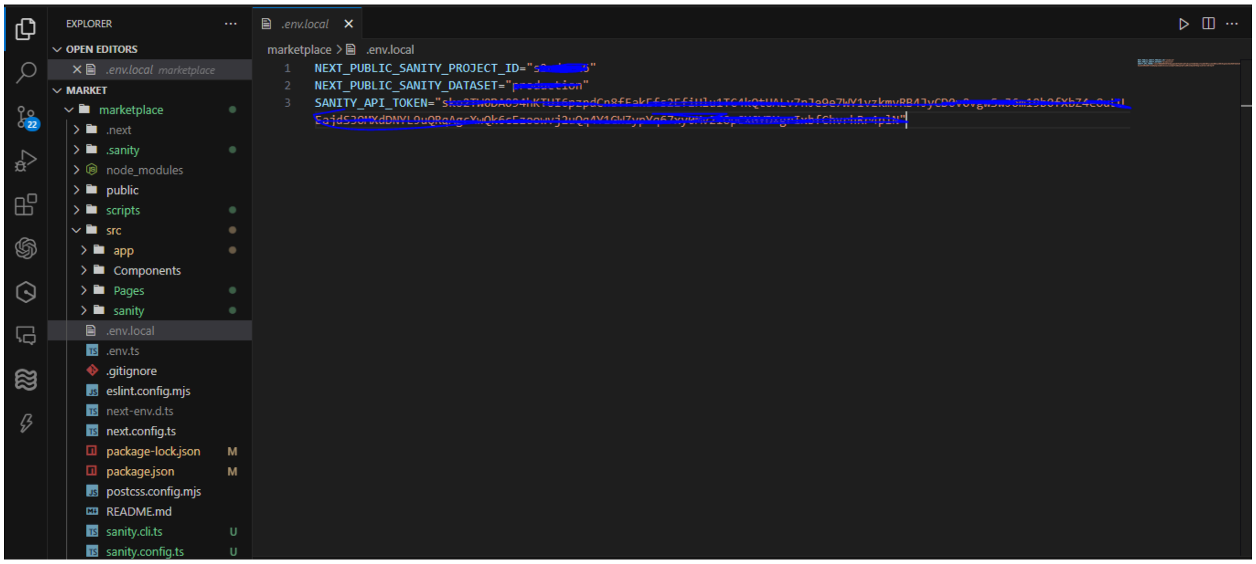
Day 3 - API Integration Report - Flex Wear Marketplace

1. API Integration Process

Overview:
The API integration process was a critical step in enabling the connection between the frontend and Sanity CMS, where we fetch and display data from the Sanity API.

