"Day 3 - API Integration Report - FurniAura"

1. API Integration Process

Steps Followed:

1. **API Data Validation:**
   - Imported the data from the provided API into Sanity CMS.
   - Verified that all data fields were accurate and complete.
2. **Testing GROQ Queries:**
   - Used the **Vision tab** in Sanity Studio to test GROQ queries for retrieving necessary data.
   - Ensured all required fields were collected successfully via query execution.
3. **Frontend Integration:**
   - Created a component file named **Item.tsx** inside the **components** folder to display all products.
   - Stored the GROQ query in a variable named **sanityData**.
   - Used **Sanity Client** to fetch data from **sanityData**.
   - Mapped the retrieved data to the UI using a map function tailored to the frontend design.

---

2. Adjustments Made to Schemas

- Modified the schema to include a **slug field** for each product.
  - Enabled **auto slug generation** based on the product's title.
  - This adjustment allowed dynamic routing for individual product pages.

---

3. Migration Steps and Tools Used

Steps Followed:

1. **Custom Script for Data Migration:**
   - Created a folder named **scripts** in the project root.

   - Added a script file named **importData.mjs** inside the folder.

2. **Package Script Setup:**
   - Updated the **package.json** file by adding a custom script:

```
"scripts": {
    "import-data": "node scripts/importData.mjs"
}
```

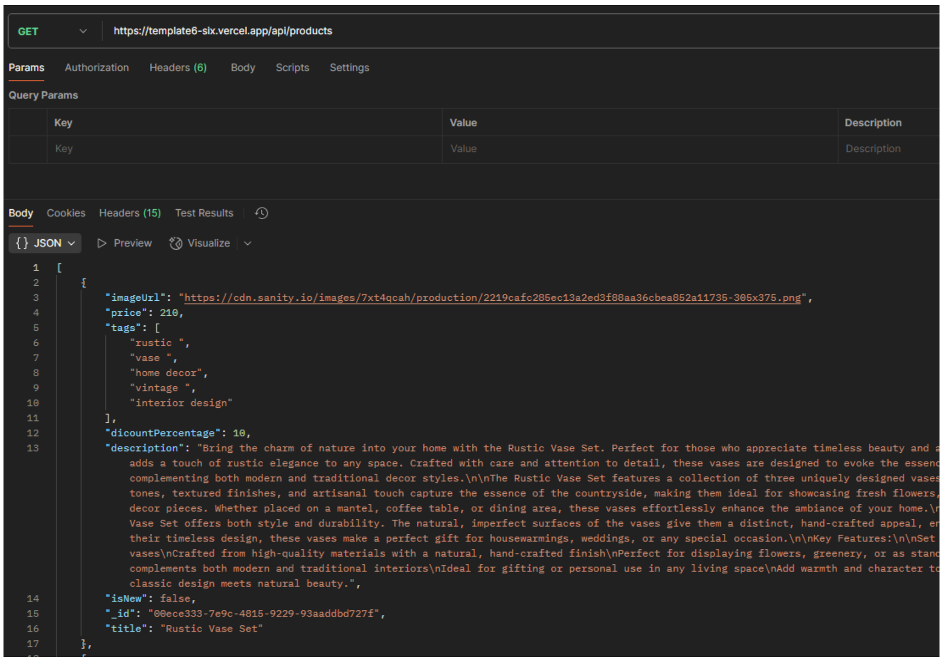
3. **Executing Migration:**
   - Ran the following command in the terminal to execute the migration:

```
npm run import-data
```

   - Successfully uploaded API data to Sanity Studio.

