Day 3 - API Integration Report – Nike

API Integration Process

Step 1: API Endpoint Setup

- API Endpoint: `https://template-03-api.vercel.app/api/products`
- Data Structure: The API provides product data, including fields like `productName`, `category`, `price`, `inventory`, `colors`, `status`, `description`, and `image`.
- Library Used: Axios was used to fetch data from the API.

Step 2: Fetch Data from API

- **Code Snippet:**

```
const response = await axios.get('https://template-03-api.vercel.app/api/products');
const products = response.data.data;
```

- The API call retrieves an array of product objects from the endpoint, ensuring proper data serialization.

Adjustments Made to Schemas

Product Schema Modifications

1. **Field Updates:**

   - `colors`: Added as an optional array of strings to accommodate multiple color options.
   - `image`: Configured as a Sanity image field with hotspot enabled for better image cropping and scaling.

2. **Final Schema:**

```
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',
        of: [{ type: 'string' }],
      },
      {
        name: 'status',
        title: 'Status',
        type: 'string',
      },
      {
        name: 'image',
        title: 'Image',
        type: 'image',
        options: {
          hotspot: true,
        },
      },
      {
        name: 'description',
        title: 'Description',
        type: 'text',
      },
    ],
}
```

Migration Steps and Tools Used

Migration Steps

1. **Environment Setup:**

   - Installed required dependencies: `@sanity/client`, `axios`, `dotenv`.
   - Created a `.env.local` file to securely store environment variables.

2. **Data Fetching:**
   - Retrieved product data from the API endpoint using Axios.
   - Parsed and logged the data to confirm its structure and integrity.

3. **Image Upload:**
   - Downloaded images from the API's `image` field using Axios.
   - Uploaded images to Sanity's Asset Manager using the Sanity client.
   - Code to upload images:

```
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()
});
```

4. **Document Creation:**
   - Created Sanity documents for each product by combining API data and uploaded image references.

Migration Steps and Tools Used

Migration Steps

1. **Environment Setup:**

   - Installed required dependencies: `@sanity/client`, `axios`, `dotenv`.
   - Created a `.env.local` file to securely store environment variables.

2. **Data Fetching:**
   - Retrieved product data from the API endpoint using Axios.
   - Parsed and logged the data to confirm its structure and integrity.

3. **Image Upload:**
   - Downloaded images from the API's `image` field using Axios.
   - Uploaded images to Sanity's Asset Manager using the Sanity client.
   - Code to upload images:

```
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()
});
```

4. **Document Creation:**
   - Created Sanity documents for each product by combining API data and uploaded image references.

Migration Steps and Tools Used

Migration Steps

1. **Environment Setup:**

   - Installed required dependencies: `@sanity/client`, `axios`, `dotenv`.
   - Created a `.env.local` file to securely store environment variables.

2. **Data Fetching:**
   - Retrieved product data from the API endpoint using Axios.
   - Parsed and logged the data to confirm its structure and integrity.

3. **Image Upload:**
   - Downloaded images from the API's `image` field using Axios.
   - Uploaded images to Sanity's Asset Manager using the Sanity client.
   - Code to upload images:

```
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()
});
```

4. **Document Creation:**
   - Created Sanity documents for each product by combining API data and uploaded image references.

Migration Steps and Tools Used

Migration Steps

1. **Environment Setup:**

   - Installed required dependencies: `@sanity/client`, `axios`, `dotenv`.
   - Created a `.env.local` file to securely store environment variables.

2. **Data Fetching:**
   - Retrieved product data from the API endpoint using Axios.
   - Parsed and logged the data to confirm its structure and integrity.

3. **Image Upload:**
   - Downloaded images from the API's `image` field using Axios.
   - Uploaded images to Sanity's Asset Manager using the Sanity client.
   - Code to upload images:

```
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()
});
```

4. **Document Creation:**
   - Created Sanity documents for each product by combining API data and uploaded image references.