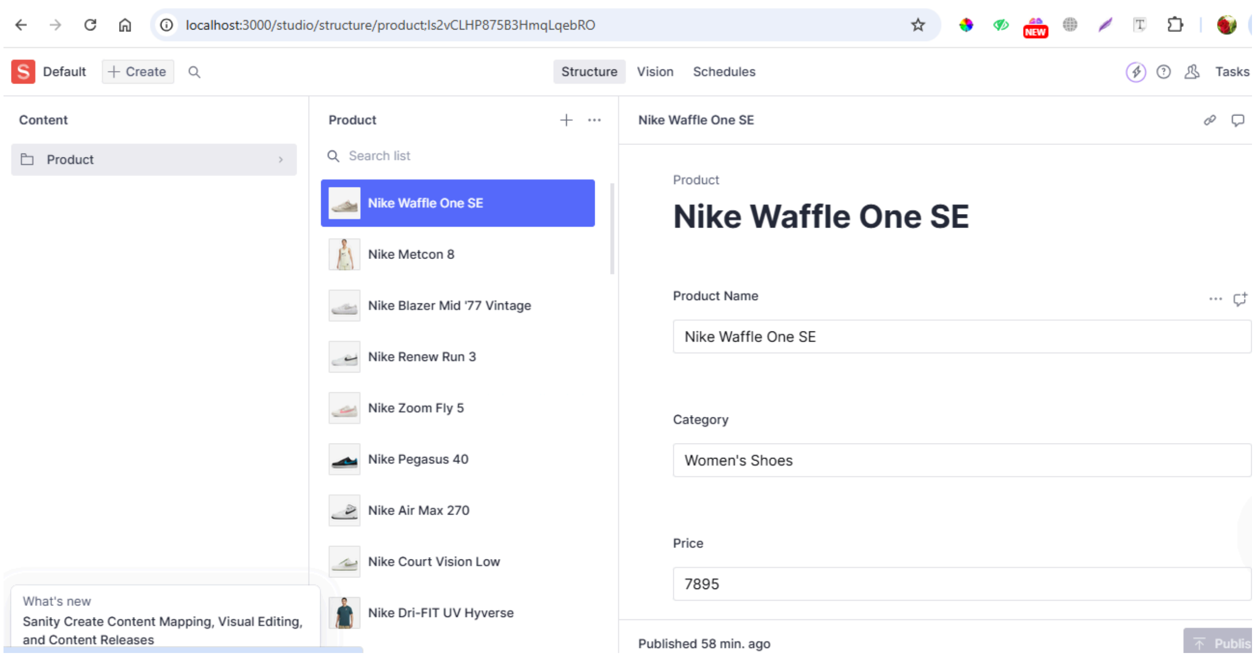
API Setup:
The first step was setting up the API connection by configuring environment variables (API version, dataset, and project ID) in the `.env.local` file. These values were pulled from the Sanity project dashboard.



```
NEXT_PUBLIC_SANITY_PROJECT_ID="[redacted]"
NEXT_PUBLIC_SANITY_DATASET="[redacted]"
SANITY_API_TOKEN="[redacted]"
```

1. **Fetching Data from Sanity**:
   We created an API handler to fetch product data from the Sanity dataset using the Groq query.

