Report on API Integration and Data Migration in Next.js Using External APIs and Sanity

1. Introduction API integration and data migration are critical components of modern web development, enabling applications to interact with external services and manage content effectively. In this report, we outline the steps required to integrate APIs and migrate data in a Next.js application using external APIs and Sanity, a popular headless CMS.

2. Prerequisites

Before starting, ensure the following tools and libraries are set up:

- **Next.js Project: Create a Next.js project using npx create-next-app.**
- **Sanity CMS: Set up a Sanity project at Sanity.io.**
- **Node.js: Install Node.js and npm.**
- **HTTP Client: Use libraries like Axios or Fetch API for HTTP requests.**
- **Sanity Client: Install the Sanity client SDK using npm install @sanity/client.**

3. API Integration Steps

1.Identify the API Endpoint:

Review the API documentation to identify endpoints, request methods (GET, POST, PUT, DELETE), authentication methods, and response formats.

2.Install Dependencies:

For external API integration, install necessary dependencies like Axios:

```
npm install axios
```

3.Create an API Service:

Abstract API calls into a service file for reusability.

Example:

```
import axios from 'axios';

 const apiClient = axios.create({
baseURL: 'https://api.example.com',
headers: {
  Authorization: `Bearer ${process.env.API_KEY}`,
},
});
```

```
export const fetchData = async () => {
  try {
    const response = await apiClient.get('/data-endpoint');
    return response.data;  } catch (error) {
    console.error('Error fetching data:', error);
    throw error;
          }
        };
```

4. Environment Variables:

Store sensitive information like API keys in environment variables.

Add these variables in a .env.local file:

```
API_KEY=your_api_key
```

5. Server-Side Data Fetching:

Use Next.js data-fetching methods like getServerSideProps or getStaticProps for integrating data during server-side rendering (SSR) or static generation (SSG).

Example:

```
export async function getServerSideProps() {
  const data = await fetchData();
  return { props: { data } };
}
```

6.Client-Side Data Fetching:

For dynamic updates, fetch data on the client side using useEffect.

7.Error Handling in API Integration:

Handle errors to ensure robustness and better debugging:

```
export const fetchData = async () => {
  try {
    const response = await apiClient.get('/data-endpoint');
    return response.data;
  } catch (error) {
    if (error.response) {
      console.error('Server responded with an error:',
error.response.data);
    } else if (error.request) {
     console.error('No response received:', error.request);
    } else {
      console.error('Error setting up request:',
error.message);
    }
    throw error;
  }
};
```

4. Data Migration Using Sanity

1. **Sanity Project Setup:**
   - **Create a new project in Sanity using:**
   - **npx sanity init**
   - **Define schemas for your content in the schemas folder.**
2. **Sanity Client Configuration:**

   Configure the Sanity client:

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

 const sanityClient = createClient({
projectId: 'your_project_id',
dataset: 'production',
useCdn: true,
});

export default sanityClient;
```

3. **Sanity Validation:**

   Sanity validation refers to ensuring the integrity and correctness of data based on defined schema rules. For example, you can define required fields or specific value types in your schema:

```
export default {
  name: 'post',
  title: 'Post',
  type: 'document',
  fields: [
    {
      name: 'title',
      type: 'string',
      title: 'Title',
      validation: (Rule) =>
Rule.required().min(5).max(100),
    },
  {
      name: 'content',
   type: 'text',
   title: 'Content',
   validation: (Rule) => Rule.required(),
    },
  ],
};
```

- Validation ensures that only data conforming to these rules can be created or updated.

4. <u>**Migrating Data:**</u>

Use Sanity's GROQ (Graph-Relational Object Queries) for querying data:

```
export const fetchSanityData = async () => {
  const query = '*[_type == "post"]{title, content}';
  return await sanityClient.fetch(query);
};
```

5. <u>Uploading External API Data to Sanity:</u>

Transform data from an external API to match Sanity's schema.

Example:

```
export const uploadToSanity = async (data) => {
  try {
    const response = await sanityClient.create({
   _type: 'post',
title: data.title,
      content: data.content,
    });
    return response;
  } catch (error) {
    console.error('Error uploading data:', error);
    throw error;
  }
};
```

6.Bulk Data Migration:

- **Loop through external API data and upload it to Sanity:**

```
const migrateData = async () => {
  const externalData = await fetchData();
  for (const item of externalData) {
    try {
      await uploadToSanity(item);
} catch (error) {
     console.error(`Failed to migrate item: ${item.id}`,
error);
    }
  }
};
```

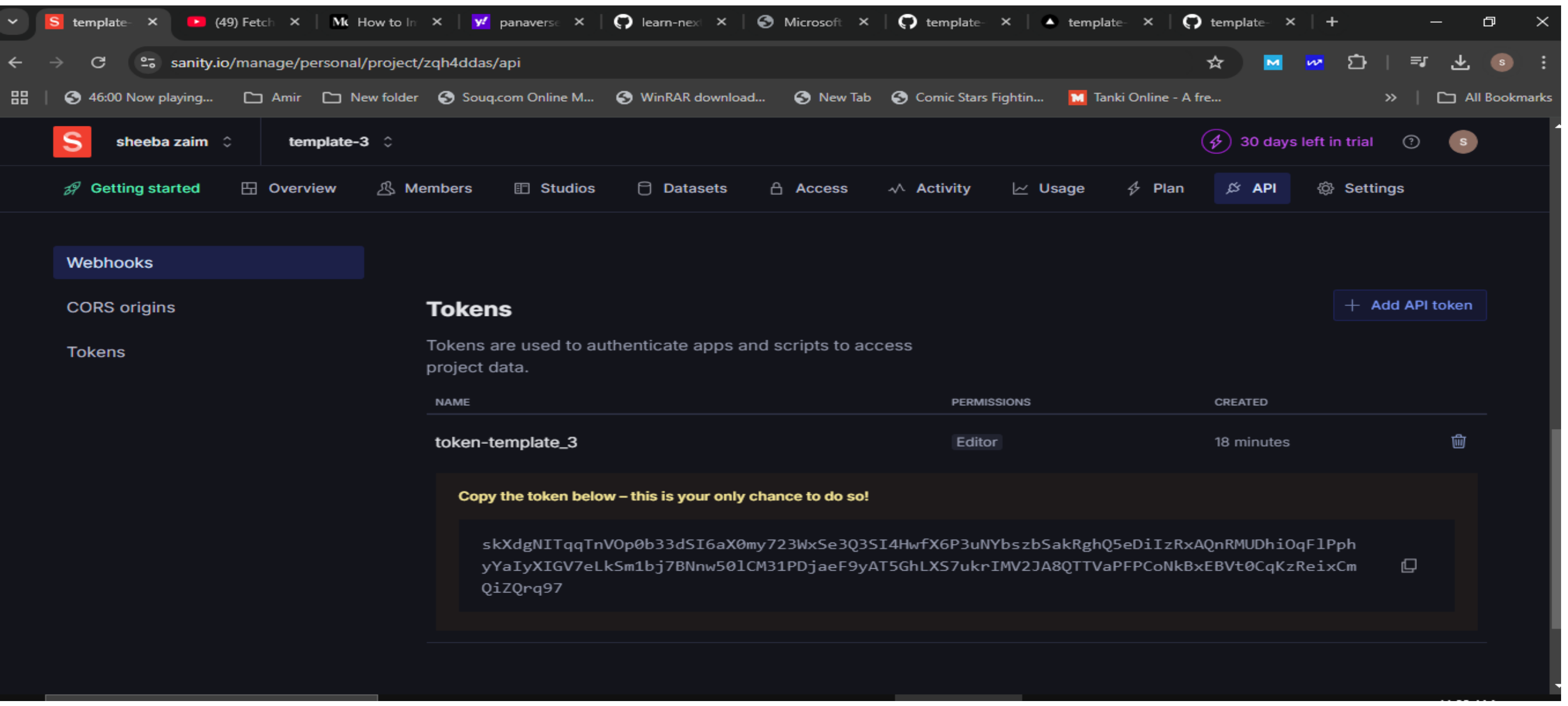
5. Testing and Debugging

1. **Test API Integration:**
   - Verify API responses using tools like Postman .