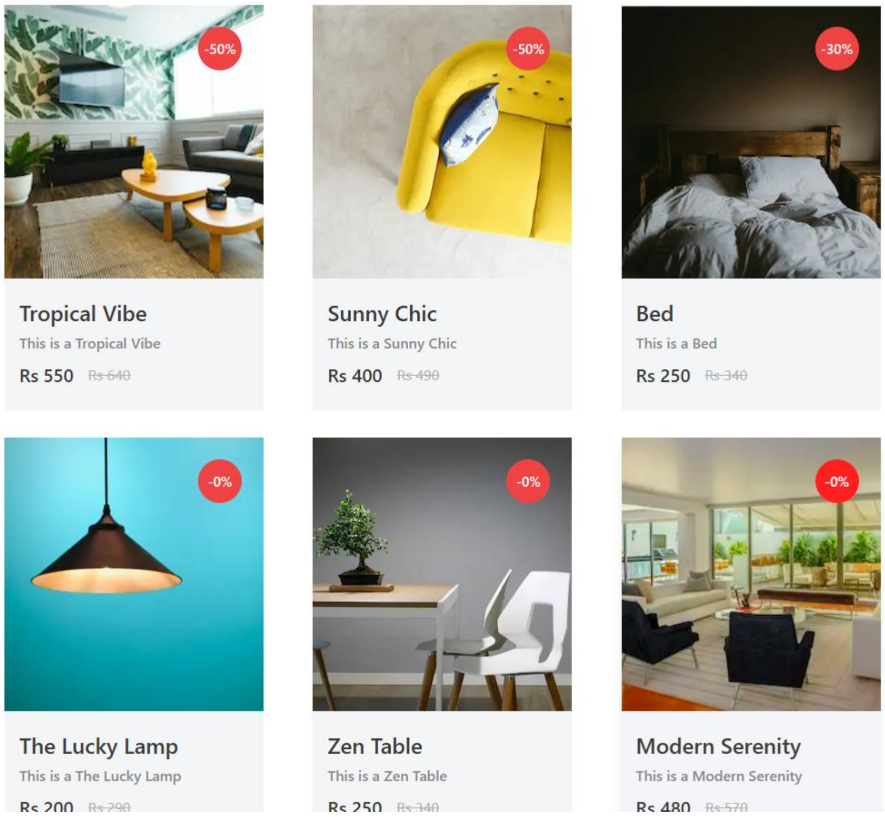
Screenshots:

1. API Calls:



```
[
    {
        "imageUrl": "https://cdn.sanity.io/images/7xt4qcah/production/2219cafc285ec13a2ed3f88aa36cbea852a11735-305x375.png",
        "price": 210,
        "tags": [
            "rustic ",
            "vase ",
            "home decor",
            "vintage ",
            "interior design"
        ],
        "dicountPercentage": 10,
        "description": "Bring the charm of nature into your home with the Rustic Vase Set. Perfect for those who appreciate timeless beauty and a
            adds a touch of rustic elegance to any space. Crafted with care and attention to detail, these vases are designed to evoke the essenc
            complementing both modern and traditional decor styles.\n\nThe Rustic Vase Set features a collection of three uniquely designed vases
            tones, textured finishes, and artisanal touch capture the essence of the countryside, making them ideal for showcasing fresh flowers,
            decor pieces. Whether placed on a mantel, coffee table, or dining area, these vases effortlessly enhance the ambiance of your home.\n
            Vase Set offers both style and durability. The natural, imperfect surfaces of the vases give them a distinct, hand-crafted appeal, en
            their timeless design, these vases make a perfect gift for housewarmings, weddings, or any special occasion.\n\nKey Features:\n\nSet
            vases\nCrafted from high-quality materials with a natural, hand-crafted finish\nPerfect for displaying flowers, greenery, or as stand
            complements both modern and traditional interiors\nIdeal for gifting or personal use in any living space\nAdd warmth and character to
            classic design meets natural beauty.",
        "isNew": false,
        "_id": "00ece333-7e9c-4815-9229-93aaddbd727f",
        "title": "Rustic Vase Set"
    },
```