Prepared by:
Ayesha Nasir



localhost:3000/studio/structure/product;ls2vCLHP875B3HmqLqebRO

Product

Nike Waffle One SE

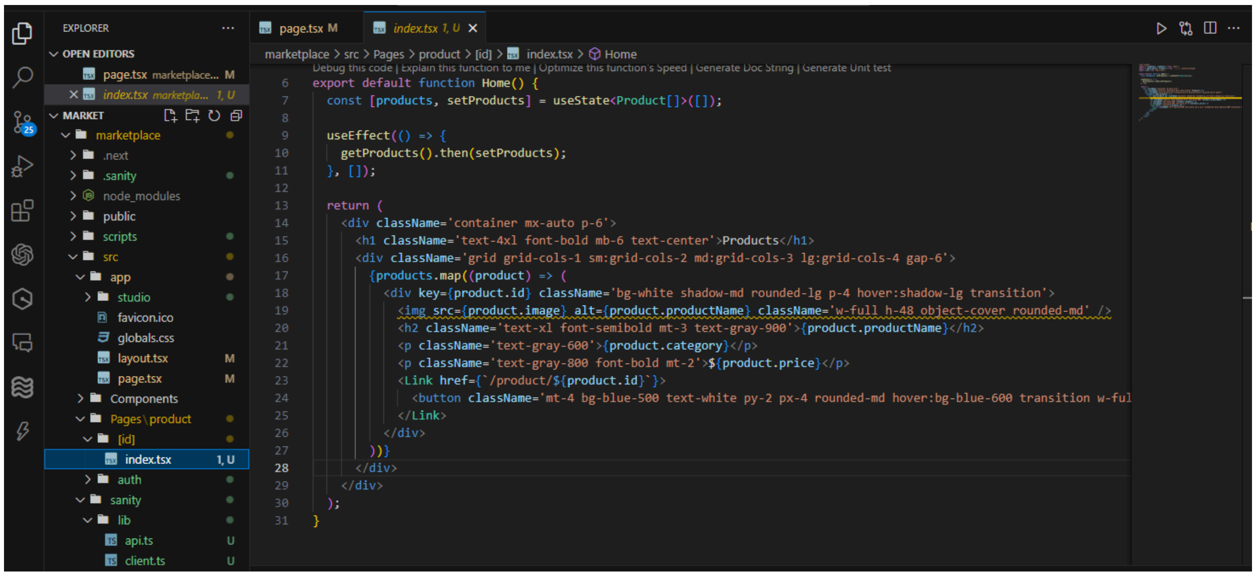
Product Name: Nike Waffle One SE

Category: Women's Shoes

Price: 7895

Published 58 min. ago

2. **Handling Data in the Frontend**:
 We integrated the API call into the React component, which uses Next.js's `getServerSideProps` to fetch data at build time.



marketplace > src > Pages > product > [id] > index.tsx > Home

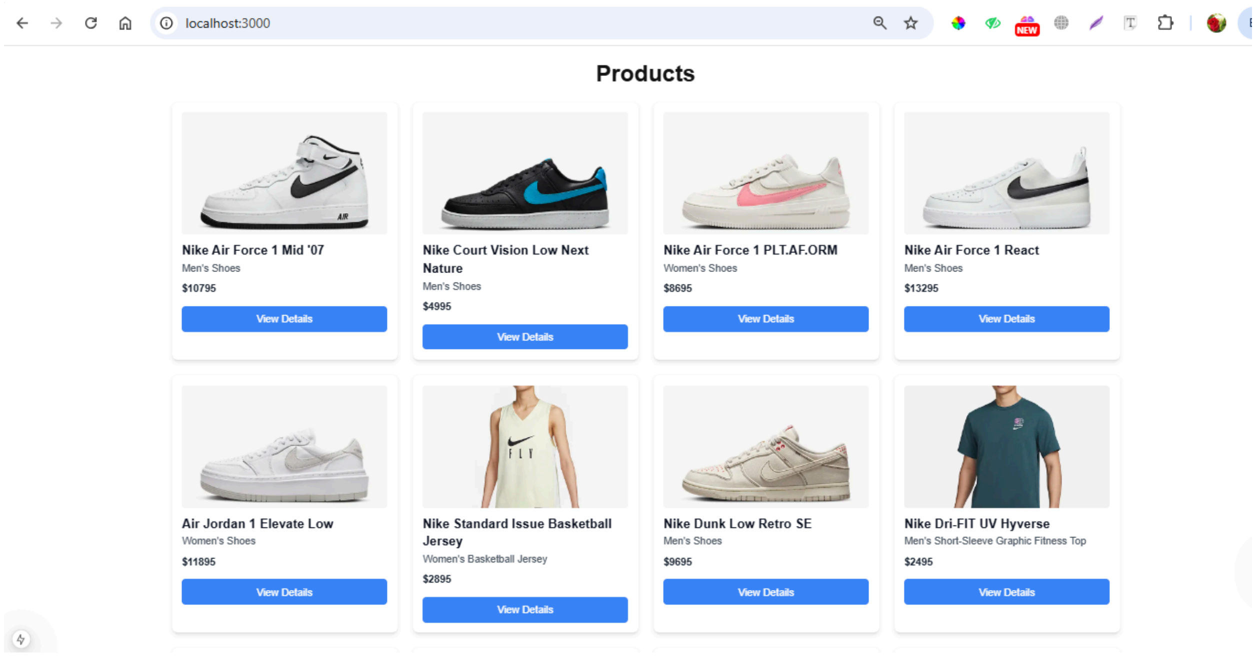
```
export default function Home() {
  const [products, setProducts] = useState<Product[]>([]);

  useEffect(() => {
    getProducts().then(setProducts);
  }, []);

  return (
    <div className='container mx-auto p-6'>
      <h1 className='text-4xl font-bold mb-6 text-center'>Products</h1>
      <div className='grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-6'>
        {products.map((product) => (
          <div key={product.id} className='bg-white shadow-md rounded-lg p-4 hover:shadow-lg transition'>
            <img src={product.image} alt={product.productName} className='w-full h-48 object-cover rounded-md' />
            <h2 className='text-xl font-semibold mt-3 text-gray-900'>{product.productName}</h2>
            <p className='text-gray-600'>{product.category}</p>
            <p className='text-gray-800 font-bold mt-2'>${product.price}</p>
            <Link href={`/product/${product.id}`}>
              <button className='mt-4 bg-blue-500 text-white py-2 px-4 rounded-md hover:bg-blue-600 transition w-ful
            </Link>
          </div>
        ))}
      </div>
    </div>
  );
}
```

