Day 3 - API Integration Report – Online Grocery store

a. Setting Up Sanity.io

1. **Initialize Sanity Studio:**
   - Install the Sanity CLI: npm install -g @sanity/cli
   - Run sanity init to initialize a new Sanity project.
2. **Configure CORS:**
   - Add the Next.js development URL and production URL to the CORS settings in the Sanity dashboard.

b. Adding API in Next.js

1. **Install Dependencies:**
2. npm install @sanity/client
3. **Initialize Sanity Client:**

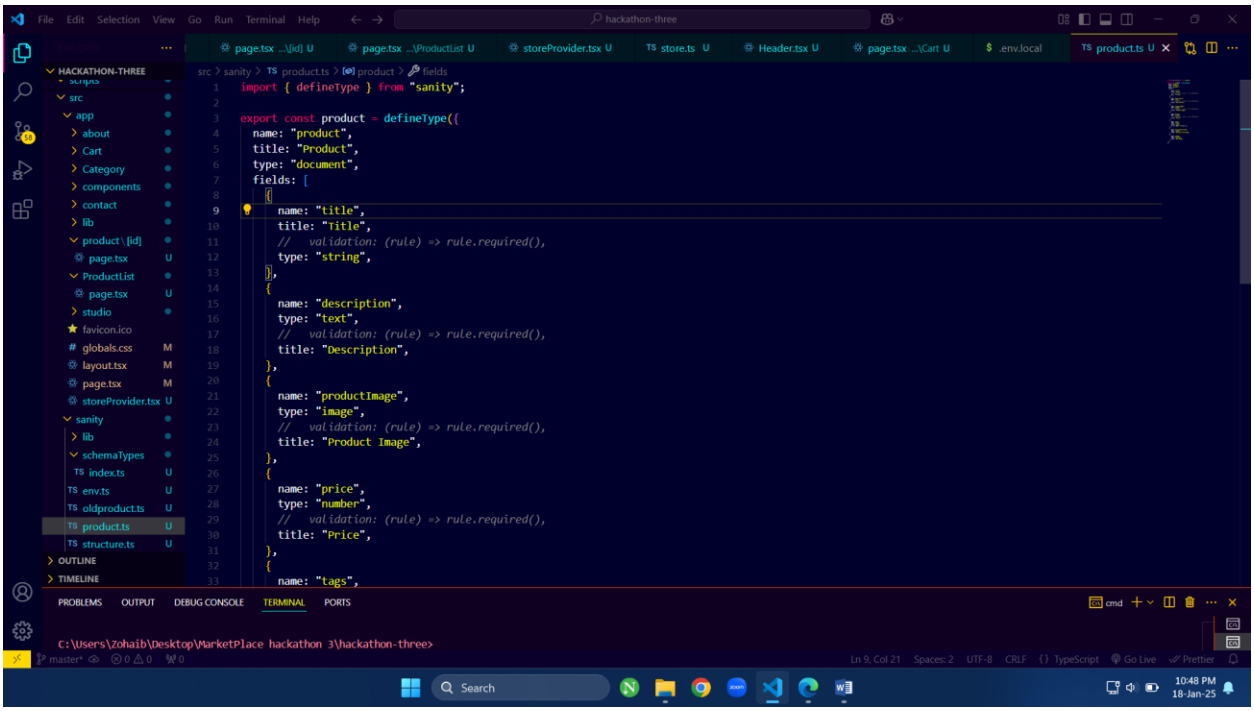
```
import sanityClient from '@sanity/client';

const client = sanityClient({
  projectId: 'your-project-id',
  dataset: 'your-dataset',
  useCdn: true, // `false` if you want to ensure fresh data
  apiVersion: '2023-01-01', // Use the latest ISO date
});

export default client;
```