2. Frontend Display:



Tropical Vibe
This is a Tropical Vibe
Rs 550 ~~Rs 640~~

Sunny Chic
This is a Sunny Chic
Rs 400 ~~Rs 490~~

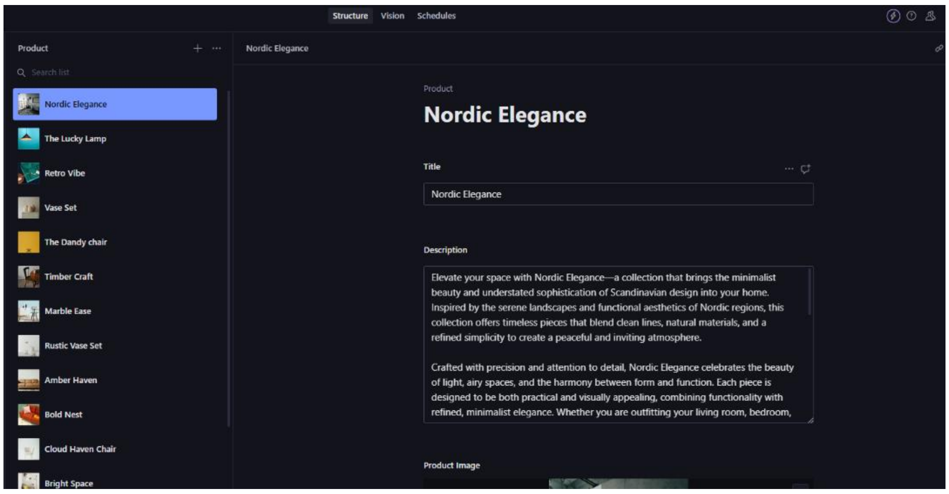
Bed
This is a Bed
Rs 250 ~~Rs 340~~

The Lucky Lamp
This is a The Lucky Lamp
Rs 200 ~~Rs 290~~

Zen Table
This is a Zen Table
Rs 250 ~~Rs 340~~

Modern Serenity
This is a Modern Serenity
Rs 480 ~~Rs 570~~

3. Sanity CMS Fields:



Nordic Elegance

Title: Nordic Elegance

Description:

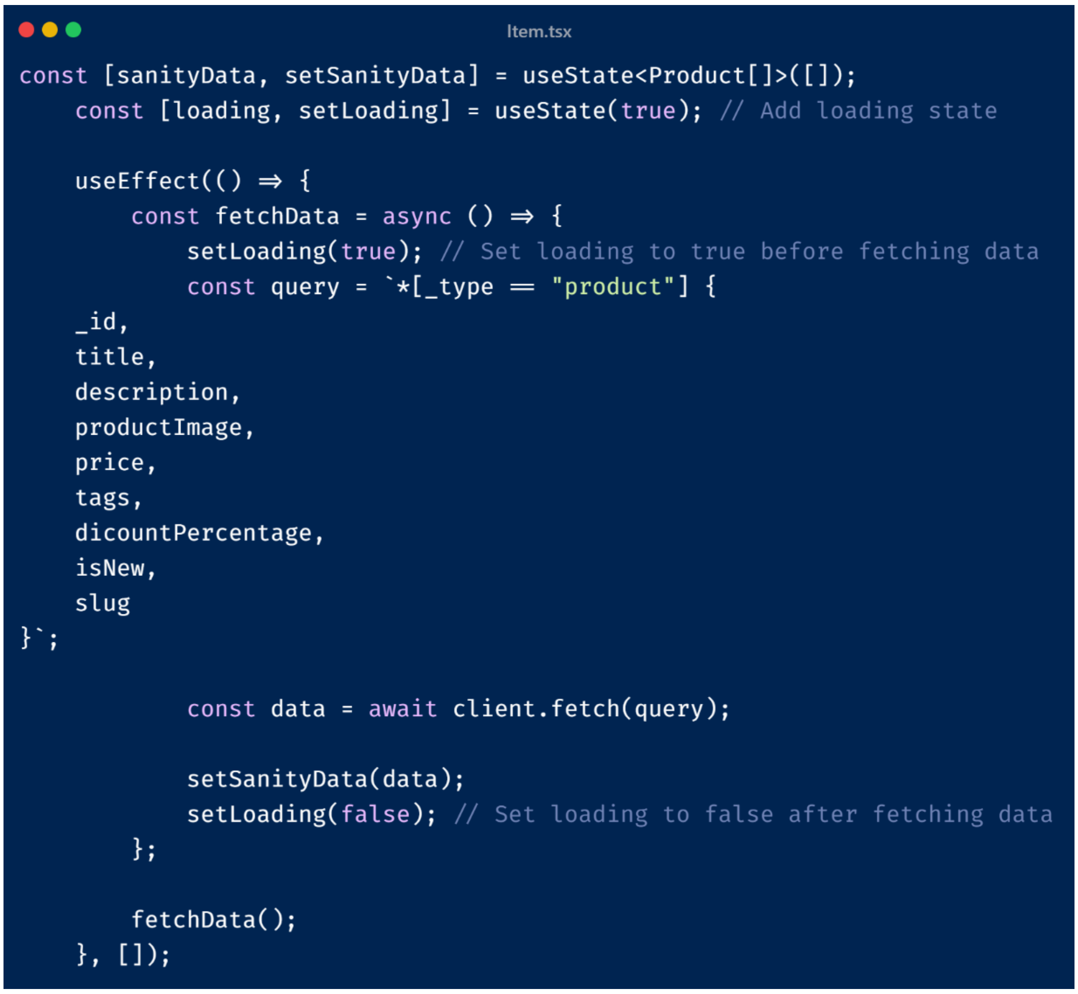
Elevate your space with Nordic Elegance—a collection that brings the minimalist beauty and understated sophistication of Scandinavian design into your home. Inspired by the serene landscapes and functional aesthetics of Nordic regions, this collection offers timeless pieces that blend clean lines, natural materials, and a refined simplicity to create a peaceful and inviting atmosphere.