Prepared by:
Ayesha Nasir

3. **Displaying Data on the Frontend**:
   The data fetched from the API was mapped into the frontend component to display products with their relevant details.



Products

Nike Air Force 1 Mid '07
Men's Shoes
$10795
View Details

Nike Court Vision Low Next Nature
Men's Shoes
$4995
View Details

Nike Air Force 1 PLT.AF.ORM
Women's Shoes
$8695
View Details

Nike Air Force 1 React
Men's Shoes
$13295
View Details

Air Jordan 1 Elevate Low
Women's Shoes
$11895
View Details

Nike Standard Issue Basketball Jersey
Women's Basketball Jersey
$2895
View Details

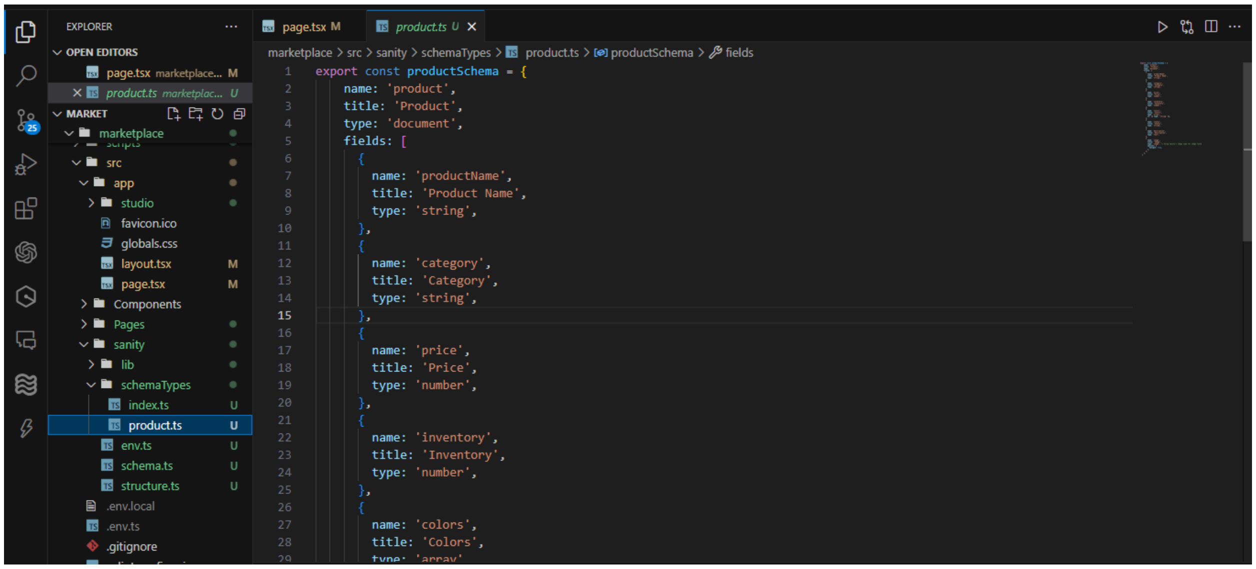
Nike Dunk Low Retro SE
Men's Shoes
$9695
View Details

Nike Dri-FIT UV Hyverse
Men's Short-Sleeve Graphic Fitness Top
$2495
View Details

2. Adjustments Made to Schemas

Schema Updates:
 The product schema was defined with various fields to represent a product in Sanity CMS. Adjustments were made to ensure that each product has the necessary fields (e.g., name, price, inventory, image, description) with appropriate types.

