API Integration Report: Avion E-Commerce

---

1. API Integration Process

API Selection:

The API integration process involved using a mock API endpoint to fetch product data:

- **API URL:** https://67808bae85151f714b070623.mockapi.io/api/products/productData

Data Fetching:

The data fetching process is performed using the Axios library to send GET requests to the mock API. The retrieved product data is then processed before being integrated into the Sanity CMS.

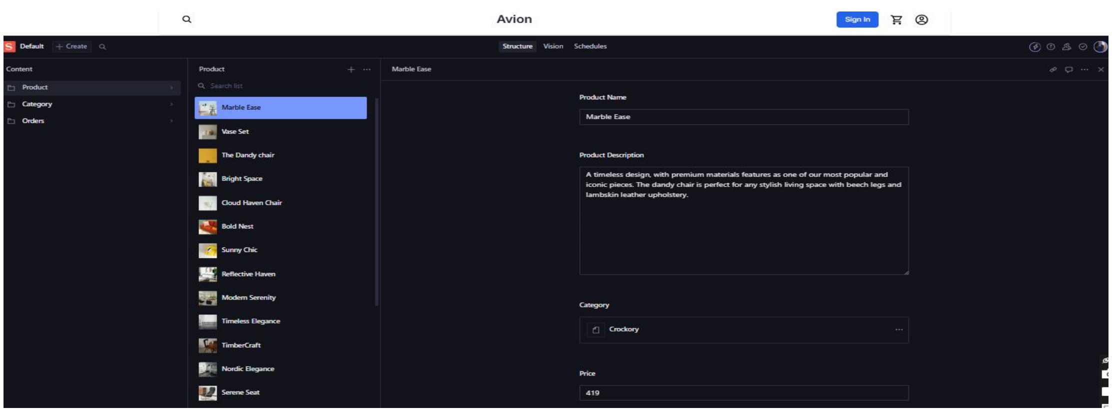
Sanity Integration:

The integration with Sanity CMS was achieved through:

- **Sanity Client Setup:** The project utilizes a custom Sanity client that includes the necessary credentials (project ID, dataset, and token).
- **Image Handling:** A custom function was created to handle image uploads, fetching the image from the URL, converting it to a buffer, and uploading it to Sanity as an asset.
- **Product Data Upload:** Each product's data was mapped into a schema and uploaded using the Sanity client. References to assets and inventories were also handled by fetching or creating them as required.

Key Steps:

- Fetching Data from the mock API.
- Uploading Images to Sanity.
- Creating Inventory References for each product.
- Inserting Product Data into Sanity using the client.



Product Name

Marble Ease

Product Description

A timeless design, with premium materials features as one of our most popular and iconic pieces. The dandy chair is perfect for any stylish living space with beech legs and lambskin leather upholstery.

Category

Crockory

Price

419

2. Adjustments Made to Schema

Initial Schema Setup:

Initially, the product schema and inventory schema were set up in Sanity. They included attributes such as name, price, description, rating, and product dimensions. However, some adjustments were required due to reference issues and the need to ensure that all data is properly linked.

Schema Adjustments:

- **Reference Adjustments:** Certain schemas needed references to link product data with related entities like inventory and images. This was handled by fetching existing references or creating new ones during the import process.
- **Stock and Inventory:** A dedicated function for handling inventory was introduced. This function ensures that stock levels, restock history, and sales data are created or fetched from Sanity.
- **Image and Product References:** The image URLs were uploaded separately, and references to the uploaded images were linked to products in the final product schema.
- **Order and Analytics Data:** Additional schemas for order analytics, inventory, shipment, and user data were also set up and populated.

---

3. Migration Steps and Tools Used

Migration Overview:

The migration process aimed at importing external data (from the mock API) into Sanity CMS while maintaining data integrity and ensuring the correct schema references were established.

Steps Involved:

1. **Sanity Project Setup:** A Sanity project was created, and the necessary credentials (project ID, dataset, and token) were obtained and integrated into the client setup.
2. **Axios for Data Fetching:** The Axios library was used to fetch data from the mock API. A function was created to handle the fetching of data and error handling.
3. **Schema Creation and Adjustment:** The schemas were first created in Sanity, including product, inventory, order, and shipment schemas. Adjustments to references between products, inventories, and images were made to ensure proper relationships.
4. **Data Import:** The data from the mock API was iterated over, and for each product:
    - Images were uploaded to Sanity using the custom image upload function.
    - Inventory references were either fetched or created.

   - Product data was mapped to the schema and uploaded to Sanity using the `client.createOrReplace()` method.

5. **Final Verification:** After the migration, the data was verified in Sanity for consistency, and the correct references were confirmed.

Tools and Technologies Used:

- **Axios:** For fetching data from the mock API.
- **Sanity Client:** For interacting with Sanity CMS, including fetching and uploading data.
- **Sanity Migration Script:** Custom migration script used for uploading data and images to Sanity, handling inventory references, and creating/updating product entries.
- **Node.js:** The entire migration process was handled using Node.js scripts.