Crafted with precision and attention to detail, Nordic Elegance celebrates the beauty of light, airy spaces, and the harmony between form and function. Each piece is designed to be both practical and visually appealing, combining functionality with refined, minimalist elegance. Whether you are outfitting your living room, bedroom,

Product Image

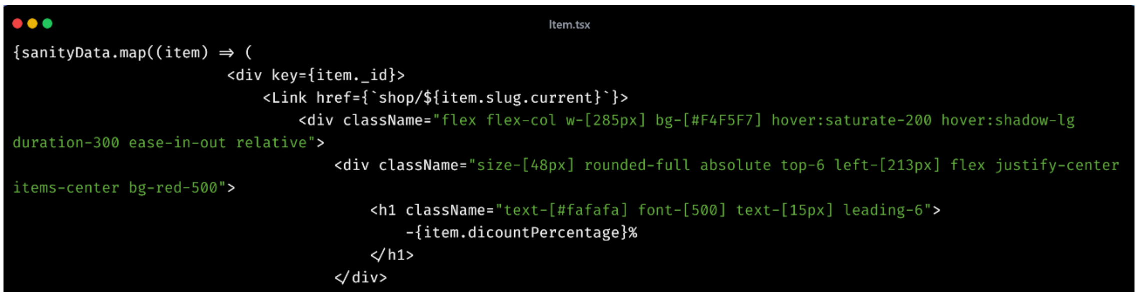
Code Snippets:

1. GROQ Query in Item.tsx:

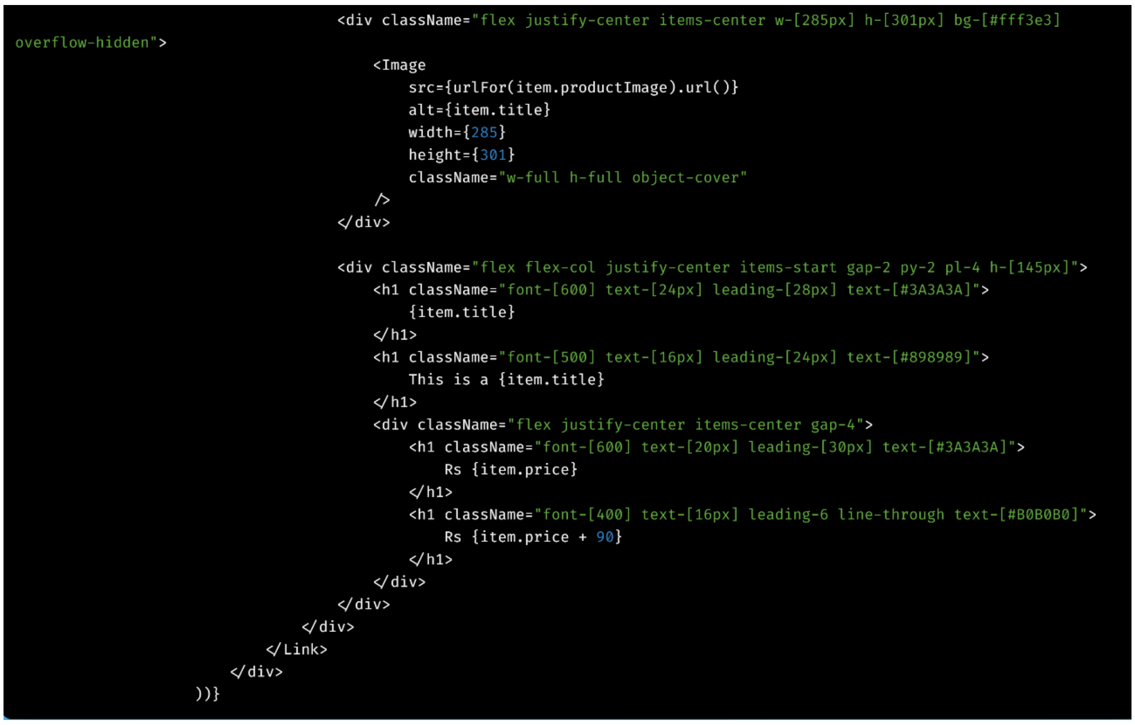


```
const [sanityData, setSanityData] = useState<Product[]>([]);
    const [loading, setLoading] = useState(true); // Add loading state

    useEffect(() => {
        const fetchData = async () => {
            setLoading(true); // Set loading to true before fetching data
            const query = `*[_type == "product"] {
    _id,
    title,
    description,
    productImage,
    price,
    tags,
    dicountPercentage,
    isNew,
    slug
}`;

            const data = await client.fetch(query);

            setSanityData(data);
            setLoading(false); // Set loading to false after fetching data
        };

        fetchData();
    }, []);
```

2. Mapping Data in Frontend:

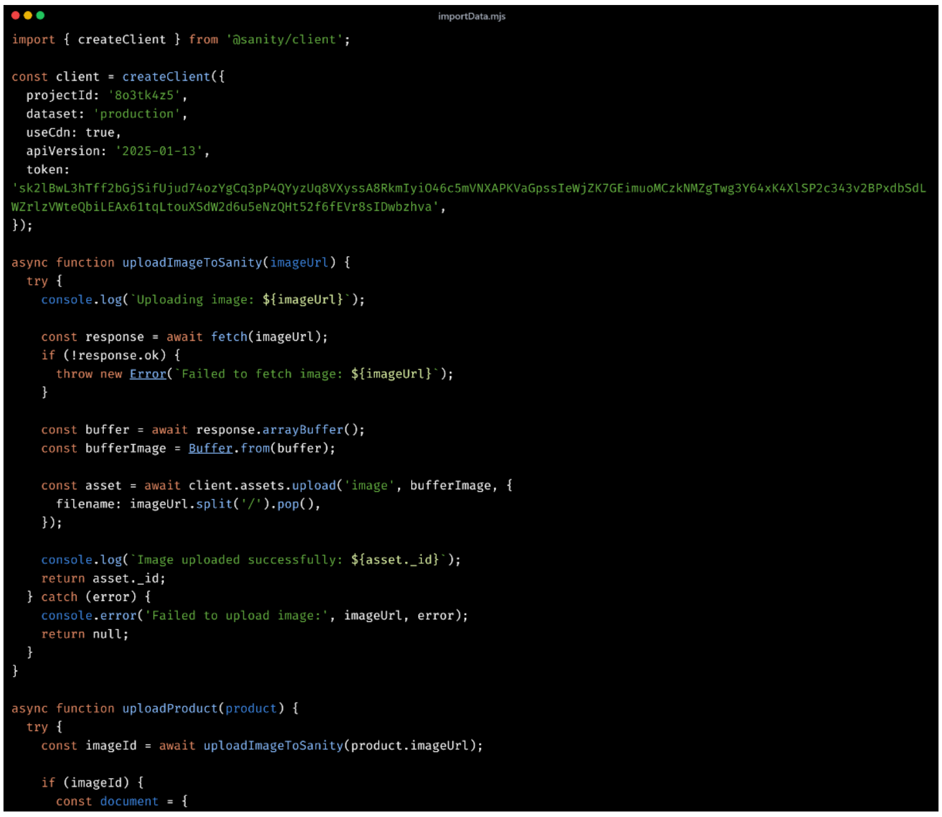


```
{sanityData.map((item) => (
                        <div key={item._id}>
                            <Link href={`shop/${item.slug.current}`}>
                                <div className="flex flex-col w-[285px] bg-[#F4F5F7] hover:saturate-200 hover:shadow-lg
duration-300 ease-in-out relative">
                                    <div className="size-[48px] rounded-full absolute top-6 left-[213px] flex justify-center
items-center bg-red-500">
                                        <h1 className="text-[#fafafa] font-[500] text-[15px] leading-6">
                                            -{item.dicountPercentage}%
                                        </h1>
                                    </div>
```