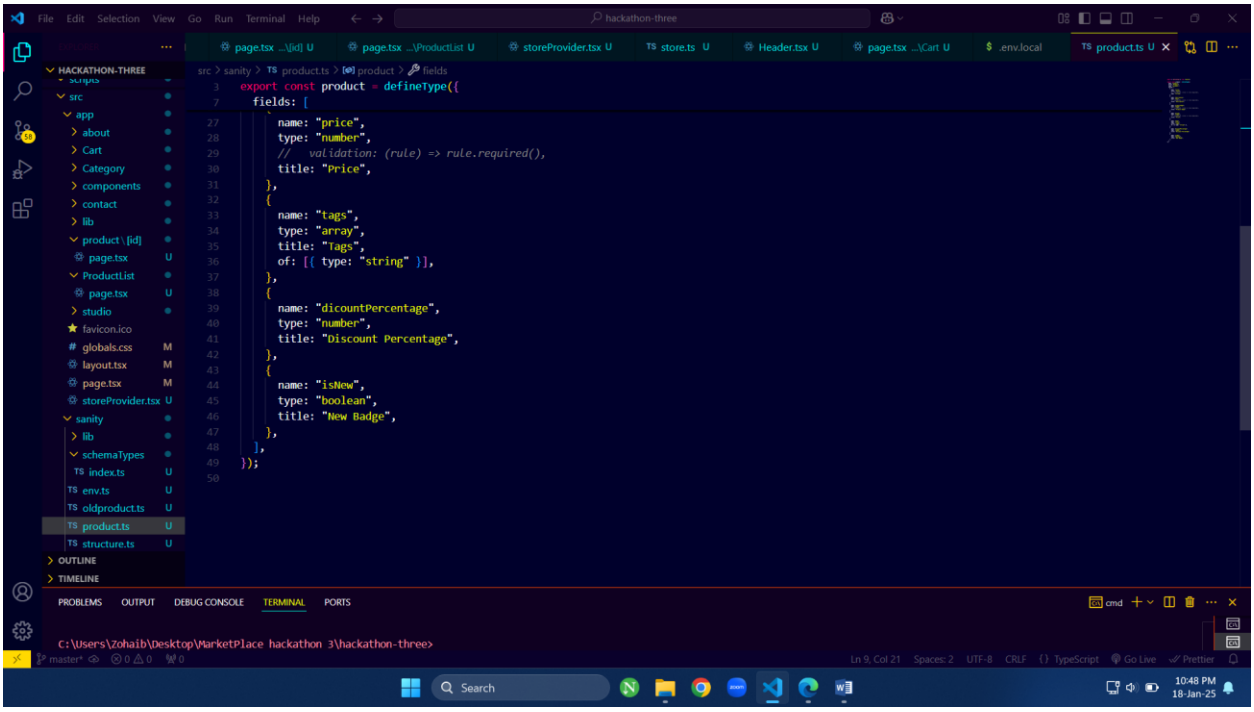
14. **Fetch Data from API:**
    - Use the groq query language to fetch data.
    - Example query:
    - import client from './sanityClient';
    -
    - export async function getData() {
    - const query = `*[_type == 'post']`;
    - const posts = await client.fetch(query);
    - return posts;
    - }
15. **Display Data in Next.js:**
    - Use use effect or usestate for data rendering