2. **Validate Data in Sanity:**
   - Use the Sanity studio dashboard to confirm data migration.
3. **Error Handling and Logs:**
   - Log errors for debugging and set up notifications for critical issues.
   - Use services like Sentry for error monitoring:
   - npm install @sentry/nextjs

6. Conclusion

By following these steps, we can effectively integrate external APIs and migrate data using Sanity in a Next.js application. This approach ensures a seamless and scalable way to manage dynamic content and external data sources.

Sanity token image:



Tokens

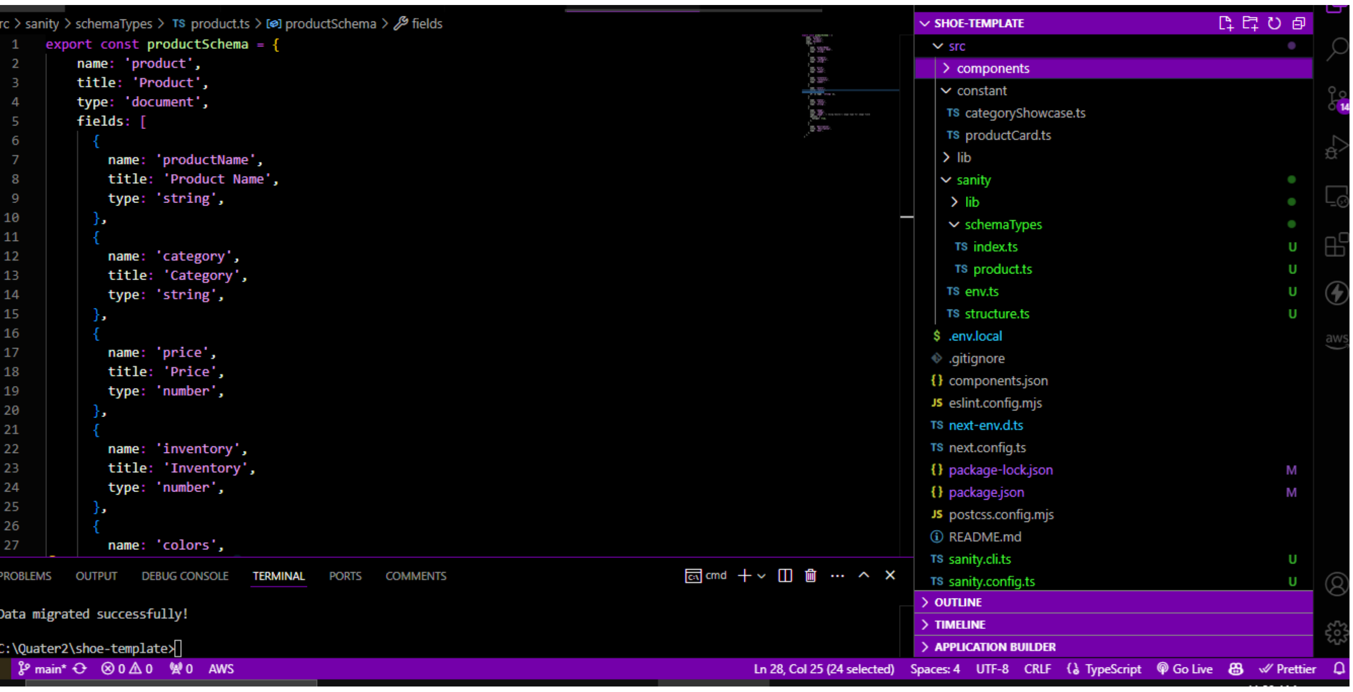
Tokens are used to authenticate apps and scripts to access project data.

NAME	PERMISSIONS	CREATED
token-template_3	Editor	18 minutes

Copy the token below – this is your only chance to do so!