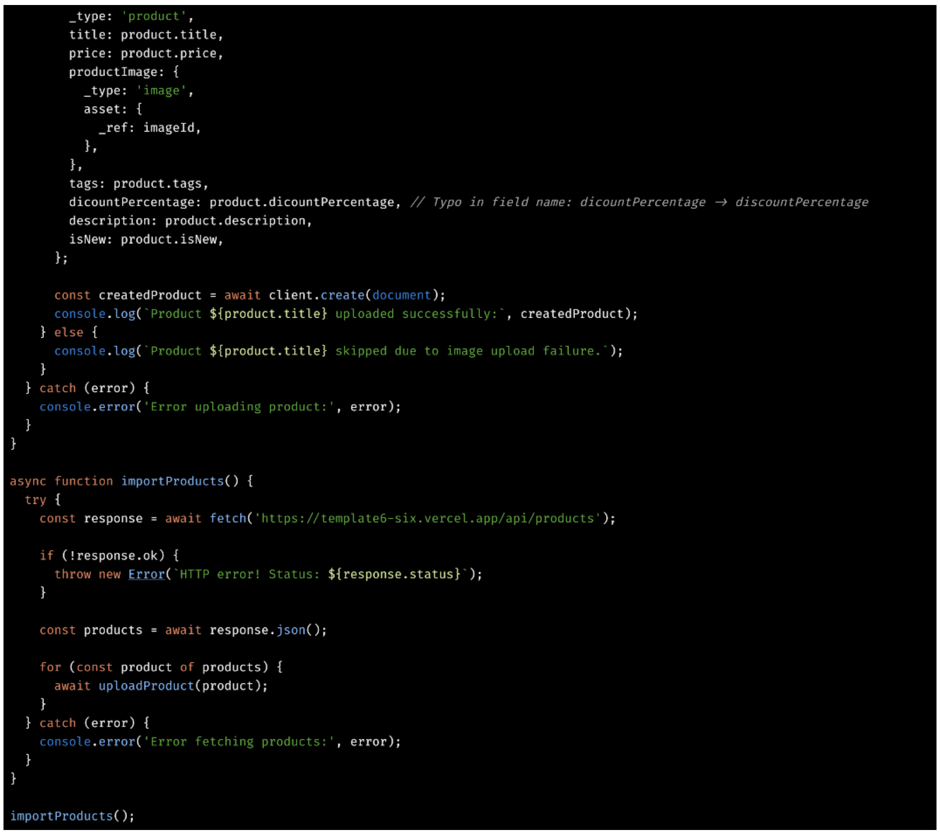
```
                                    <div className="flex justify-center items-center w-[285px] h-[301px] bg-[#fff3e3]
overflow-hidden">
                                        <Image
                                            src={urlFor(item.productImage).url()}
                                            alt={item.title}
                                            width={285}
                                            height={301}
                                            className="w-full h-full object-cover"
                                        />
                                    </div>

                                    <div className="flex flex-col justify-center items-start gap-2 py-2 pl-4 h-[145px]">
                                        <h1 className="font-[600] text-[24px] leading-[28px] text-[#3A3A3A]">
                                            {item.title}
                                        </h1>
                                        <h1 className="font-[500] text-[16px] leading-[24px] text-[#898989]">
                                            This is a {item.title}
                                        </h1>
                                        <div className="flex justify-center items-center gap-4">
                                            <h1 className="font-[600] text-[20px] leading-[30px] text-[#3A3A3A]">
                                                Rs {item.price}
                                            </h1>
                                            <h1 className="font-[400] text-[16px] leading-6 line-through text-[#B0B0B0]">
                                                Rs {item.price + 90}
                                            </h1>
                                        </div>
                                    </div>
                                </div>
                            </Link>
                        </div>
                    ))}
```