Migration Steps and Tools Used

Migration Steps

1. **Environment Setup:**

   - Installed required dependencies: `@sanity/client`, `axios`, `dotenv`.
   - Created a `.env.local` file to securely store environment variables.

2. **Data Fetching:**
   - Retrieved product data from the API endpoint using Axios.
   - Parsed and logged the data to confirm its structure and integrity.

3. **Image Upload:**
   - Downloaded images from the API's `image` field using Axios.
   - Uploaded images to Sanity's Asset Manager using the Sanity client.
   - Code to upload images:

```
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()
});
```

4. **Document Creation:**
   - Created Sanity documents for each product by combining API data and uploaded image references.

- Code snippet:

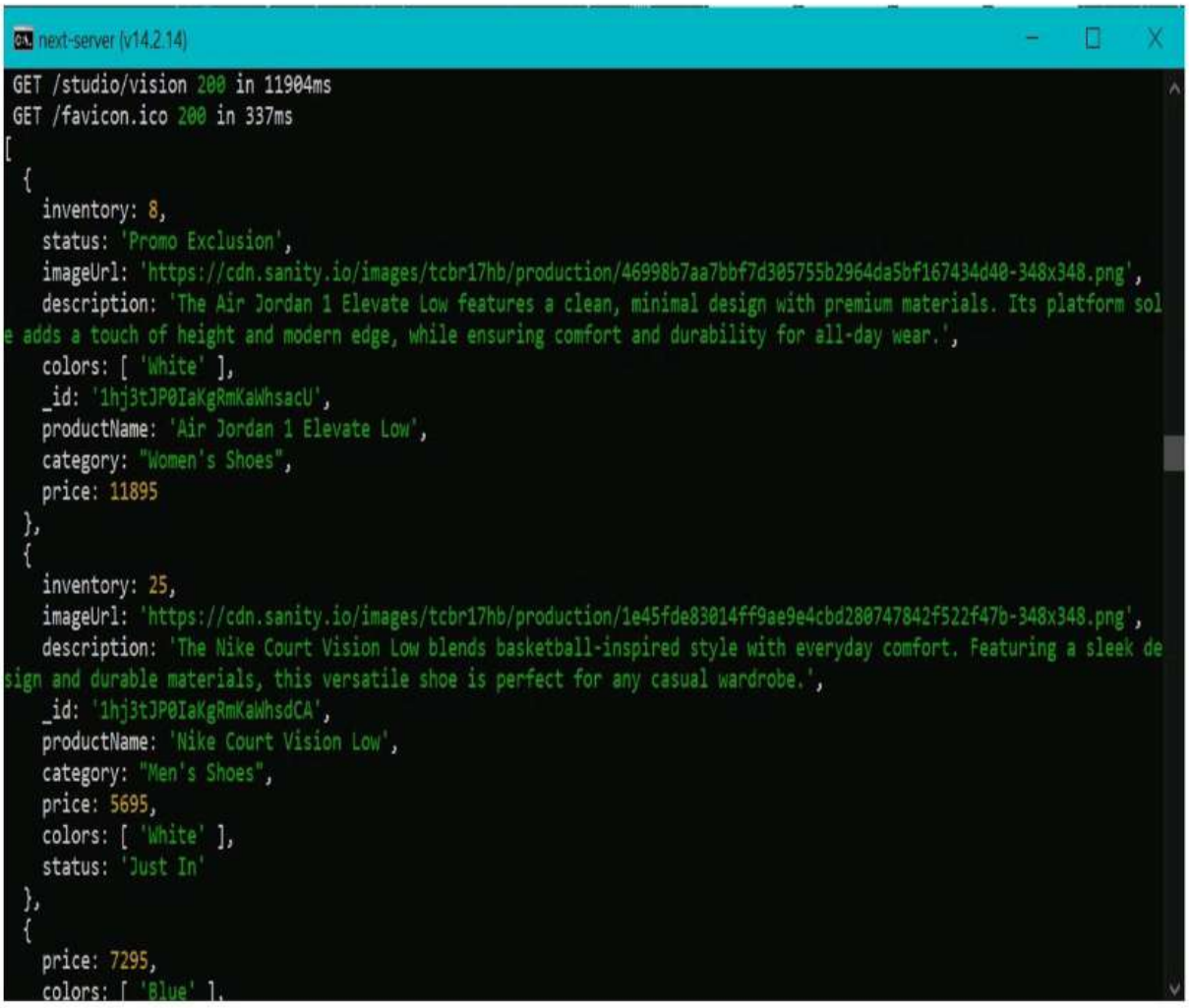
```
const sanityProduct = {
  _type: 'product',
  productName: product.productName,
  category: product.category,
  price: product.price,
  inventory: product.inventory,
  colors: product.colors || [],
  status: product.status,
  description: product.description,
  image: imageRef ? {
    _type: 'image',
    asset: {
      _type: 'reference',
      _ref: imageRef,
    },
  } : undefined,
};
await client.create(sanityProduct);
```