---

2. Adjustments Made to Schemas

1. **Define Schemas:**

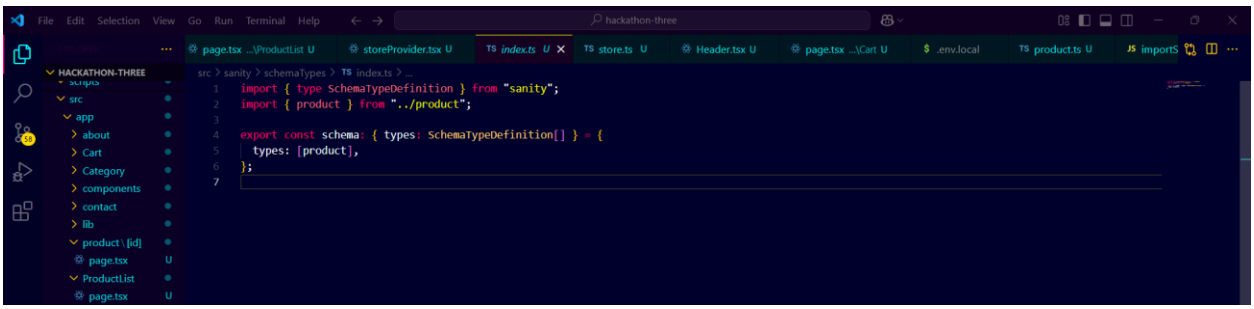




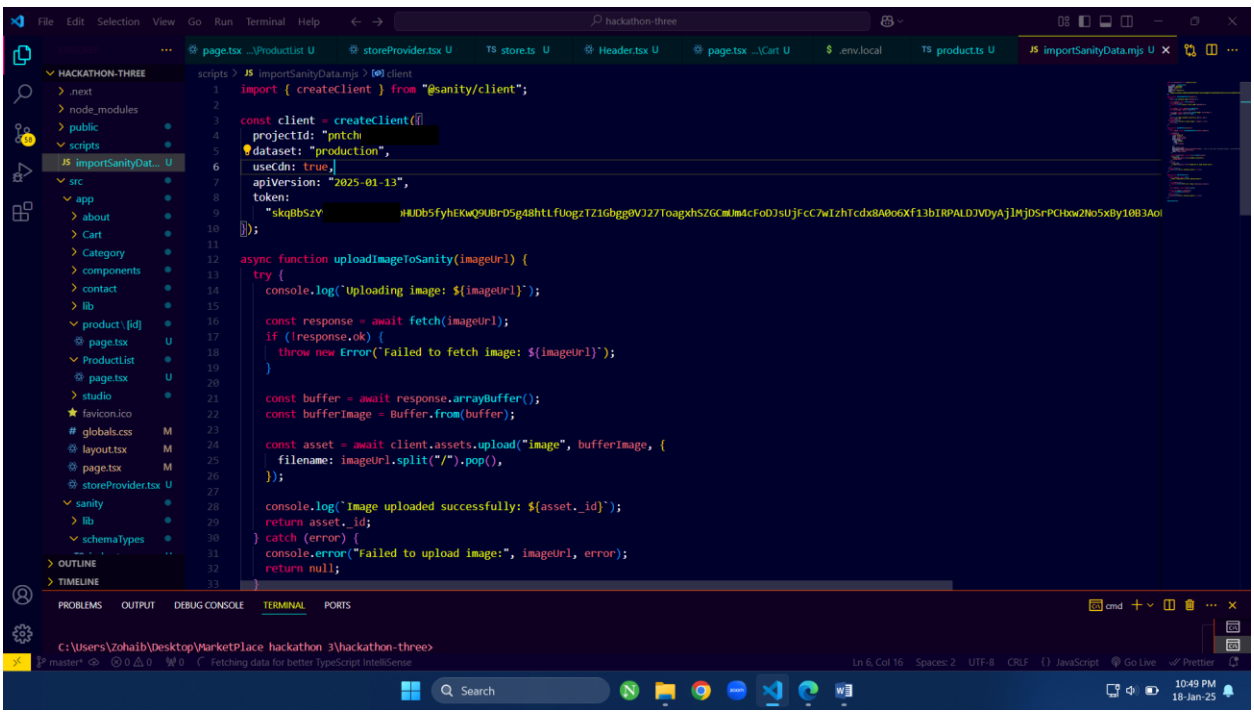
2. **Testing Adjustments:**
   - Test new fields using the Sanity Studio interface.
   - Verify correct data fetching in Next.js.

