Day 03: API Integration & Migration - Avion

Integration Workflow and Migration Summary

API Connection Steps:

1. Linking Product and Category Endpoints:

- Endpoints Used:
 - Product Data: hackathon-apis.vercel.app/api/products
 - Category Data: hackathon-apis.vercel.app/api/products
- Utilized fetch for data retrieval in the custom migration script.
- o Validated API responses to ensure they matched the schema fields in Sanity CMS.

2. Schema Refinement in Sanity:

- Addressed discrepancies between manual schema design and API response format.
- Updated schemas to ensure proper structure:
 - Product Schema Adjustments: Added fields like categories and tags.
 - Category Schema Modifications: Introduced an image as a reference field for handling category visuals.

3. Building the Migration Utility:

- o Created migrate.mjs for automated data migration.
- o Configured the Sanity client to manage image uploads and save data efficiently.
- Explicitly defined .env.local to resolve environmental variable conflicts.

Migration Process Overview:

1. Sanity Platform Installation:

Successfully installed and initialized Sanity.

2. Schema Development:

Designed product.ts schema matching API data.

3. Environment Configuration:

- o Created an .env file to securely store Sanity credentials.
- o Defined the .env file path explicitly within the migration script.

4. Custom Migration Script:

- o Developed migrate.mjs to fetch and upload data to Sanity CMS.
- o Mapped API fields accurately to resolve inconsistencies.
- o Added the following command in package.json for execution:

5. Executing the Migration:

Verified successful integration of product datasets in Sanity CMS.

Implementation Evidence:

1. Sanity CMS Dashboard View:

Displays populated product entities.

2. Application Frontend Display:

o Renders live data seamlessly on the user interface.

[&]quot;migrate": "node migrate.mjs"

Obstacles Addressed:

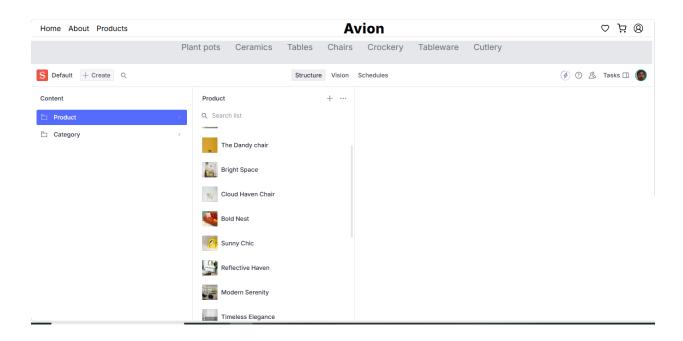
- Resolved .env.local file conflicts by explicitly declaring its path.
- Modified Sanity schemas to accommodate new data fields.
- Ensured correct image uploads and category-to-product mapping.

Strategic Choices:

- Opted for .env configuration to ensure secure and consistent credential management.
- Updated schema designs to harmonize API data with Sanity requirements.

```
import { defineType, defineField } from "sanity"
export const product = defineType({
    name: "product",
title: "Product",
type: "document",
     fields: [
          defineField({
              name:"category",
title:"Category",
type:"reference",
                    type: "category"
          defineField({
              name: "name",
title: "Title",
               validation: (rule) => rule.required(),
              type: "string"
          defineField({
              name: "slug",
title: "Slug",
               validation: (rule) => rule.required(),
               type: "slug"
          defineField({
              name: "image",
type: "image",
validation: (rule) => rule.required(),
```

Product Schema In Sanity:



New Ceramics



The Dandy Chai £250



£155



The Silky Vase £125



The Lucy Lamp £399

View Collection