# Day 3

# API Integration and Data Migration Report

[Furniture  Marketplace (Chairs) ]

---

## 1. API Integration Process

**Overview of API Integration:**

- **Purpose:** The objective was to integrate external or internal APIs into the application to facilitate seamless communication and data exchange.
- **Tools Used:**
  - REST APIs
  - Postman for API testing
  - Axios for making HTTP requests in the code

**Process Steps:**

1. **Understanding API Requirements:**
   - Analysis of the API documentation.
   - Identifying key endpoints for integration.
   - Determining authentication methods and headers.
2. **Making HTTP Requests:**
   - Using Axios or Fetch for making GET, POST, PUT, DELETE requests.
   - Handling API responses, including successful and error scenarios.
3. **Data Mapping and Transformation:**
   - Mapping data from the API response to the application's required format.
   - Transforming data to ensure compatibility with front-end components.
4. **Error Handling and Debugging:**
   - Implementing proper error handling mechanisms to deal with API failures.
   - Logging errors and using Postman for debugging API responses.
5. **Testing and Optimization:**
   - Thorough testing to ensure all API endpoints are integrated correctly.
   - Performance optimization for faster data loading times.

**Challenges Faced:**

- API rate-limiting and authentication issues.
- Handling large datasets from API responses efficiently.

---

## 2. Adjustments Made to Schemas

**Overview of Schema Adjustments:**

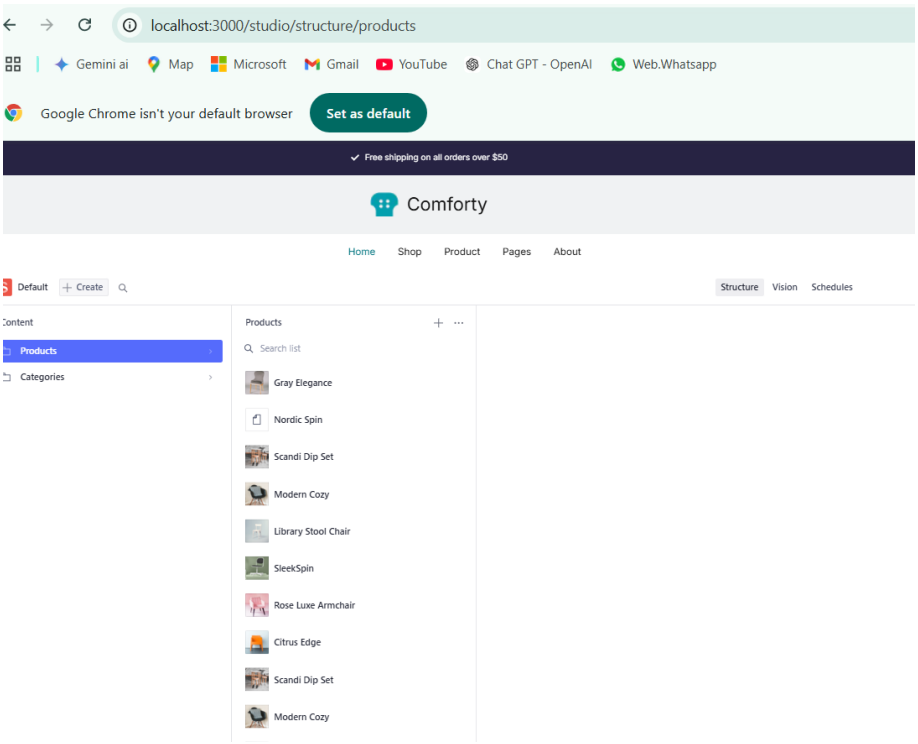
- Schema adjustments were necessary to accommodate new data structures coming from integrated APIs.

**Adjustments Details:**

1. **Modification of Data Models:**
    - Adding new fields to existing schemas to store additional data.
    - Adjusting relationships between data models to reflect new API integrations.
2. **Field Type Adjustments:**
    - Updating field types to ensure compatibility with API responses (e.g., converting strings to arrays or objects).
3. **Validation Rules:**
    - Enhancing schema validation to account for new data types and values coming from APIs.
    - Ensuring mandatory fields and constraints are properly enforced.
4. **Versioning:**
    - Implementing schema versioning to ensure backward compatibility with older data structures.