3. Custom Migration Script in importData.mjs:



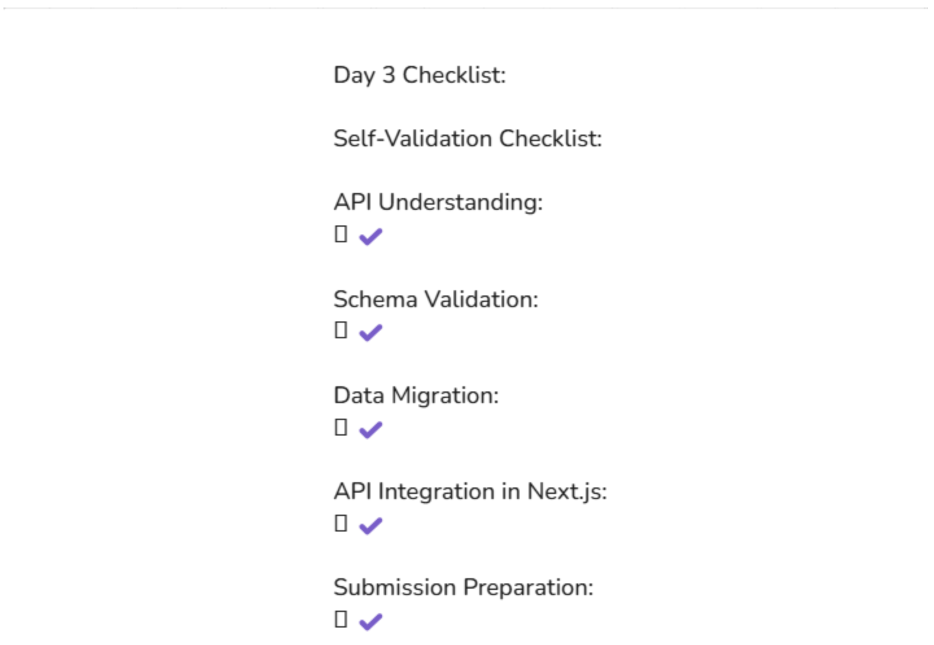
importData.mjs

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

const client = createClient({
  projectId: '8o3tk4z5',
  dataset: 'production',
  useCdn: true,
  apiVersion: '2025-01-13',
  token:
'sk2lBwL3hTff2bGjSifUjud74ozYgCq3pP4QYyzUq8VXyssA8RkmIyiO46c5mVNXAPKVaGpssIeWjZK7GEimuoMCzkNMZgTwg3Y64xK4XlSP2c343v2BPxdbSdL
WZrlzVWteQbiLEAx61tqLtouXSdW2d6u5eNzQHt52f6fEVr8sIDwbzhva',
});

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();
```

4. Self-Validation Checklist:



Day 3 Checklist:

Self-Validation Checklist:

API Understanding:
□ ✔

Schema Validation:
□ ✔

Data Migration:
□ ✔

API Integration in Next.js:
□ ✔

Submission Preparation:
□ ✔