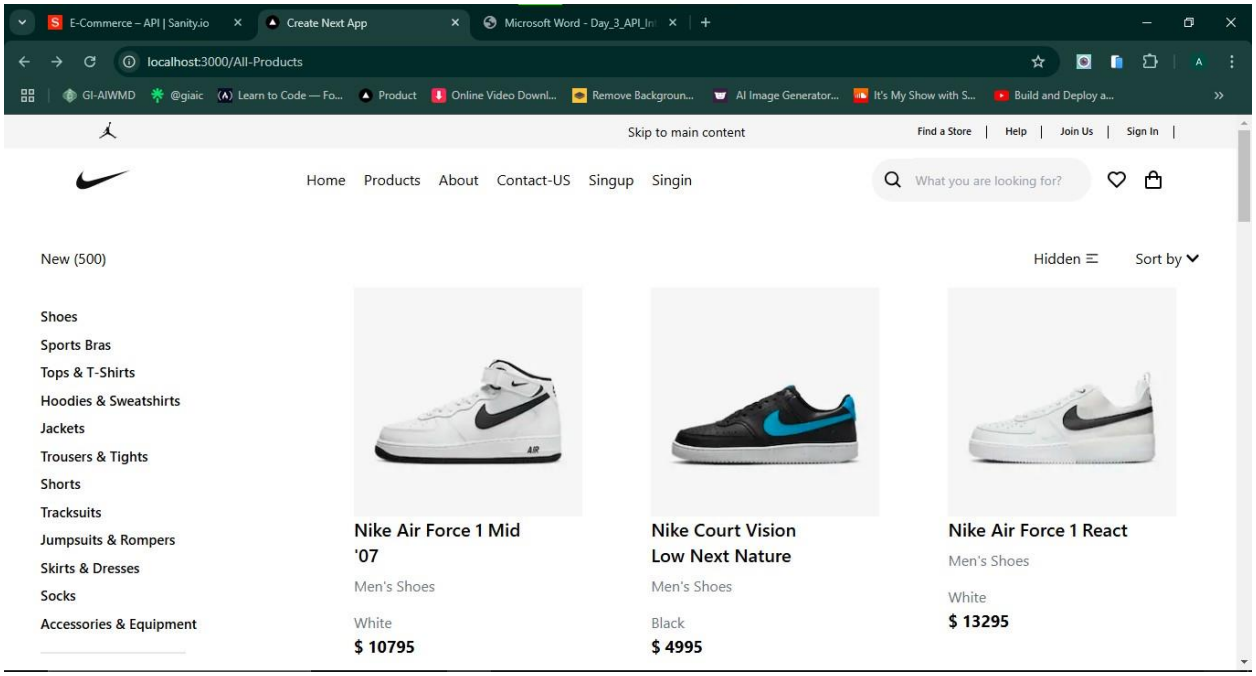
4. Screen Shots:

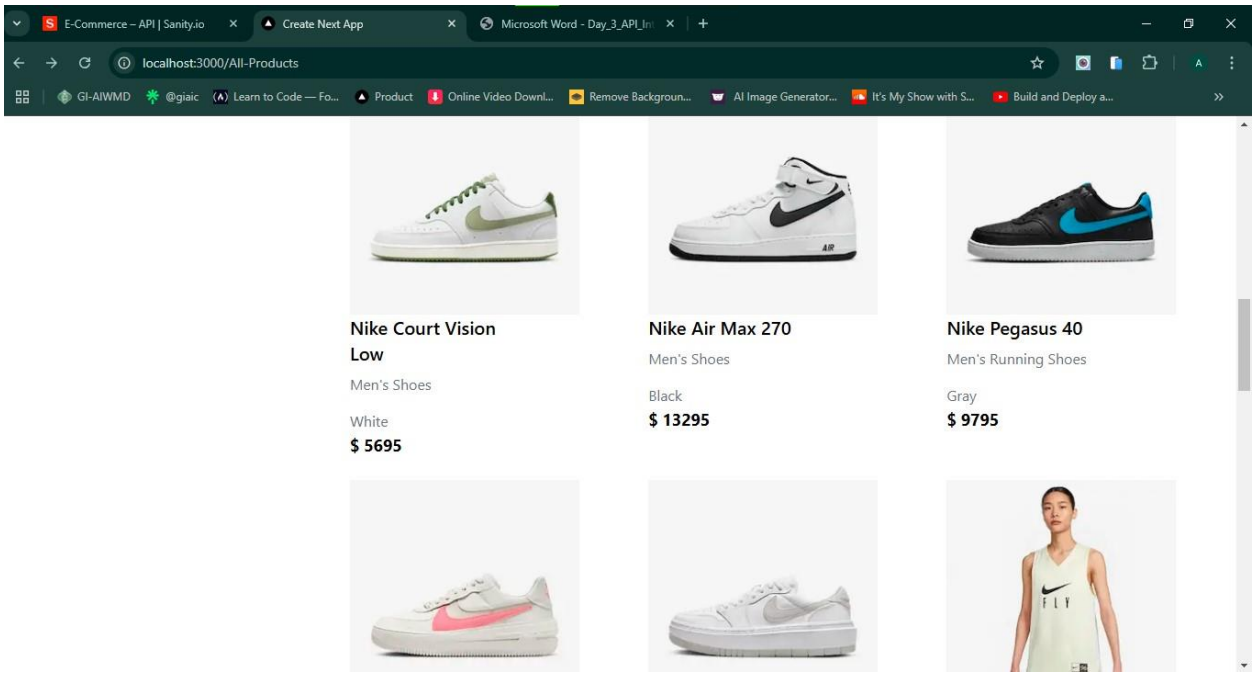
Api Call:



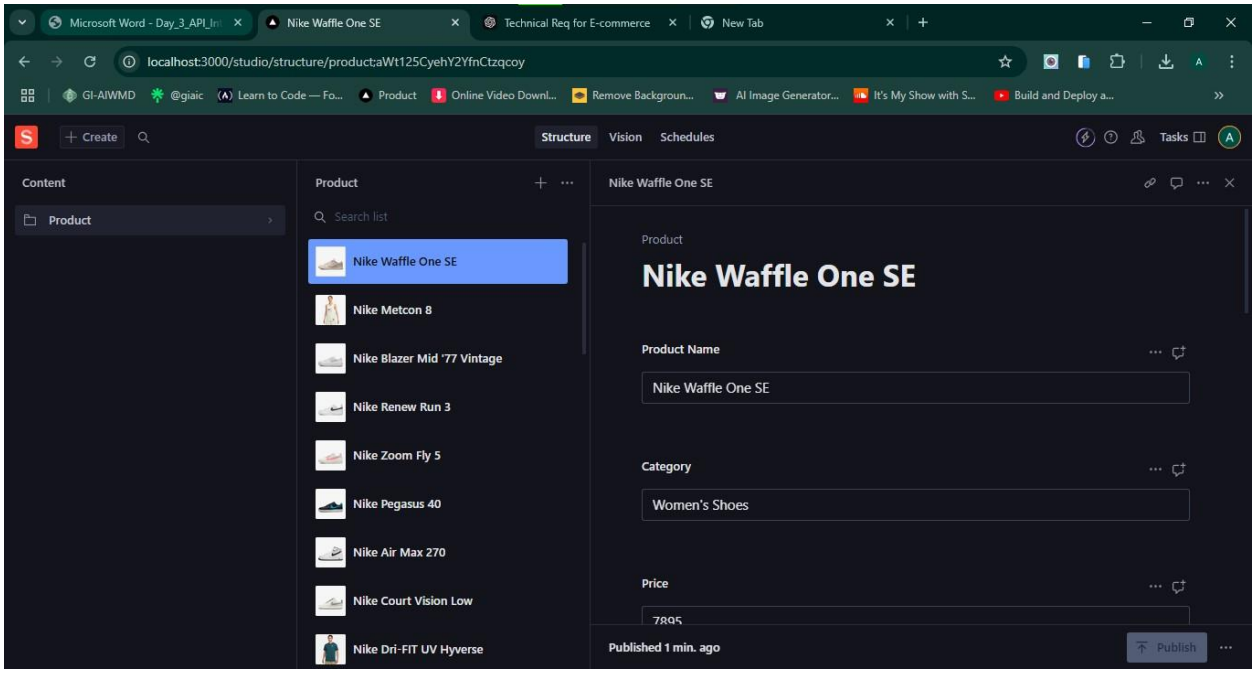


Data successfully displayed in the frontend:





Populated Sanity CMS fields:



Code snippets for API integration and migration scripts.

5. Code snippets for API integration and Migration:

For Data Migration:

Script For Migrating Data

products.tsx U data-migration.mjs U

scripts > data-migration.mjs > uploadImageToSanity

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

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

// Create Sanity client
const client = createClient({
  projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
  dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
  useCdn: false,
  token: process.env.SANITY_TOKEN,
  apiVersion: "2021-08-31",
});

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}`);
    return asset._id;
  } catch (error) {
    console.error("Failed to upload image:", imageUrl, error);
    return null;
  }
}

async function importData() {
  try {
    console.log("migrating data please wait...");

    // API endpoint containing car data
    const response = await axios.get(
      "https://template-03-api.vercel.app/api/products"
    );
    const products = response.data.data;
    console.log("products ==>> ", products);
```

products.tsx U data-migration.mjs U

scripts > data-migration.mjs > uploadImageToSanity

```
async function importData() {
    // API endpoint containing car data
    const response = await axios.get(
      "https://template-03-api.vercel.app/api/products"
    );
    const products = response.data.data;
    console.log("products ==>> ", products);

    for (const product of products) {
      let imageRef = null;
      if (product.image) {
        imageRef = await uploadImageToSanity(product.image);
      }

      const sanityProduct = {
        _type: "product",
        productName: product.productName,
        category: product.category,
        price: product.price,
        inventory: product.inventory,
        colors: product.colors || [], // Optional, as per your schema
        status: product.status,
        description: product.description,
        image: imageRef
          ? {
              _type: "image",
              asset: {
                _type: "reference",
                _ref: imageRef,
              },
            }
          : undefined,
      };

      await client.create(sanityProduct);
    }

    console.log("Data migrated successfully!");
  } catch (error) {
    console.error("Error in migrating data ==>> ", error);
  }
}

importData();
```

For Api Integration:

Fetch data from Sanity

```
interface Product {
  _id: string;
  productName: string;
  category: string;
  price: number;
  inventory: number;
  colors: string[];
  status: string;
  imageUrl: string;
  description: string;
}
const ProductDetails = () => {
  const api = `*[_type == "product"]{
  _id,
  productName,
  category,
  price,
  inventory,
  colors,
  status,
  "imageUrl": image.asset->url,
  description
}`;
  const [productData, setProductData] = useState<Product[]>([]);
  useEffect(() => {
    async function fetchData() {
      const product = await client.fetch(api);
      console.log(product);
      setProductData(product);
    }
    fetchData();
  });
```

Rendering Product Data That Is Fetched Form Sanity

```
<div className="col-span-4 md:col-span-3 ">
  <div className="grid grid-cols-1 justify-items-center md:grid-cols-3 gap-6">
    {productData.map((product) => {
      return (
        <div key={product._id}>
          <Image
            src={urlFor(product.imageUrl).url()}
            alt="Product Img"
            width={250}
            height={250}
          />
          <h3 className="text-xl font-semibold mb-2 w-[200px]">
            {product.productName}
          </h3>
          <p className="text-gray-500 mb-4 w-[200px]">
            {product.category}
          </p>
          <p className="text-gray-500">{product.colors}</p>
          <p className="text-lg font-bold">$ {product.price}</p>
        </div>
      );
    })}
  </div>
```