**Challenges Faced:**

- Schema conflicts between old and new data structures.
- Ensuring smooth transitions between schema versions during migration.


localhost:3000/studio/structure/products
Gemini ai
Map
Microsoft
Gmail
YouTube
Chat GPT - OpenAI
Web.Whatsapp
Google Chrome isn't your default browser
Set as default
Free shipping on all orders over $50
Comforty
Home
Shop
Product
Pages
About
Default
Create
Structure
Vision
Schedules
Content
Products
Categories
Products
Search list
Gray Elegance
Nordic Spin
Scandi Dip Set
Modern Cozy
Library Stool Chair
SleekSpin
Rose Luxe Armchair
Citrus Edge
Scandi Dip Set
Modern Cozy

```
You, 2 days ago | 1 author (You)
import { defineType } from "sanity";

export const productSchema = defineType({
  name: "products",
  title: "Products",
  type: "document",
  fields: [
    {
      name: "title",
      title: "Product Title",
      type: "string",       You, 2 days ago • first comm
    },
    {
      name: "price",
      title: "Price",
      type: "number",
    },
    {
      title: "Price without Discount",
      name: "priceWithoutDiscount",
      type: "number",
    },
    {
      name: "badge",
      title: "Badge",
      type: "string",
    },
    {
      name: "image",
      title: "Product Image",
      type: "image",
    },
    {
      name: "category",
      title: "Category",
      type: "reference",
```

```
sanity > schemaTypes > TS index.ts > [⊘] schema
       You, 2 days ago | 1 author (You)
1      import { type SchemaTypeDefinition } from "sanity";
2      import { productSchema } from "./products";
3      import { categorySchema } from "./categories";
4
5      export const schema: { types: SchemaTypeDefinition[] } = {
6        types: [productSchema, categorySchema],       You, 2 days ago •
7      };
8
```

---

## 3. Migration Steps and Tools Used

**Overview of Data Migration:**

- Data migration was performed to update the existing system with new data formats and values obtained from the integrated APIs.

**Migration Steps:**

1. **Pre-Migration Planning:**
    - Identifying the data to be migrated.
    - Mapping old data structures to new schema structures.
2. **Migration Execution:**
    - Using tools like **Knex.js** or **Sequelize** to migrate data from old tables to new ones.
    - Writing custom migration scripts to handle complex data transformations.
3. **Data Validation:**
    - Verifying the integrity of the migrated data.
    - Running consistency checks to ensure no data loss or corruption.
4. **Post-Migration Testing:**
    - Running tests on the migrated data to ensure everything works as expected.
    - Confirming that the API integrations are still functional post-migration.

**Tools Used for Migration:**

- **Database Migration Tools:** Knex.js, Sequelize
- **Data Backup Tools:** AWS RDS Snapshots, MongoDB Backup (for NoSQL migrations)
- **Data Transformation Tools:** Custom scripts, ETL frameworks (Extract, Transform, Load)

**Challenges Faced:**

- Migrating large datasets with minimal downtime.
- Handling data inconsistencies during the migration process.

page.tsx ...\product M | page.tsx ...\[id] U | migrate.mjs

scripts > migrate.mjs > ...