```
skXdgNITqqTnVOp0b33dSI6aX0my723WxSe3Q3SI4HwfX6P3uNYbszbSakRghQ5eDiIzRxAQnRMUDhiOqFlPph
yYaIyXIGV7eLkSm1bj7BNnw50lCM31PDjaeF9yAT5GhLXS7ukrIMV2JA8QTTVaPFPCoNkBxEBVt0CqKzReixCm
QiZQrq97
```

Schema file:

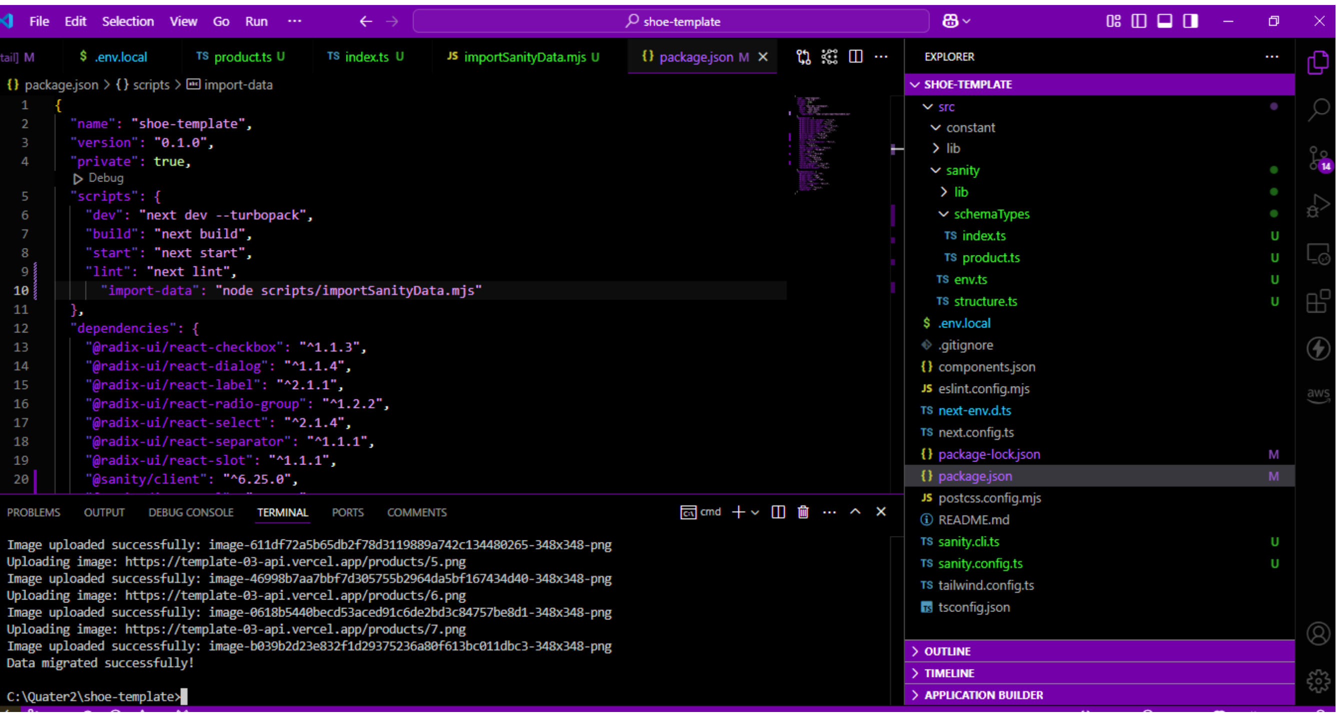


rc > sanity > schemaTypes > product.ts > productSchema > fields

```ts
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',
```

```
Data migrated successfully!