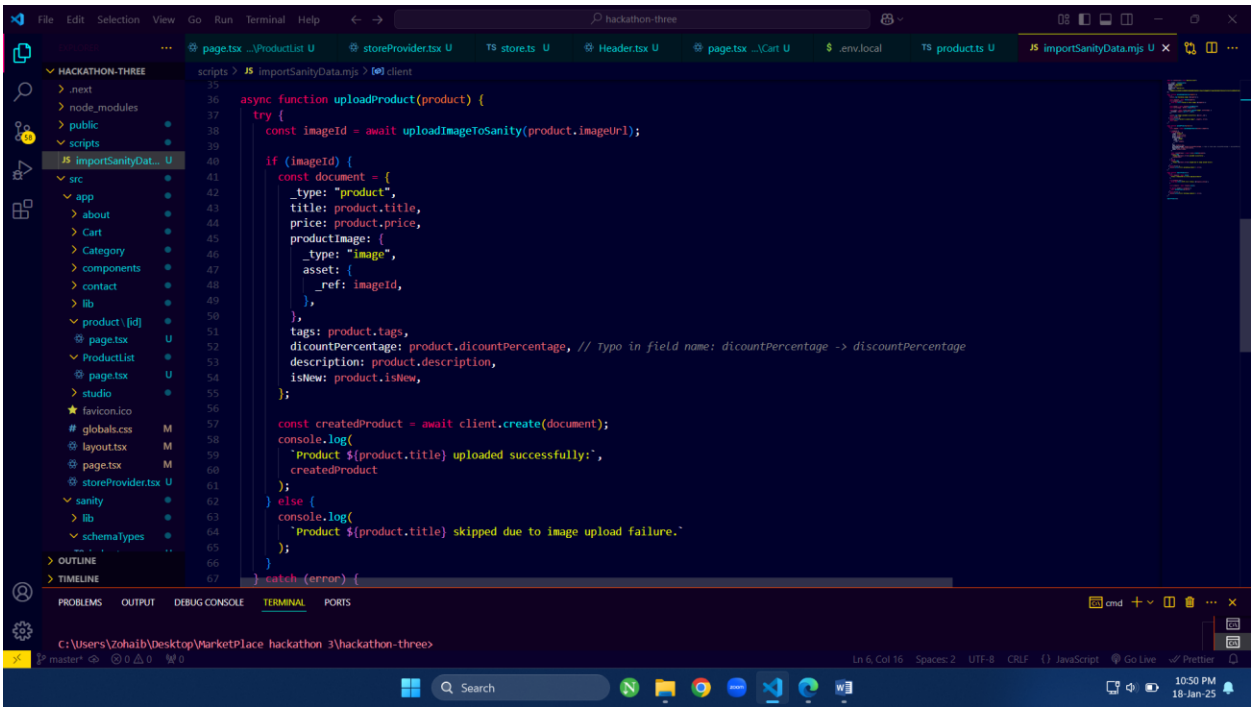
3. Migration Steps and Tools

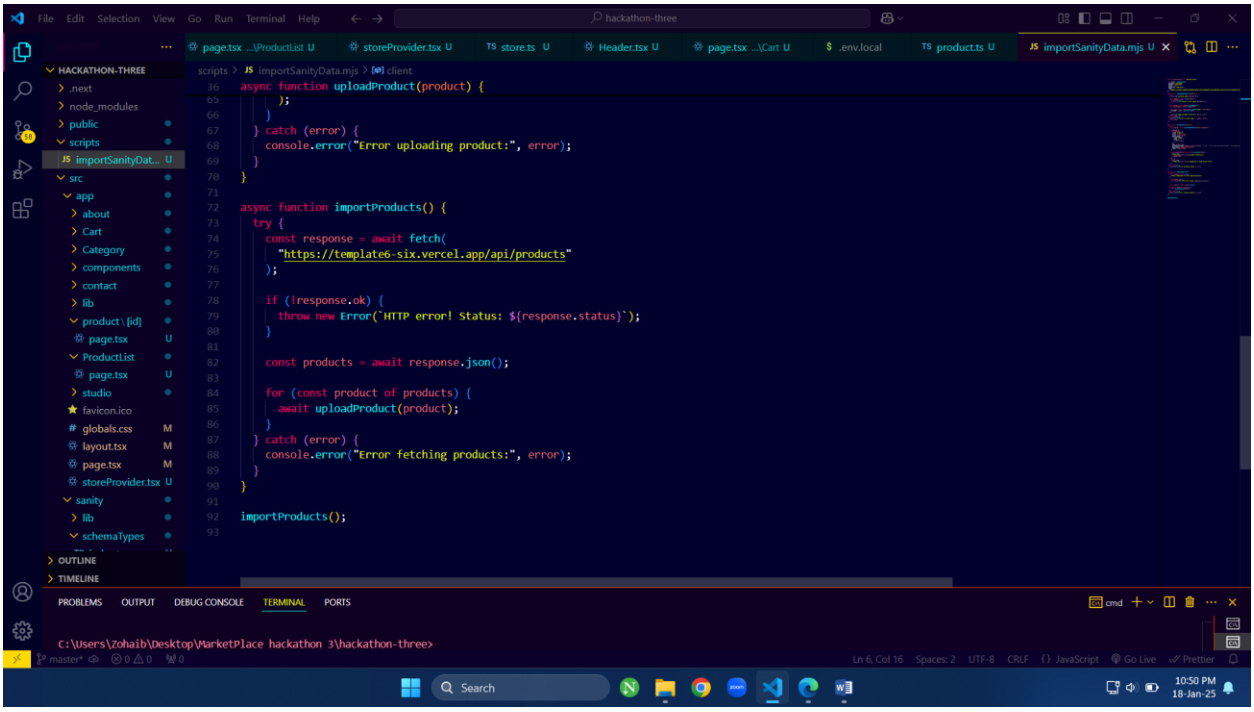
1. **Schema Updates:**
   - Use `sanity schema update` to deploy schema changes.
   - Validate the schema locally before deploying.