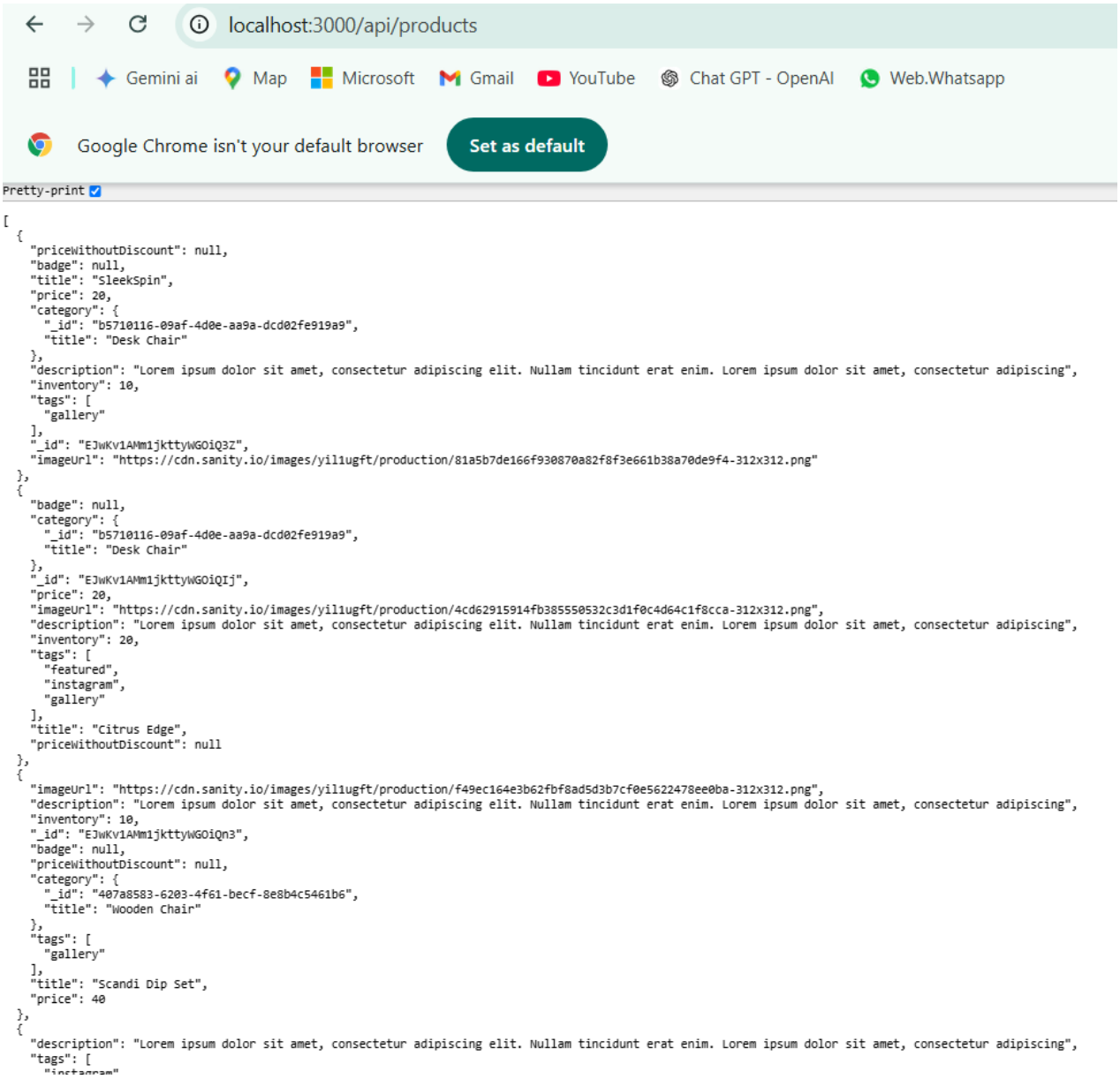
You, 2 days ago | 1 author (You)

```
// Import environment variables from .env.local
import "dotenv/config";

// Import the Sanity client to interact with the Sanity backend
import { createClient } from "@sanity/client";

// Load required environment variables
const {
  NEXT_PUBLIC_SANITY_PROJECT_ID, // Sanity project ID
  NEXT_PUBLIC_SANITY_DATASET, // Sanity dataset (e.g., "production")
  NEXT_PUBLIC_SANITY_AUTH_TOKEN, // Sanity API token
  BASE_URL = "https://giaic-hackathon-template-08.vercel.app", // API base URL for products and cat
} = process.env;

// Check if the required environment variables are provided
if (!NEXT_PUBLIC_SANITY_PROJECT_ID || !NEXT_PUBLIC_SANITY_AUTH_TOKEN) {
  console.error("Missing required environment variables. Please check your .env.local file.");
  process.exit(1); // Stop execution if variables are missing
}

// Create a Sanity client instance to interact with the target Sanity dataset
const targetClient = createClient({
  projectId: NEXT_PUBLIC_SANITY_PROJECT_ID, // Your Sanity project ID
  dataset: NEXT_PUBLIC_SANITY_DATASET || "production", // Default to "production" if not set
  useCdn: false, // Disable CDN for real-time updates
  apiVersion: "2023-01-01", // Sanity API version
  token: NEXT_PUBLIC_SANITY_AUTH_TOKEN, // API token for authentication
});

// Function to upload an image to Sanity
Tabnine | Edit | Test | Explain | Document
async function uploadImageToSanity(imageUrl) {
  try {
    // Fetch the image from the provided URL
    const response = await fetch(imageUrl);
    if (!response.ok) throw new Error(`Failed to fetch image: ${imageUrl}`);