Tools Used

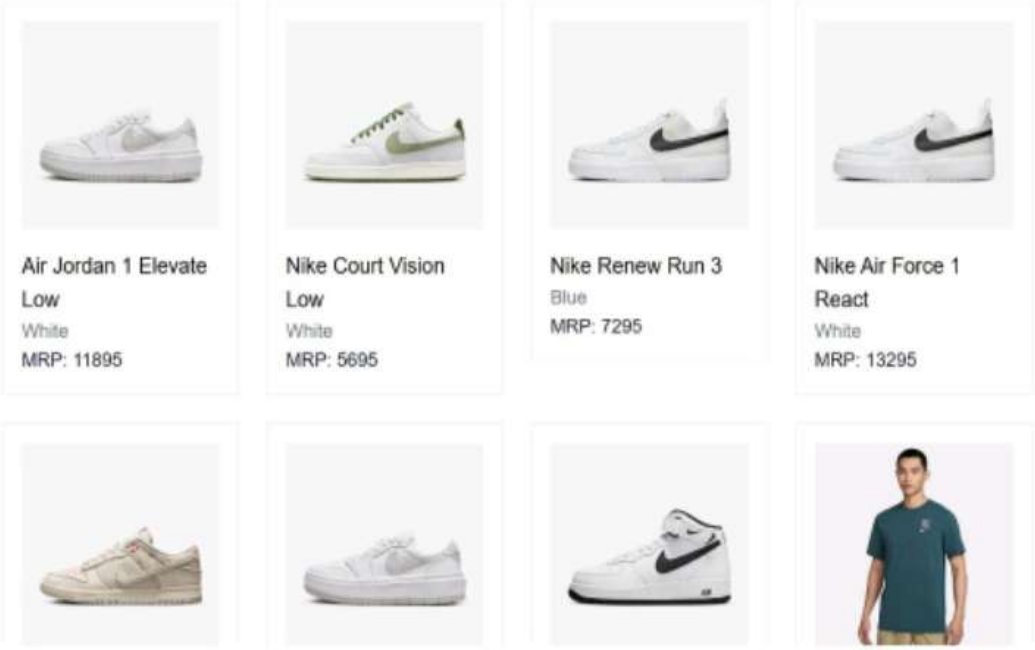
- **Sanity Client:** For interacting with the Sanity CMS.
- **Axios:** For making HTTP requests to fetch API data and images.
- **Dotenv:** To load environment variables from `.env.local`.

