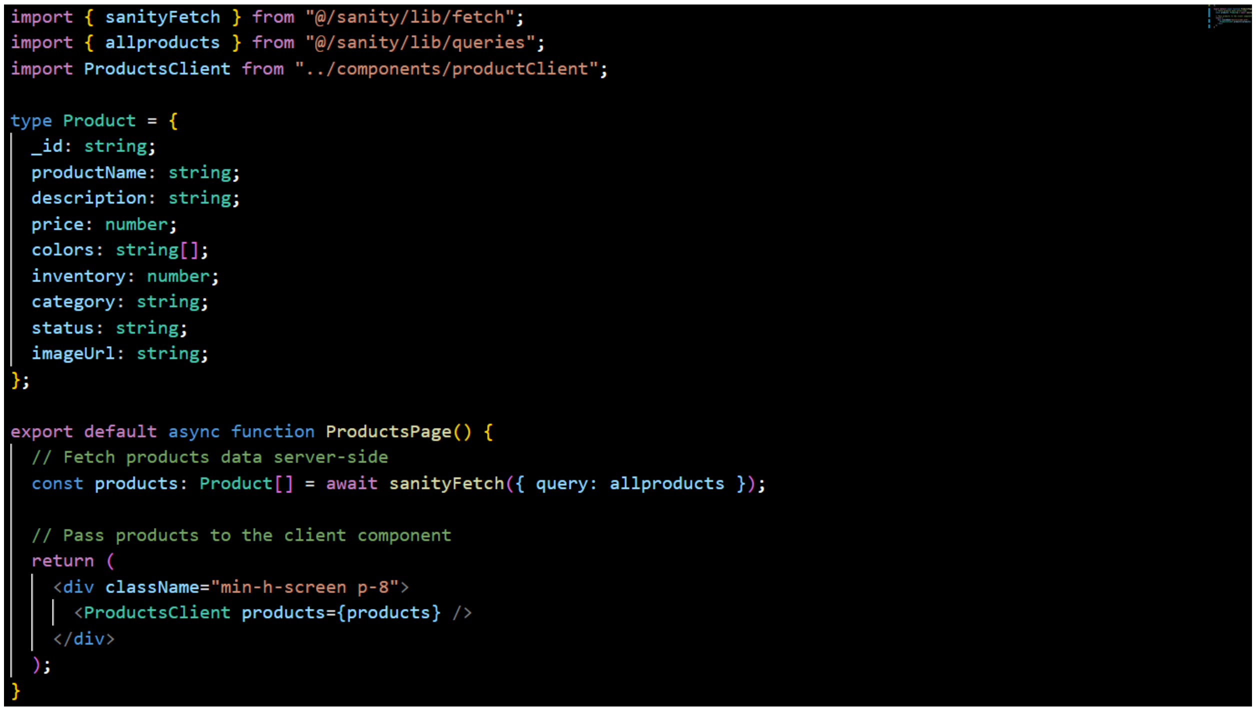
Day 3 - API Integration and Data Migration Report - Nike Marketplace

1. API Integration Process:

The **API integration process** for the Nike Marketplace involves several steps to fetch product data from an external API and display it on the frontend. Here's how the process was executed:

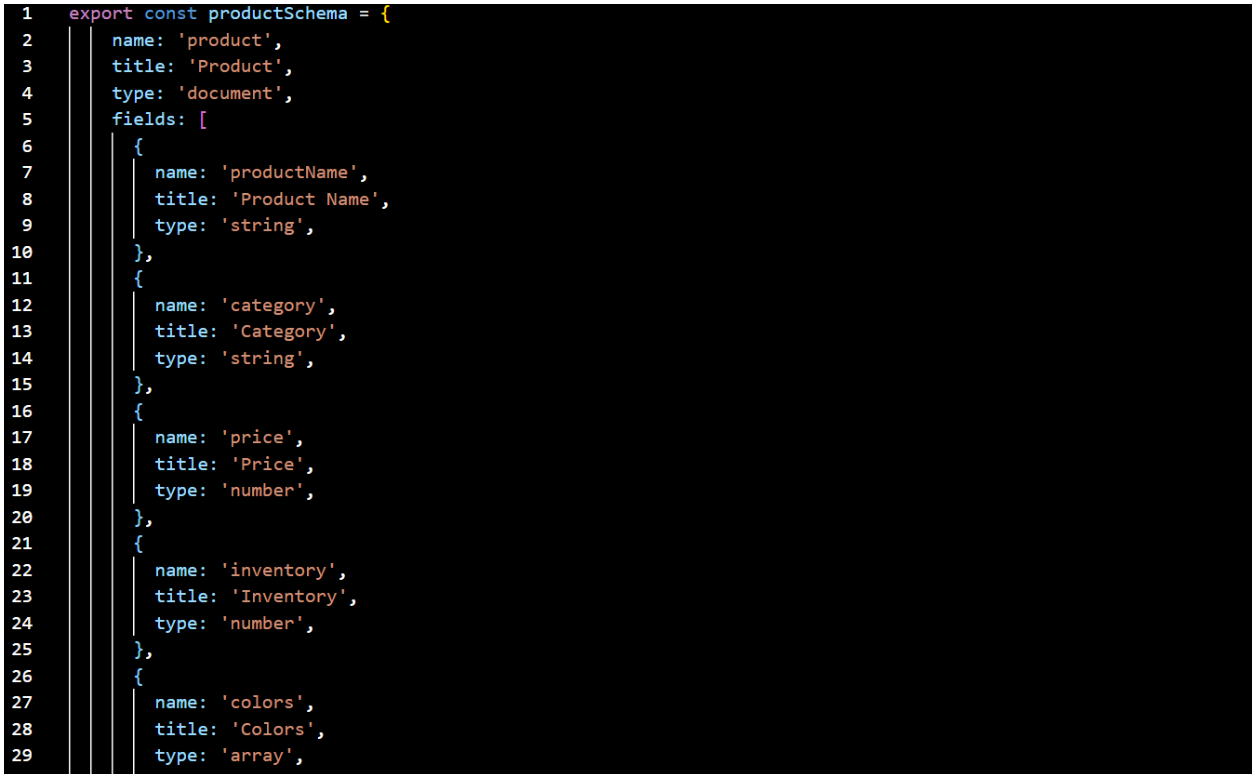
1. **API Provided**: The external API was provided to fetch product-related data for the marketplace. This API enabled us to retrieve key details such as product names, descriptions, prices, and images for each product.
2. **Sanity Project Creation**: A project was created on **Sanity.io**, which is used for managing content. This platform was selected to store and manage product data fetched from the external API. By integrating this with the frontend, we aimed to dynamically display products.
3. **API Token Configuration**: The **API token** provided for accessing the external API was added to the **.env** file of the project. This token allows secure access to the API and prevents unauthorized access to our system.
4. **API Integration**:
    a. Using the token, **GET** requests were made to the API to fetch product data.
    b. The data fetched was processed on the backend, structured appropriately, and then inserted into the system.
    c. API integration also involves handling various types of data responses, managing errors, and formatting the data for frontend display.
5. **Frontend Display**: Once the data was fetched and processed, it was displayed on the frontend of the marketplace using dynamic rendering techniques. Data like product names, descriptions, prices, and images were displayed on the product pages of the marketplace.



```
import { sanityFetch } from "@/sanity/lib/fetch";
import { allproducts } from "@/sanity/lib/queries";
import ProductsClient from "../components/productClient";

type Product = {
  _id: string;
  productName: string;
  description: string;
  price: number;
  colors: string[];
  inventory: number;
  category: string;
  status: string;
  imageUrl: string;
};

export default async function ProductsPage() {
  // Fetch products data server-side
  const products: Product[] = await sanityFetch({ query: allproducts });

  // Pass products to the client component
  return (
    <div className="min-h-screen p-8">
      <ProductsClient products={products} />
    </div>
  );
}
```

2. Adjustments Made to Schemas:

- As part of the **API integration**, the schema in **Sanity.io** was modified to store the product data fetched from the external API. These adjustments included creating and altering fields to match the structure of the API data:
- **Sanity Schema Creation**: A new schema was created in **Sanity** to store product details. This schema includes the following fields:
    - **productId**: A unique identifier for each product.
    - **name**: The name of the product.
    - **description**: A detailed description of the product.
    - **price**: The price of the product.
    - Additional fields like **imageUrl** and **category** were also added, depending on the needs of the marketplace.
- **Schema Modification**: The schema was adjusted to ensure it matched the structure of the data fetched from the external API. Each field was carefully mapped to allow smooth data storage and retrieval. This ensures that the data fetched through the API could be stored in a consistent manner and displayed accurately on the frontend.