```

page.tsx ...\product M | page.tsx ...\[id] U | migrate.mjs

scripts > migrate.mjs > ...

```
async function migrateData() {

  try {
    // Fetch categories from the REST API
    const categoriesResponse = await fetch(`${BASE_URL}/api/categories`);
    if (!categoriesResponse.ok) throw new Error("Failed to fetch categories.");
    const categoriesData = await categoriesResponse.json(); // Parse response to JSON

    // Fetch products from the REST API
    const productsResponse = await fetch(`${BASE_URL}/api/products`);
    if (!productsResponse.ok) throw new Error("Failed to fetch products.");
    const productsData = await productsResponse.json(); // Parse response to JSON

    const categoryIdMap = {}; // Map to store migrated category IDs

    // Migrate categories
    for (const category of categoriesData) {
      console.log(`Migrating category: ${category.title}`);
      const imageId = await uploadImageToSanity(category.imageUrl); // Upload category image

      // Prepare the new category object
      const newCategory = {
        _id: category._id, // Use the same ID for reference mapping
        _type: "categories",
        title: category.title,
        image: imageId ? { _type: "image", asset: { _ref: imageId } } : undefined, // Add image
      };

      // Save the category to Sanity
      const result = await targetClient.createOrReplace(newCategory);
      categoryIdMap[category._id] = result._id; // Store the new category ID
      console.log(`Migrated category: ${category.title} (ID: ${result._id})`);
    }

    // Migrate products
    for (const product of productsData) {
      console.log(`Migrating product: ${product.title}`);
      const imageId = await uploadImageToSanity(product.imageUrl); // Upload product image
```





```
[
  {
    "priceWithoutDiscount": null,
    "badge": null,
    "title": "SleekSpin",
    "price": 20,
    "category": {
      "_id": "b5710116-09af-4d0e-aa9a-dcd02fe919a9",
      "title": "Desk Chair"
    },
    "description": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam tincidunt erat enim. Lorem ipsum dolor sit amet, consectetur adipiscing",
    "inventory": 10,
    "tags": [
      "gallery"
    ],
    "_id": "EJwKv1AMm1jkttyWGOiQ3Z",
    "imageUrl": "https://cdn.sanity.io/images/yil1ugft/production/81a5b7de166f930870a82f8f3e661b38a70de9f4-312x312.png"
  },
  {
    "badge": null,
    "category": {
      "_id": "b5710116-09af-4d0e-aa9a-dcd02fe919a9",
      "title": "Desk Chair"
    },
    "_id": "EJwKv1AMm1jkttyWGOiQIj",
    "price": 20,
    "imageUrl": "https://cdn.sanity.io/images/yil1ugft/production/4cd62915914fb385550532c3d1f0c4d64c1f8cca-312x312.png",
    "description": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam tincidunt erat enim. Lorem ipsum dolor sit amet, consectetur adipiscing",
    "inventory": 20,
    "tags": [
      "featured",
      "instagram",
      "gallery"
    ],
    "title": "Citrus Edge",
    "priceWithoutDiscount": null
  },
  {
    "imageUrl": "https://cdn.sanity.io/images/yil1ugft/production/f49ec164e3b62fbf8ad5d3b7cf0e5622478ee0ba-312x312.png",
    "description": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam tincidunt erat enim. Lorem ipsum dolor sit amet, consectetur adipiscing",
    "inventory": 10,
    "_id": "EJwKv1AMm1jkttyWGOiQn3",
    "badge": null,
    "priceWithoutDiscount": null,
    "category": {
      "_id": "407a8583-6203-4f61-becf-8e8b4c5461b6",
      "title": "Wooden Chair"
    },
    "tags": [
      "gallery"
    ],
    "title": "Scandi Dip Set",
    "price": 40
  },
  {
    "description": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam tincidunt erat enim. Lorem ipsum dolor sit amet, consectetur adipiscing",
    "tags": [
      "instagram"
```


---

## Conclusion:

- The API integration was successfully completed with all necessary adjustments to the schemas and a smooth data migration process. The system is now equipped to handle new API data sources and provide real-time data integration seamlessly.

---

Made by Chandafatiama$mylife