Data Import from Sanity to Next.js & Display on Browser

Fetching Data from Sanity:

- To display product data on the Next.js frontend, a Sanity client was used to query the stored data.

Steps to Fetch and Display Data:

Sanity Client Setup in Next.js:

- A custom Sanity client was created using the project ID and dataset.
- GROQ queries were used to fetch product data.

API Route in Next.js:

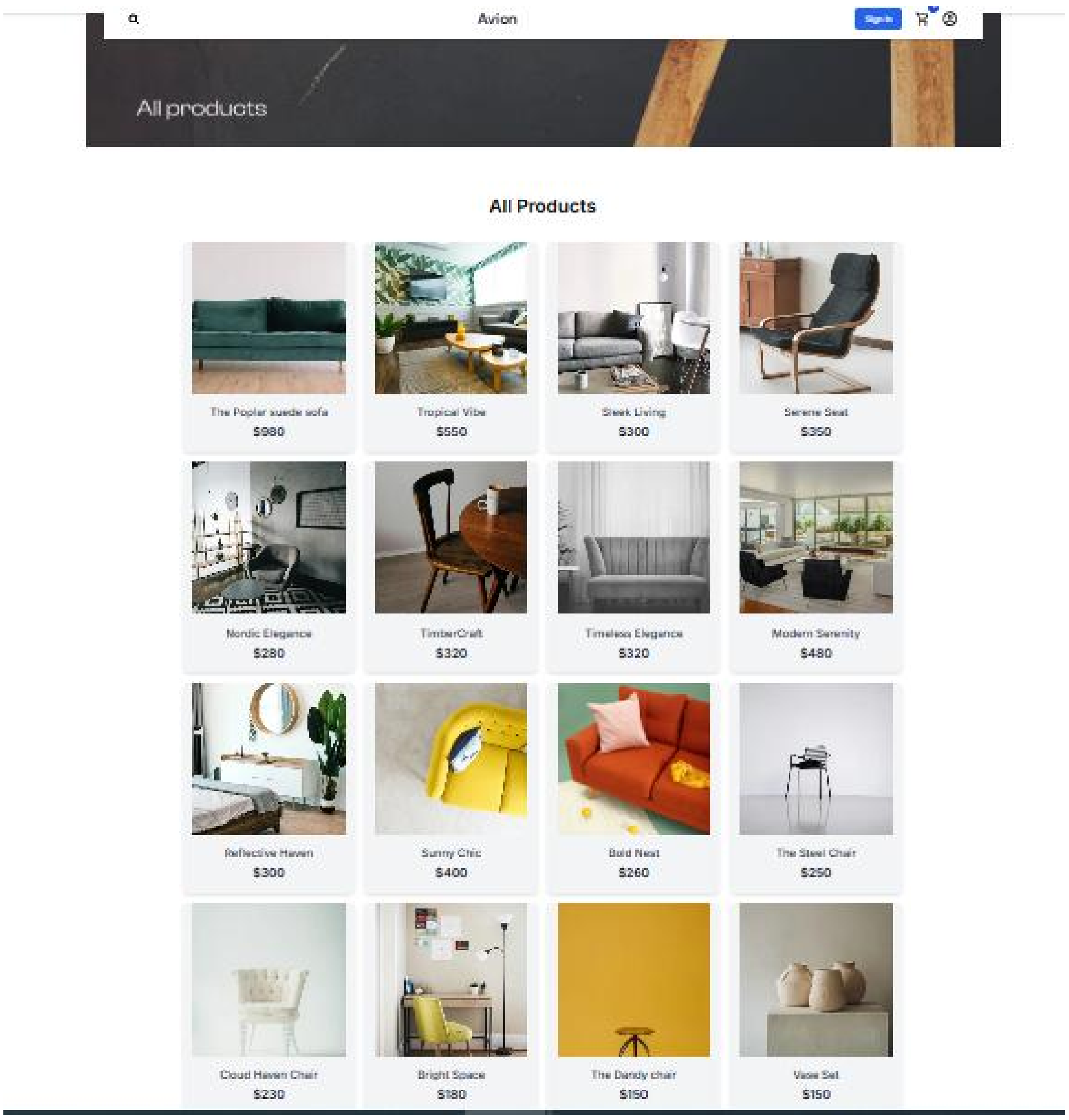
- A Next.js API route (`/api/products`) was set up to fetch data from Sanity and return it as JSON.

Fetching Data on the Frontend:

- The `getServerSideProps` function was used in a Next.js page to fetch data from the API route.
- Data was then passed as props to the component.

Rendering Data on the Browser:

- The fetched product data was mapped to display product cards with images, names, and prices.
- Tailwind CSS was used for styling.



Conclusion

The API integration for the furniture project was successfully completed using the mock API, with data seamlessly imported into Sanity CMS. Custom functions were implemented for image handling and inventory referencing, ensuring data consistency across the platform. Schema adjustments were made to properly link product data, images, and inventories. With these steps, the system is now well-prepared for efficient product and inventory management, setting the stage for future development and scalability.