C:\Quater2\shoe-template>
```

Data Migration:

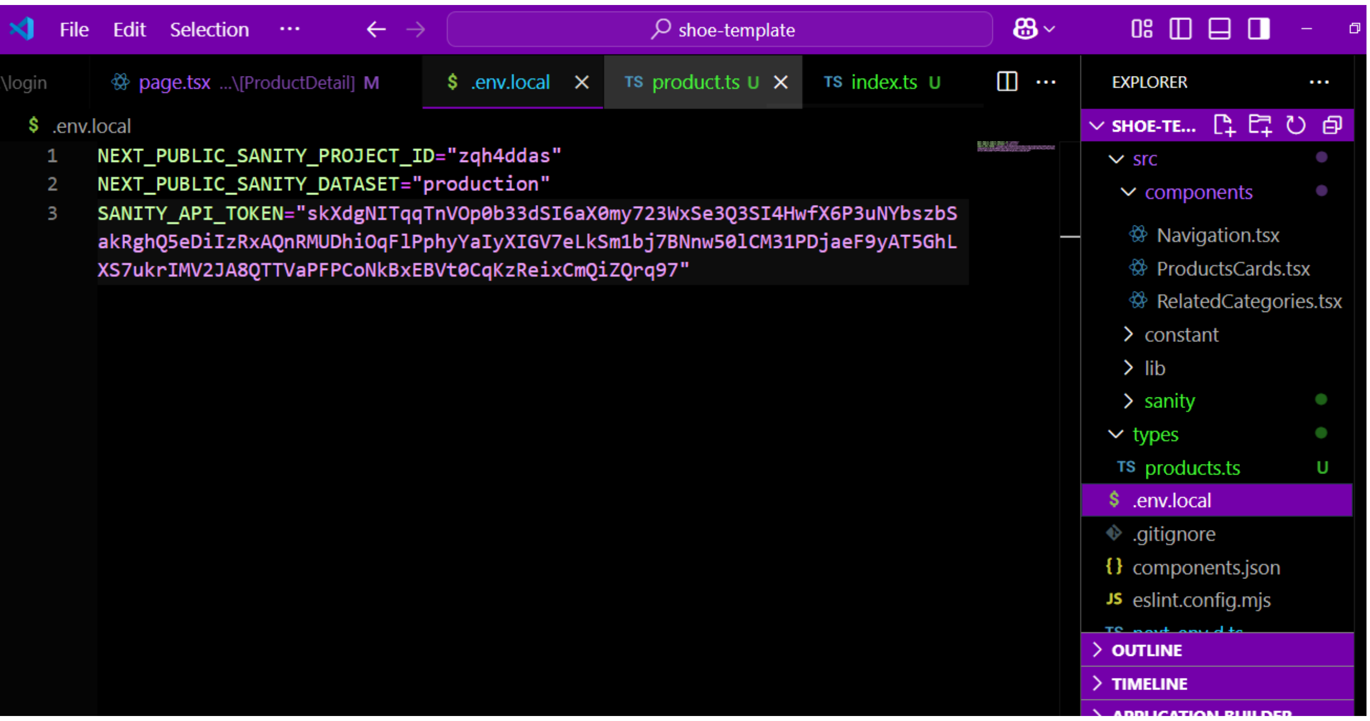


package.json > scripts > import-data