Prepared by:
Ayesha Nasir



```
export const productSchema = {
    name: 'product',
    title: 'Product',
    type: 'document',
    fields: [
      {
        name: 'productName',
        title: 'Product Name',
        type: 'string',
      },
      {
        name: 'category',
        title: 'Category',
        type: 'string',
      },
      {
        name: 'price',
        title: 'Price',
        type: 'number',
      },
      {
        name: 'inventory',
        title: 'Inventory',
        type: 'number',
      },
      {
        name: 'colors',
        title: 'Colors',
        type: 'array'
```

3. Migration Steps and Tools Used

Migration Process:
The migration involved migrating the product data into Sanity from a local source. The migration process included:

- Using Sanity CLI tools for importing data: `sanity import` for bulk data upload.
- Writing a custom script to format the data in a way that could be successfully imported into the product schema.

Tools Used:

- **Sanity CLI**: To import data and manage schemas.
- **CSV/JSON files**: Data was imported from these files after being pre-processed.
- **Custom Scripts**: For transforming and validating data before the import.

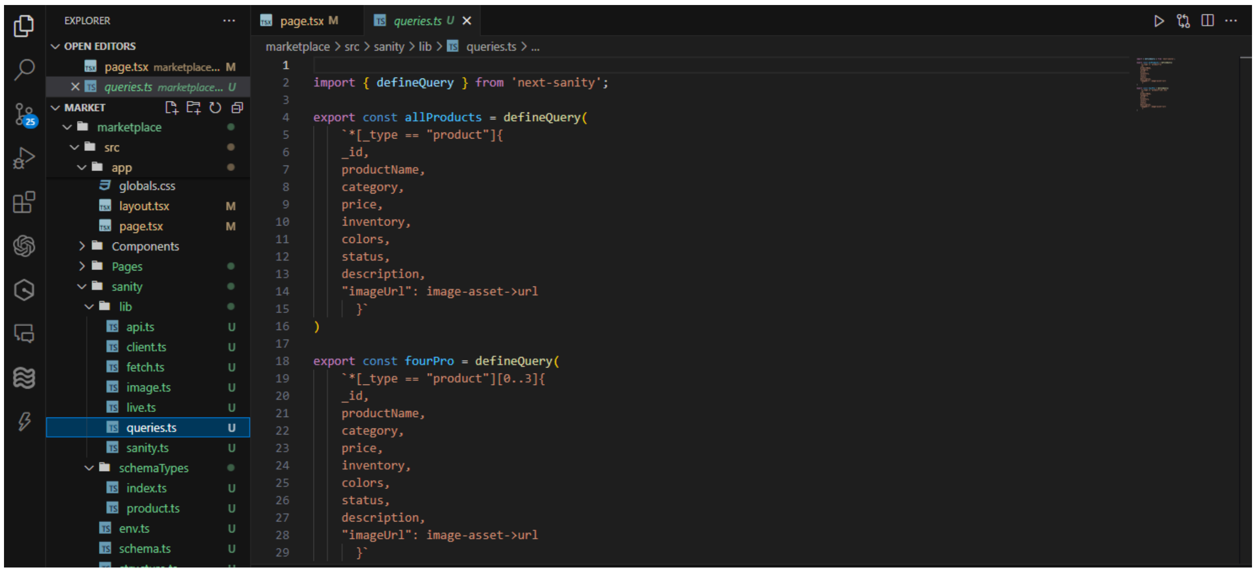
```
sanity import ./data/products.json --dataset production
```

Validation:
To ensure that the data was correctly migrated, we validated the data in the Sanity Studio after the import, checking for any discrepancies or missing values. Logs were kept for any discrepancies for further investigation.