2. **Data Migration:**

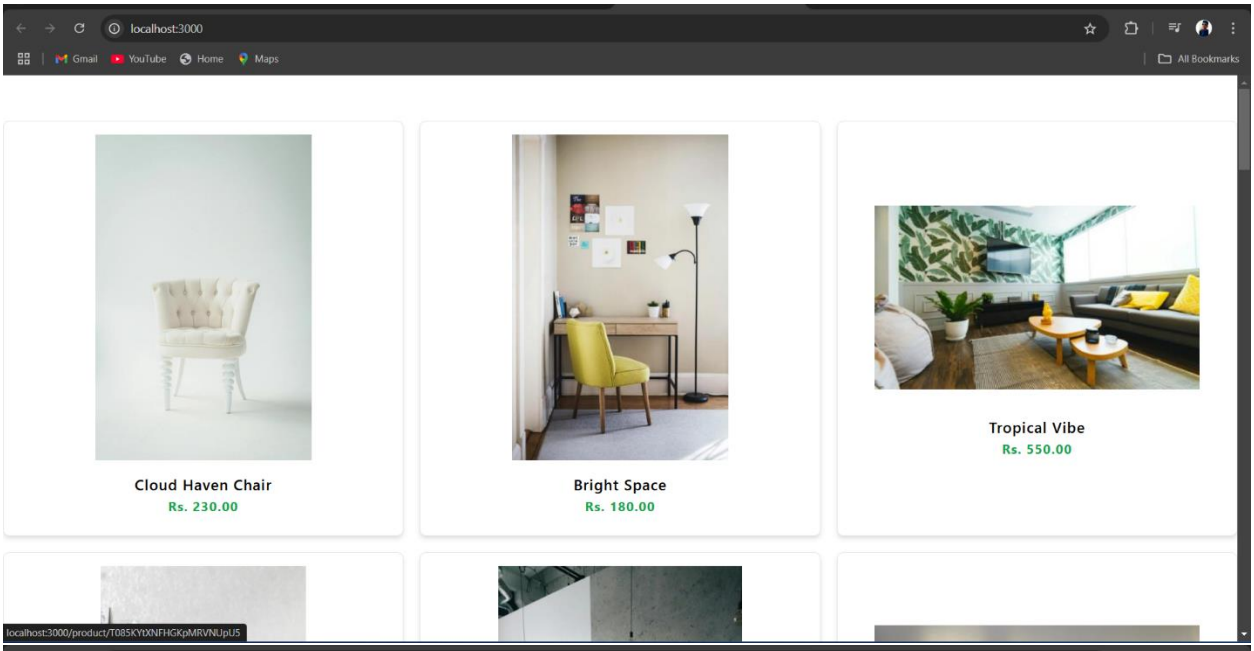


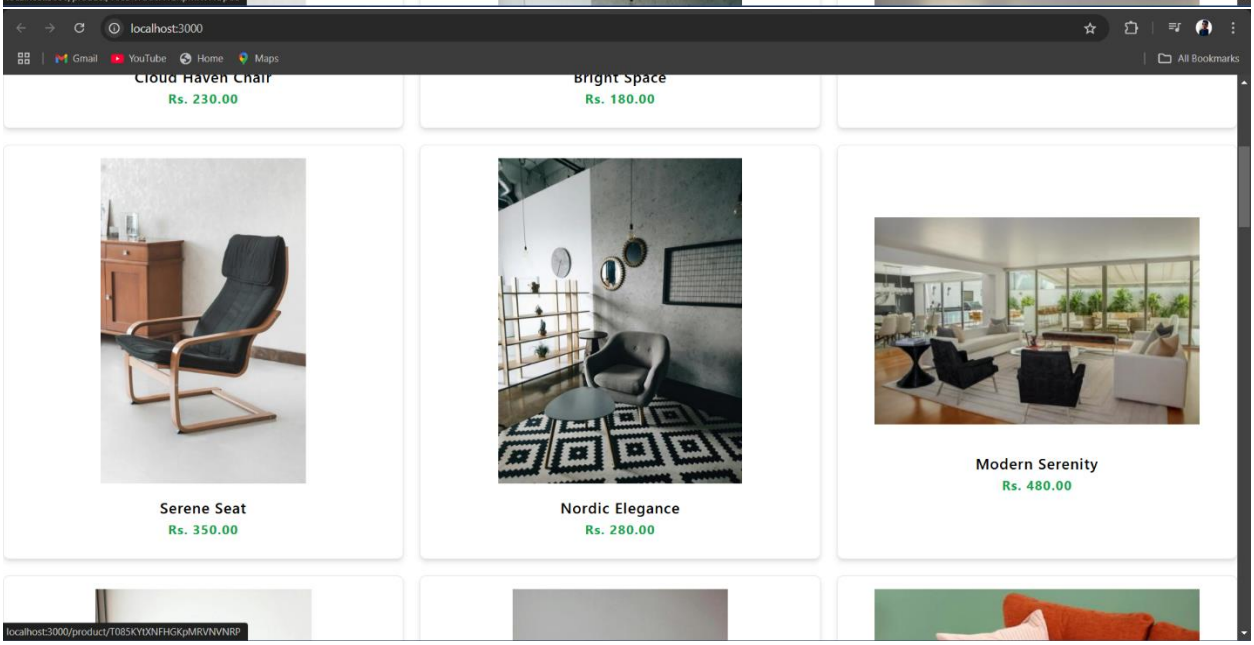




3. **Verify Migration:**
   - Query the updated dataset to ensure data integrity.
4. **Tools Used:**
   - **Postman:** To test API endpoints.