```json
{
  "name": "shoe-template",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "dev": "next dev --turbopack",
    "build": "next build",
    "start": "next start",
    "lint": "next lint",
      "import-data": "node scripts/importSanityData.mjs"
  },
  "dependencies": {
    "@radix-ui/react-checkbox": "^1.1.3",
    "@radix-ui/react-dialog": "^1.1.4",
    "@radix-ui/react-label": "^2.1.1",
    "@radix-ui/react-radio-group": "^1.2.2",
    "@radix-ui/react-select": "^2.1.4",
    "@radix-ui/react-separator": "^1.1.1",
    "@radix-ui/react-slot": "^1.1.1",
    "@sanity/client": "^6.25.0",
```

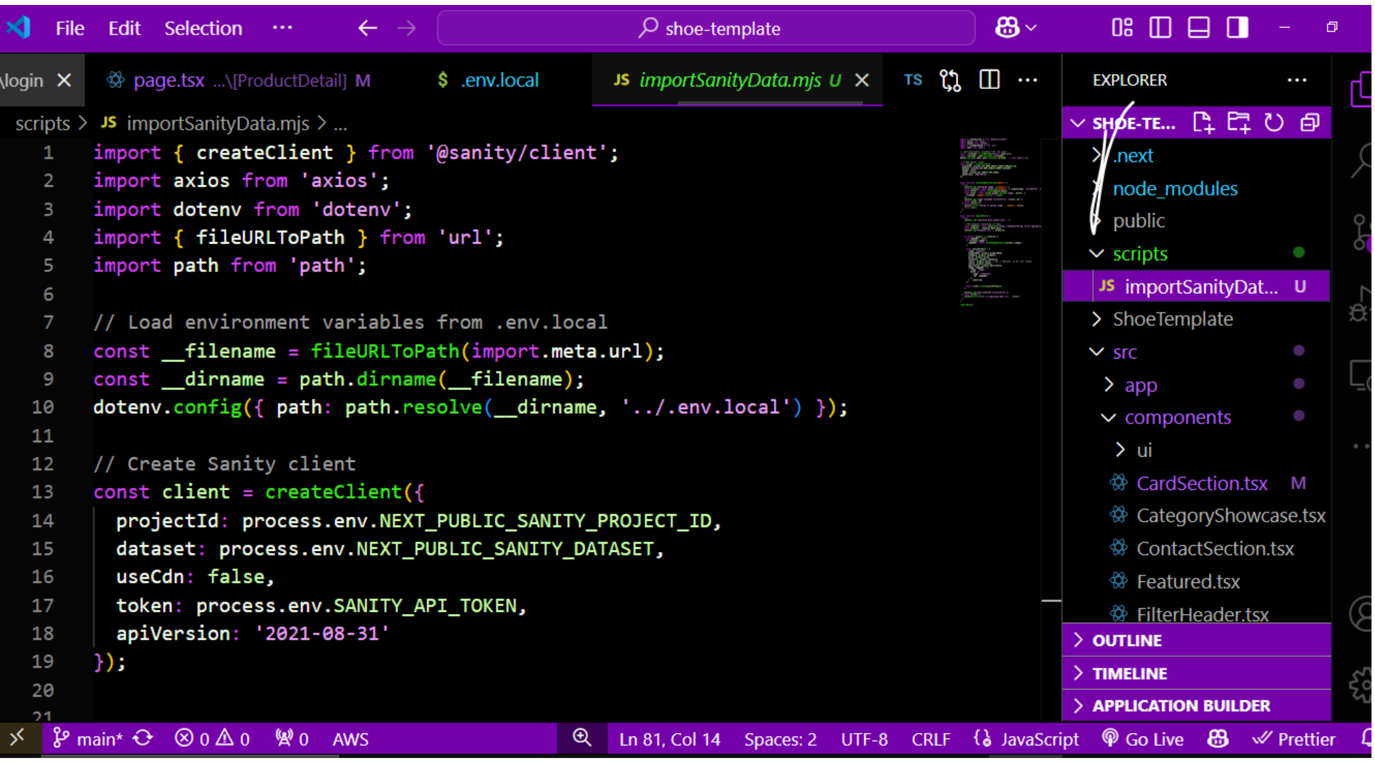
```
Image uploaded successfully: image-611df72a5b65db2f78d3119889a742c134480265-348x348-png
Uploading image: https://template-03-api.vercel.app/products/5.png
Image uploaded successfully: image-46998b7aa7bbf7d305755b2964da5bf167434d40-348x348-png
Uploading image: https://template-03-api.vercel.app/products/6.png
Image uploaded successfully: image-0618b5440becd53aced91c6de2bd3c84757be8d1-348x348-png
Uploading image: https://template-03-api.vercel.app/products/7.png
Image uploaded successfully: image-b039b2d23e832f1d29375236a80f613bc011dbc3-348x348-png
Data migrated successfully!