Prepared by:
Ayesha Nasir

5. Code Snippets for API Integration and Migration Scripts

API Integration Code:

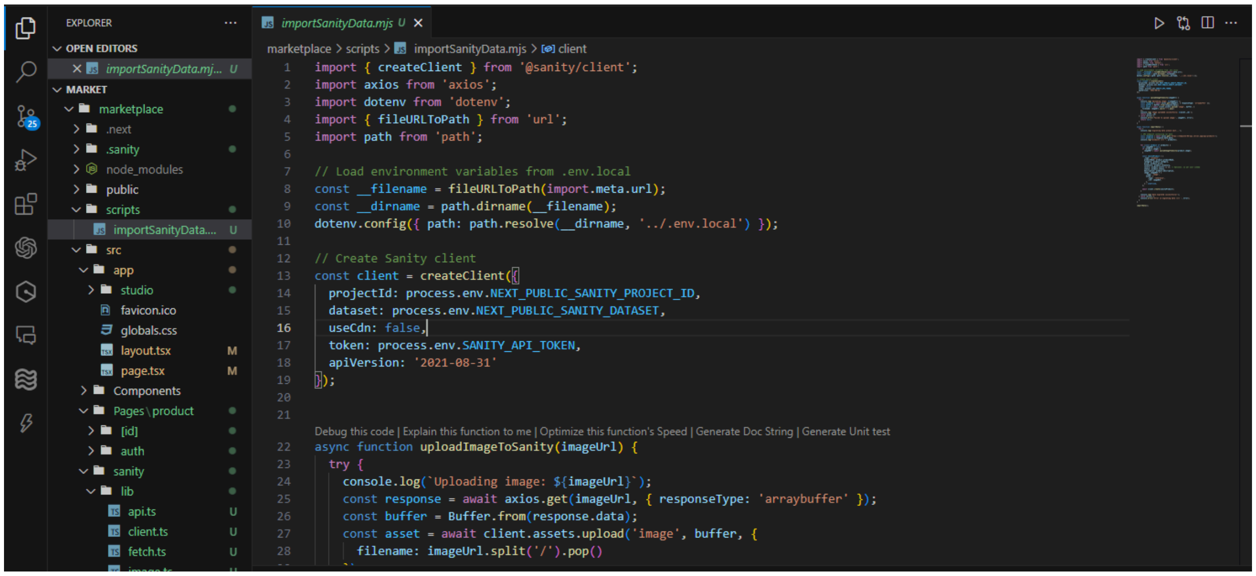


```
import { defineQuery } from 'next-sanity';

export const allProducts = defineQuery(
    `*[_type == "product"]{
    _id,
    productName,
    category,
    price,
    inventory,
    colors,
    status,
    description,
    "imageUrl": image-asset->url
      }`
)

export const fourPro = defineQuery(
    `*[_type == "product"][0..3]{
    _id,
    productName,
    category,
    price,
    inventory,
    colors,
    status,
    description,
    "imageUrl": image-asset->url
      }`
```

Migration Script:



```
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_API_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()
```

Prepared by:
Ayesha Nasir

6. Best Practices Followed

1. **Using `.env` files for Sensitive Data**:
   Environment variables were used to securely store sensitive information like API keys and project details.

2. **Clean Code Practices**:

   - Descriptive variable names like `products`, `productName`, `price`.
   - Functions were modularized for reusability.
   - Complex logic was documented with comments.
3. **Data Validation During Migration**:
   Schemas were used to ensure correct field types. Any discrepancies were logged for later investigation.

4. **Version Control**:
   Changes were committed frequently with meaningful commit messages such as "Added product schema" and "Completed API integration".

5. **Thorough Testing**:
   The API integration was tested using tools like Postman to ensure that endpoints were correctly returning data.

6. **Peer Review**:
   Code and documentation were shared with peers for feedback, and improvements were made based on the suggestions.

Conclusion:

The API integration and data migration were successfully implemented, ensuring that product data is dynamically fetched from Sanity CMS and displayed on the frontend. All steps were documented, and the code is now ready for further development and testing.

Prepared by:
Ayesha Nasir