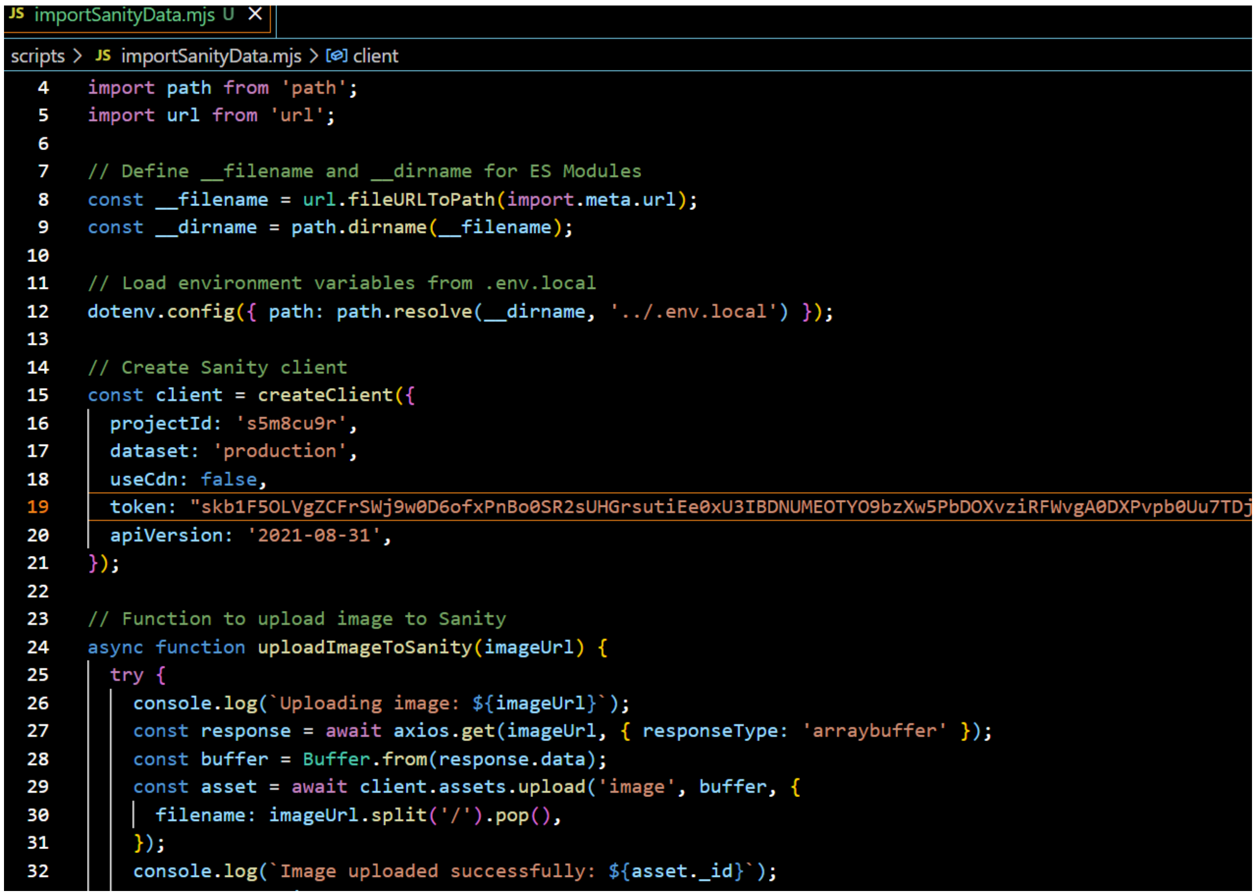
```
export const productSchema = {
    name: 'product',
    title: 'Product',
    type: 'document',
    fields: [
      {
        name: 'productName',
        title: 'Product Name',
        type: 'string',
      },
      {
        name: 'category',
        title: 'Category',
        type: 'string',
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

.

3. Migration Steps and Tools Used:

- **Data migration** involves moving product data from an external system into the new system, in this case, from the external API to **Sanity CMS**. Below are the steps involved in the migration process:
- **Data Fetching**: The data was fetched from the external system using the provided API. The product data, such as name, description, price, and image, was collected in a structured format.
- **Migration Script**: A **migration script** was created to map the fetched data to the Sanity schema. The migration process involves:
  - Fetching data from the old system.
  - Mapping the fetched data to the new schema in **Sanity CMS**.
  - Inserting the mapped data into the **Sanity Studio**.
- **Build and Data Import**:
  - After creating the migration file, **npm run build** was executed. This command triggered the migration process, allowing the data to be imported into **Sanity CMS**.

    - Once the data was imported, it was verified in the **Sanity Vision** tab to ensure it was correctly populated.
- **GRoQ Query**: After the migration was successful, a **GRoQ query** was used to retrieve the data from Sanity CMS and display it on the frontend. The query was tested and verified in **Sanity Vision**.