C:\Quater2\shoe-template>
```

Env local file:



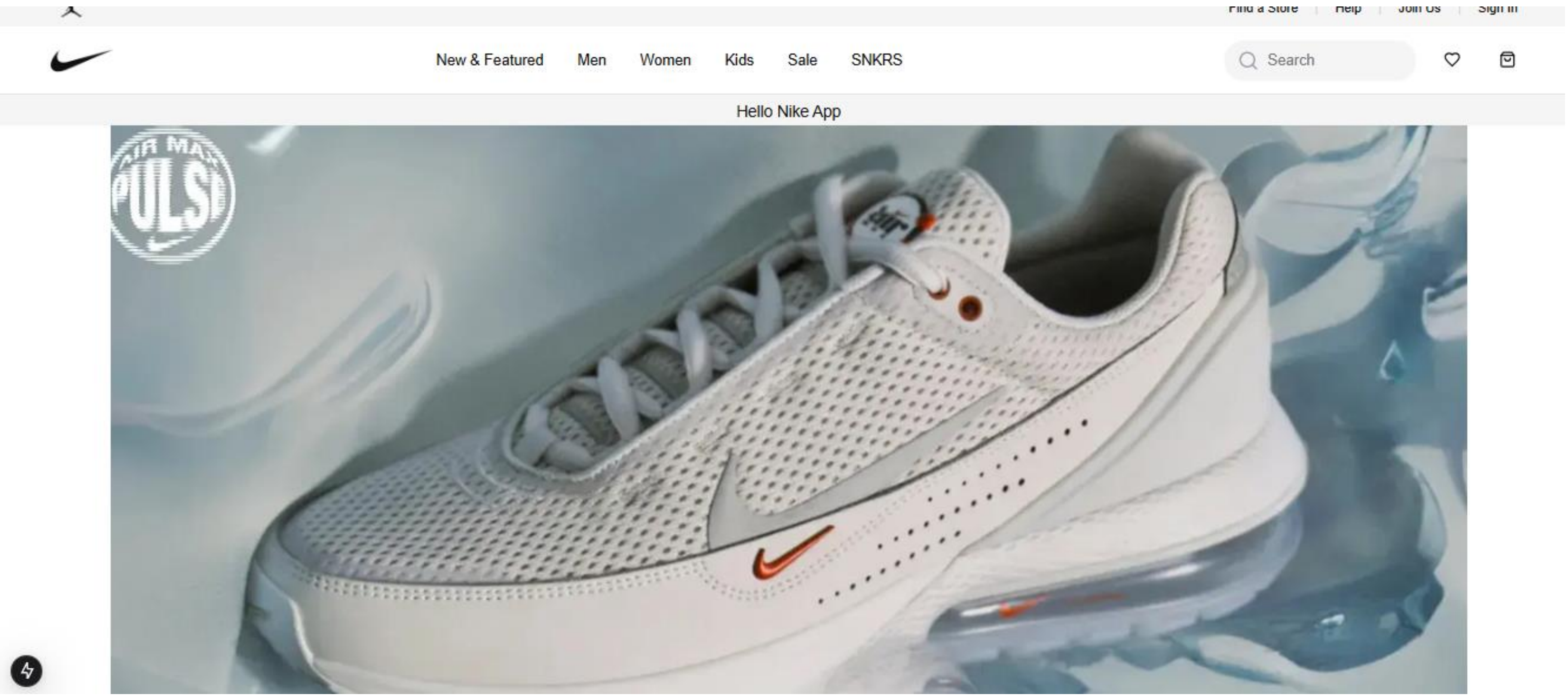
```
NEXT_PUBLIC_SANITY_PROJECT_ID="zqh4ddas"
NEXT_PUBLIC_SANITY_DATASET="production"
SANITY_API_TOKEN="skXdgNITqqTnVOp0b33dSI6aX0my723WxSe3Q3SI4HwfX6P3uNYbszbSakRghQ5eDiIzRxAQnRMUDhiOqFlPphyYaIyXIGV7eLkSm1bj7BNnw50lCM31PDjaeF9yAT5GhLXS7ukrIMV2JA8QTTVaPFPCoNkBxEBVt0CqKzReixCmQiZQrq97"
```

Script folder mjs file:



```
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'
});
```

View after integration:


Find a Store
Help
Join Us
Sign In
New & Featured
Men
Women
Kids
Sale
SNKRS
Search
Hello Nike App
AIR MAX
PULSE