Day 3 - API Integration Report – AuraBox

Documentation Author: Umar Ali

Slot: Tuesday(2 to 5)

Task given by: Sir Ameen Alam

Teachers: Sir Ali Aftab Sheikh & Sir Fahad Khan

API INTEGRATION PROCESS:

- **Identifying the API endpoints:** The external API that provides product data was identified, lecluding endpoints for product information, images, and categories.
- **Creating API fetch functions:** We created a fetch function using “fetch()” to retrieve product data from the external API.
- **Integrating API calls in project:** API calls were made during the data fetching process in NextJS. The API was connected to the frontend and used in pages like Product Listning.

API CALLS SCREEN SHOT:

.

```
const ProductCards: React.FC = () => {
    const [products, setProducts] = useState<Product[]>([]);

    const fetchProducts = async () => {
        try {
            const query = `
            *[_type == "product"]{
            _id,
            title,
            price,
            description,
            "imageUrl": productImage.assest-> url,
            tags
            }
            `;
            const data = await sanity.fetch(query);
            setProducts(data);
        } catch (error) {
            console.error("Error While Fetching Products", error);
        }
    };
```

FRONTEND DISPLAY OF API DATA:

Products From API Data

Rustic Vase Set

Bring the charm of nature into your home with the Rustic Vase Set. Perfect for those who appreciate ...

210

rustic vase home decor vintage interior design

Add To Cart

The Dandy chair

Meet The Dandy Chair—the epitome of comfort, style, and sophistication. Designed for those who appre...

150

chair elegant vintage classic furniture

Add To Cart

Bed

Introducing the Bed—your sanctuary for rest and relaxation, designed with both comfort and style in ...

250

bed furniture sleep cozy modern

Add To Cart

Retro Vibe

Introducing RetroVibe—a perfect blend of vintage charm and modern style, designed for those who appr...

340

retro vintage furniture modern decor

Add To Cart

HANDLING DATA:

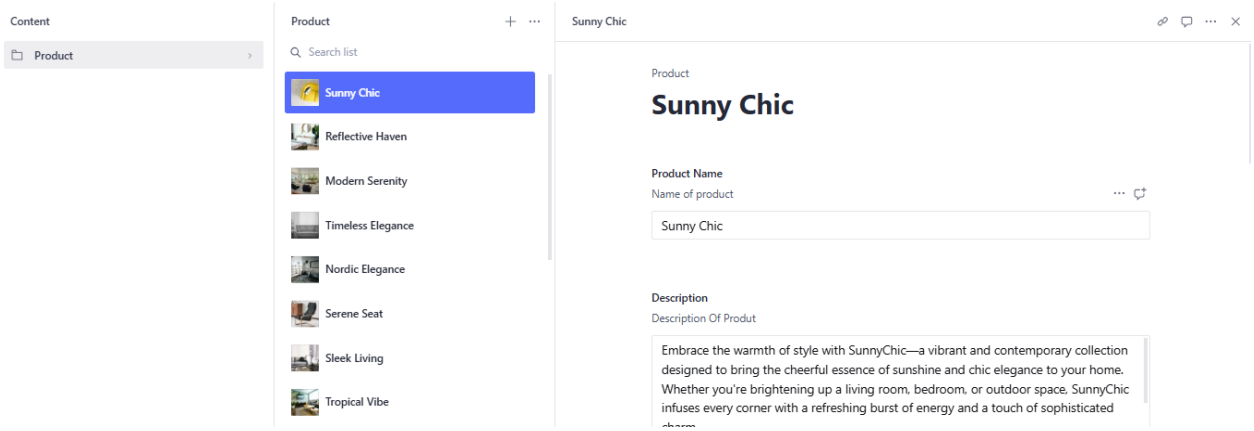
Upon successful API calls, product data was shown in front-end. The product details were rendered on the product Listning Page.

ERROR HANDLING:

A fallback mechanism was implemented to handle any error during the API fetch process.

ADJUSTMENT MADE TO SCHEMAS:

New fields were added to the product schema yo handle API data (e.g. discountPercentage, tags etc).



DATA MIGRATION:

The product data was migrated from the given API into sanityCMS by following the steps:

- **Preparing the migration script:** A nodeJS script was given to automate the migration process.
- **Script to migrate product data:** The script literated over the product data and pushed it to Sanity's CMS by following code:

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

const client = createClient({
  projectId: 'itewl73m',
  dataset: 'production',
  useCdn: true,
  apiVersion: '2025-01-13',
  token: 'skSJ1EpcMUGHSkniHL1QdnG0PIdkiyMk3J1kxpf6UGzL16XNa6kU2juUOgX1N4Hmj
});

Pieces: Comment | Pieces: Explain
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;
  }
}
```

```
Pieces: Comment | Pieces: Explain
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: dicountPercenta
        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);
  }
}
```

```
Pieces: Comment | Pieces: Explain
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 STUDIO:

Sanity studio was used to verify that the data was correctly migrated and populated. We can check the product documents and their fields to ensure that all relevant information was restored.

VERIFICATION:

After migration, the data was verified in the Sanity Studio interface, ensuring that all product data, including pricing, description and categories were correctly added.