Display Data on frontend





Fronend code:

```
"use client";
import { useState, useEffect } from "react";
import Link from "next/link";
import { client } from "@/sanity/lib/client";
import imageUrlBuilder from "@sanity/image-url";

export default function Home() {
  const [products, setProducts] = useState<any[]>([]);

  const builder = imageUrlBuilder(client);
  const urlFor = (source: any) => builder.image(source);

  useEffect(() => {
    const fetchData = async () => {
      try {
        // Fetching products from Sanity
        const sanityData = await client.fetch(`
          *[_type == "product"]{
            _id,
            title,
            description,
            price,
            discountPercentage,
            isNew,
            tags,
            productImage{
              asset->{
                _id,
                url
              },
              alt
            }
          }
        `);

        // Setting state for products
        setProducts(sanityData);
      } catch (error) {
        console.error("Error fetching data:", error);
      }
    };

    fetchData();
```

```
  }, []); // Empty dependency array means this effect runs only once after the
initial render

  return (
    <main>
      {/* Displaying the products in grid format */}
      <div className="grid grid-cols-1 lg:grid-cols-3 gap-5 my-14 px-5 md:px-
0">
        {products.map((product: any) => (
          <Link href={`/product/${product._id}`} key={product._id}>
            <div className="border p-4 h-full rounded-lg shadow-md flex
justify-center items-center flex-col">
              {product.productImage?.asset?.url && (
                <img
                  src={urlFor(product.productImage.asset)
                    .width(400) // Adjust this size if needed
                    .url()}
                  alt={product.productImage.alt || "Product Image"}
                  className="w-full max-h-[400px] object-scale-down mb-4"
                />
              )}
              <h2 className="font-semibold text-lg text-center">
                {product.title}
              </h2>
              <p className="text-green-600 font-bold mb-2">
                Rs. {product.price.toFixed(2)}
              </p>
              {product.discountPercentage && (
                <p className="text-red-500 text-sm">
                  {product.discountPercentage}% OFF
                </p>
              )}
            </div>
          </Link>
        ))}
      </div>
    </main>
  );
}
```

Migration Script code :

```
import { createClient } from "@sanity/client";

const client = createClient({
  projectId: "my id here",
  dataset: "production",
  useCdn: true,
  apiVersion: "2025-01-13",
  token:
    "my token here",
});

async function uploadImageToSanity(imageUrl) {
  try {
    console.log(`Uploading image: ${imageUrl}`);

    const response = await fetch(imageUrl);
    if (!response.ok) {
      throw new Error(`Failed to fetch image: ${imageUrl}`);
    }

    const buffer = await response.arrayBuffer();
    const bufferImage = Buffer.from(buffer);

    const asset = await client.assets.upload("image", bufferImage, {
      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 uploadProduct(product) {
  try {
    const imageId = await uploadImageToSanity(product.imageUrl);

    if (imageId) {
      const document = {
        _type: "product",
        title: product.title,
```

```
        price: product.price,
        productImage: {
          _type: "image",
          asset: {
            _ref: imageId,
          },
        },
        tags: product.tags,
        dicountPercentage: product.dicountPercentage, // Typo in field name:
dicountPercentage -> discountPercentage
        description: product.description,
        isNew: product.isNew,
      };

      const createdProduct = await client.create(document);
      console.log(
        `Product ${product.title} uploaded successfully:`,
        createdProduct
      );
    } else {
      console.log(
        `Product ${product.title} skipped due to image upload failure.`
      );
    }
  } catch (error) {
    console.error("Error uploading product:", error);
  }
}

async function importProducts() {
  try {
    const response = await fetch(
      "https://template6-six.vercel.app/api/products"
    );

    if (!response.ok) {
      throw new Error(`HTTP error! Status: ${response.status}`);
    }

    const products = await response.json();

    for (const product of products) {
      await uploadProduct(product);
    }
  } catch (error) {
```

```
    console.error("Error fetching products:", error);
  }
}

importProducts();
```

Sanity Schema code :

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

export const product = defineType({
  name: "product",
  title: "Product",
  type: "document",
  fields: [
    {
      name: "title",
      title: "Title",
      //   validation: (rule) => rule.required(),
      type: "string",
    },
    {
      name: "description",
      type: "text",
      //   validation: (rule) => rule.required(),
      title: "Description",
    },
    {
      name: "productImage",
      type: "image",
      //   validation: (rule) => rule.required(),
      title: "Product Image",
    },
    {
      name: "price",
      type: "number",
      //   validation: (rule) => rule.required(),
      title: "Price",
    },
    {
      name: "tags",
```

```
      type: "array",
      title: "Tags",
      of: [{ type: "string" }],
    },
    {
      name: "dicountPercentage",
      type: "number",
      title: "Discount Percentage",
    },
    {
      name: "isNew",
      type: "boolean",
      title: "New Badge",
    },
  ],
});
```