```
import path from 'path';
import url from 'url';

// Define __filename and __dirname for ES Modules
const __filename = url.fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);

// Load environment variables from .env.local
dotenv.config({ path: path.resolve(__dirname, '../.env.local') });

// Create Sanity client
const client = createClient({
  projectId: 's5m8cu9r',
  dataset: 'production',
  useCdn: false,
  token: "skb1F5OLVgZCFrSWj9w0D6ofxPnBo0SR2sUHGrsutiEe0xU3IBDNUMEOTYO9bzXw5PbDOXvziRFWvgA0DXPvpb0Uu7TDj
  apiVersion: '2021-08-31',
});

// Function to upload image to Sanity
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}`);
```

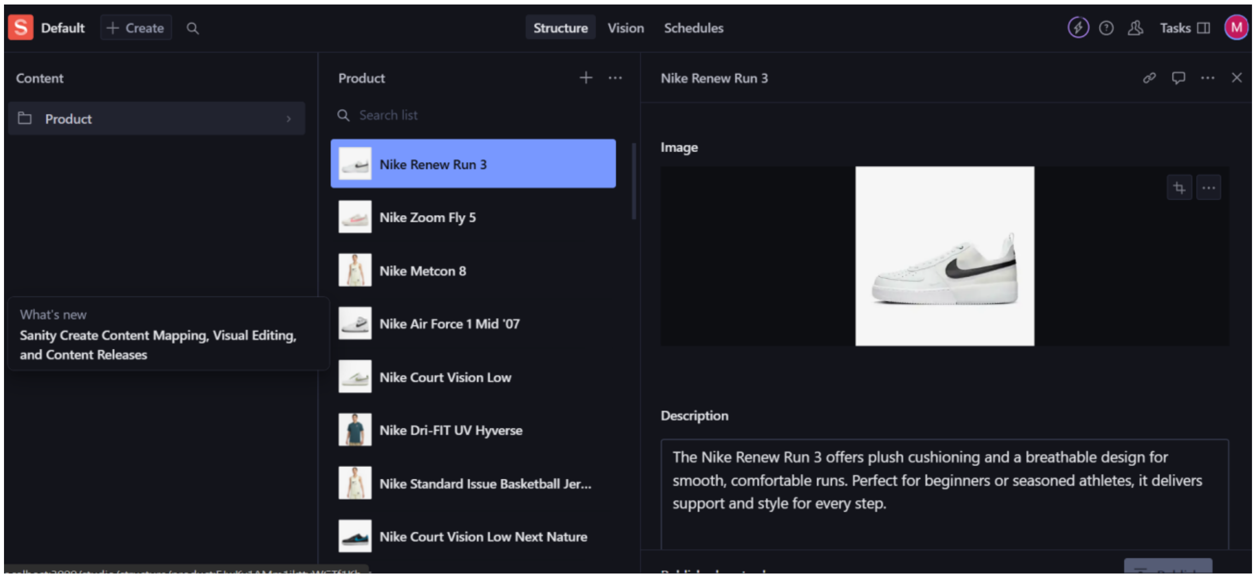
4. Data Successfully Displayed on Frontend:

- Once the data was successfully migrated and verified in **Sanity CMS**, it was displayed on the **Nike Marketplace frontend**. This process involved fetching the data dynamically and rendering it on the product pages.
- **Frontend Data Display**: The product data, including **name**, **description**, **price**, and **image**, was dynamically fetched from **Sanity CMS** and rendered using frontend technologies like **React.js** or another frontend framework.

- **Frontend Testing**: The product details were rendered successfully on the **local development server**. The data was displayed in a well-structured manner, with each product's details properly formatted and visible to the user.
- **Screenshots**:
- **Frontend Display Screenshot**: Include a screenshot showing how the **product details** (name, description, price, image) are displayed on the frontend of the marketplace.

5. Populated Sanity CMS Fields:

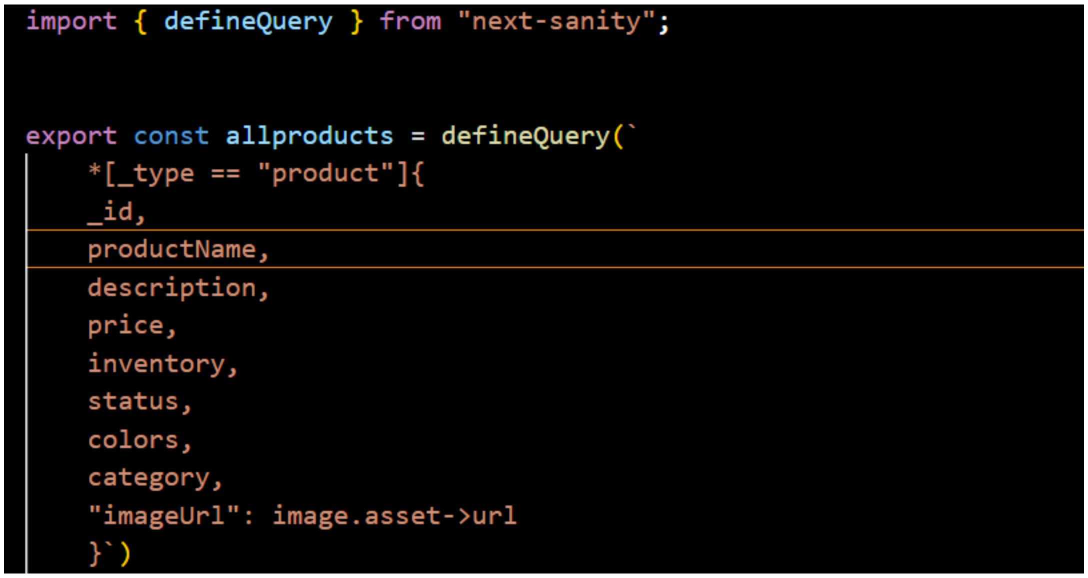
- After the migration, the product data was successfully populated into the fields of **Sanity CMS**. The fields were populated with product data fetched from the API.
- **Data Population**: In **Sanity Studio**, the product information was confirmed to be populated correctly in the corresponding fields. This includes the **productId**, **name**, **description**, and **price** for each product.
- **Verification**: We verified that all the fields were populated correctly by checking the entries in **Sanity Studio**. The data was correctly mapped and appeared in the appropriate format.



The Nike Renew Run 3 offers plush cushioning and a breathable design for smooth, comfortable runs. Perfect for beginners or seasoned athletes, it delivers support and style for every step.

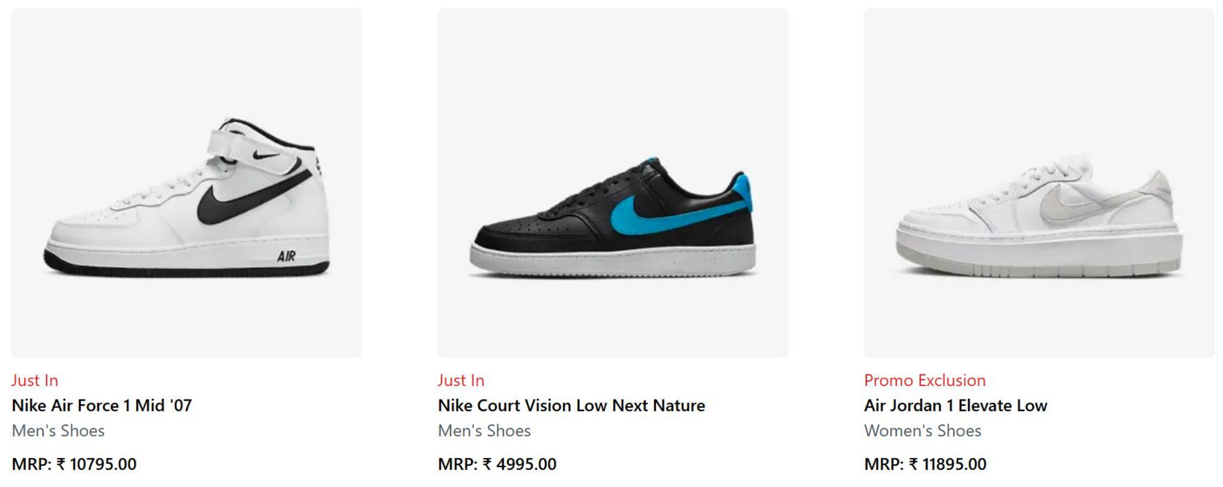
Code Snippets for API Integration and Migration Scripts:

1-Queries Implementation:



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


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

2-Frontend Display:



Just In
Nike Air Force 1 Mid '07
Men's Shoes
MRP: ₹ 10795.00

Just In
Nike Court Vision Low Next Nature
Men's Shoes
MRP: ₹ 4995.00

Promo Exclusion
Air Jordan 1 Elevate Low
Women's Shoes
MRP: ₹ 11895.00

Conclusion:

In conclusion, this report outlines the successful integration of the external API into the **Nike Marketplace**. The product data was fetched from the external API, migrated to **Sanity**

CMS, and successfully displayed on the frontend. The process involved adjusting schemas, migrating data using scripts, and querying the CMS using **GRoQ queries** to ensure the data was dynamically rendered.