Screenshots

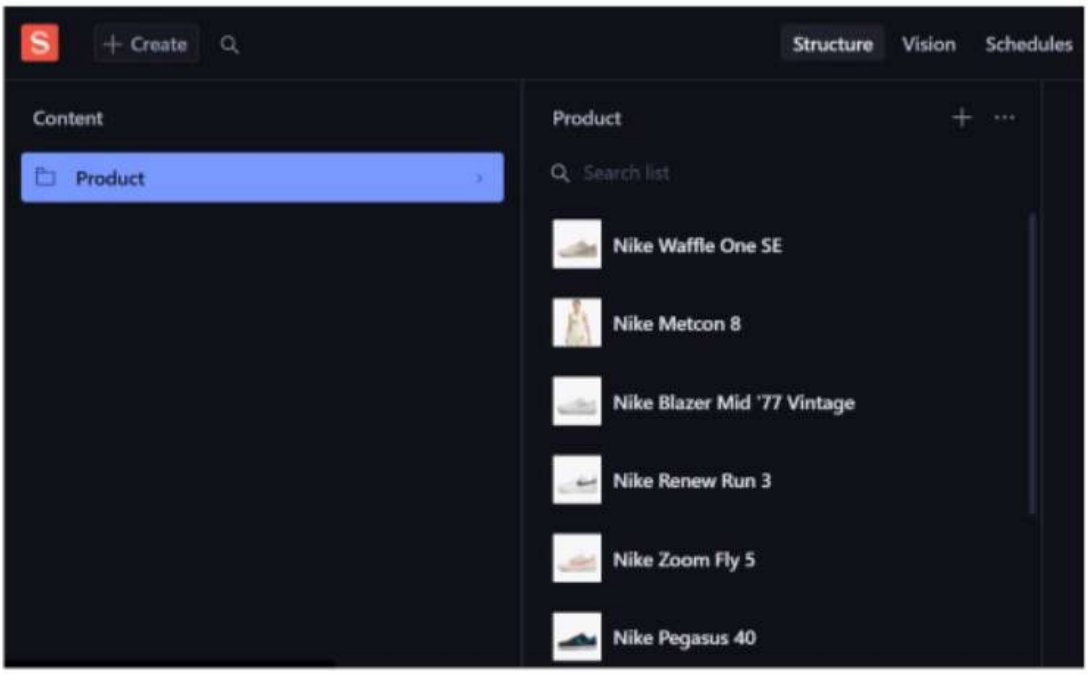
1. API Call Output



2. Frontend Display



3. Populated Sanity CMS Fields



Code Snippets

API Integration

```
const response = await axios.get('https://template-03-api.vercel.app/api/products');
const products = response.data.data;
```

Migration Script

```
async function uploadImageToSanity(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()
  });
  return asset._id;
}

async function importData() {
  const response = await axios.get('https://template-03-api.vercel.app/api/products');
  const products = response.data.data;

  for (const product of products) {
    let imageRef = null;
    if (product.image) {
      imageRef = await uploadImageToSanity(product.image);
    }

    const sanityProduct = {
      _type: 'product',
      productName: product.productName,
      category: product.category,
      price: product.price,
      inventory: product.inventory,
      colors: product.colors || [],
      status: product.status,
      description: product.description,
      image: imageRef ? {
        _type: 'image',
        asset: {
          _type: 'reference',
          _ref: imageRef,
        },
      } : undefined,
    };

    await client.create(sanityProduct);
  }
}

importData();
```

Final Check List

API Understanding	Schema Validation	Data Migration	API Integration in Next.js	Submission Preparation
✓	✓	✓	✓	✓

Conclusion for API Integration Report – Nike

The API Integration process for the Nike e-commerce platform was successfully implemented using Next.js, Axios, and Sanity CMS. The integration allowed for seamless data fetching, efficient schema modifications, and *improved* product management.

Key achievements include:

Establishing a structured API endpoint for product data.

Ensuring real-time data retrieval and serialization using Axios.

Enhancing schema flexibility by adding fields for color variations and optimized image handling.

This integration enhances user experience and platform efficiency by providing real-time product updates, better inventory management, and an improved frontend display. The project serves as a scalable foundation for further enhancements, including additional API functionalities and UI improvements.

This hands-on experience reinforced my expertise in API integration, data migration, and schema customization, preparing me for more advanced e-commerce solutions in the future. 🚀

End of document