

API Integration and Data Migration Report

Prepared by: Ammad Ali

Date: 18-01-2025

1. Introduction

This report documents the process of integrating APIs and migrating data into Sanity.io using Next.js. The objective was to build a functional marketplace backend that fetches and displays data dynamically on the user interface. By following real-world API integration techniques, this project enhances the ability to manage headless CMS-based applications efficiently.

2. Objectives

The main goals of this project include:

Integrating Sanity.io API into a Next.js project.

Migrating and structuring data within Sanity CMS.

Ensuring schema validation and data consistency.

Rendering the migrated data dynamically on the frontend.

Handling errors and ensuring smooth API communication.

3. API Overview

This project uses Sanity.io, a headless CMS, to manage and retrieve content efficiently. The API allows fetching product data, categories, and other essential details for the marketplace.

4. Data Migration Process

4.1 Schema Validation and Adjustments

The Sanity schema was analyzed and modified to ensure compatibility with the API data.

Adjustments were made to maintain consistency across the system.

4.2 Data Migration Methods

The migration process was executed in three steps:

1. Automated API Integration

Fetches and structures data using Next.js API utility functions.

Verifies API responses before insertion.

2. Manual Import (Optional Approach)

Data was structured in JSON format and uploaded manually for testing.

3. Handling API Requests & Data Transformation

Used Next.js API Routes to fetch and process API data.

Ensured proper field mapping for smooth data flow.

5. API Integration in Next.js

5.1 Utility Functions

Created reusable API functions for fetching and displaying data efficiently.

5.2 Rendering Data in the Frontend

Successfully displayed API-fetched data on the UI.

Used Next.js dynamic rendering for real-time updates.

5.3 Testing API Responses

Used Postman and Browser DevTools to validate API responses.

Ensured that all API calls returned correct data.

6. Challenges Faced

I did, however, face some challenges, but I believe that changes are part of success. The key challenges and solutions included:

Schema mismatches → Adjusted API field mapping.

Slow API response → Implemented caching techniques.

Data inconsistency issues → Added validation before inserting data.

7. Error Handling & Optimization

Implemented proper error handling to prevent UI crashes.

Logged API errors for debugging and monitoring.

Used fallback data to improve user experience.

8. Final Output

After completing the integration, the API data was successfully fetched and displayed on the frontend. The marketplace backend is now fully functional with live data updates.

9. Best Practices Followed

- ✓ .env files were used to secure API keys.
- ✓ Clean Code Practices were followed (modular functions, meaningful variable names).
- ✓ Schema Validation ensured data consistency.
- ✓ Version Control (Git) was used for tracking changes.
- ✓ Testing and Debugging with Postman and Browser DevTools.

10. Conclusion

Finally, I successfully integrated the API, and the data is now displaying on the user interface. The project improved my skills in API handling, data migration, and Next.js integration. This experience has strengthened my ability to work with headless CMS solutions and real-world API-based applications.

Next Steps

Further optimize API requests to improve performance.

Implement additional security measures for better data protection.

Explore new CMS features to enhance functionality.

11. Code Snippets & Screenshots

```
create-sanity@3.74.1
Ok to proceed? (y) y
```

```
✓ You are logged in as starxammad@gmail.com using GitHub
✓ Fetching existing projects
```

Your content will be stored in a dataset that can be public or private, depending on whether you want to query your content with or without authentication. The default dataset configuration has a public dataset named "production".

✓ Creating dataset

```
? Select project template to use Blog (schema)
```

```
Added http://localhost:3000 to CORS origins
```

```
added 911 packages, changed 6 packages, and audited 1274 packages in 3m
```

found 0 vulnerabilities

```
243 packages are looking for funding
  run `npm fund` for details
```

```
Success! Your Sanity configuration files has been added to this project
PS D:\nextjs\LagGayeBoss\HacthonQ2-main\my-app>
```

AmmadAli1122

hackathon3

HA

hackathon3

PLAN
Growth Trial

STATUS
Active

PROJECT ID
q937fv81

Getting started

Overview

Members

Studios

Datasets

Access

Activity

Usage

Plan

API

Settings

Webhooks

CORS origins

Tokens

GROQ-powered webhooks


HTTP callbacks to a given URL triggered by changes in your content lake

[Learn more about webhooks](#)

+ Create webhook

0 of 2 webhooks
(2 included in plan)



[Get more webhooks](#)



There are no GROQ-powered webhooks in this project
Maybe try creating a new webhook?

CORS origins

Hosts that can connect to the project API.

ORIGIN	CREDENTIALS	CREATED	
http://localhost:3000	Allowed	7 minutes	
http://localhost:3333	Allowed	8 minutes	

+ Add CORS origin

Webhooks

CORS origins

Tokens

CORS origins

Tokens

Hosts that can connect to the project API.

ORIGIN	CREDENTIALS	CREATED	
http://localhost:3000	Allowed	8 minutes	
http://localhost:3333	Allowed	9 minutes	

Tokens

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

NAME	PERMISSIONS	CREATED	
Employee import	Developer	4 minutes	

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

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXZWQ0a6j6ENgbG82Vl0PKF5FFTD3FF5tS08wlg8DsRC07Fc5lYqzeGxtfWRjPvNZVoxal0q5ia0...n96LmGS3x7Rn9ZJwL3yzYxHa3Kg5VaYRyeXtuH2xVeA3khfx2fwdxexi00i86gano

my-app > src > sanity > schemasTypes > product > product

1 import { defineType } from "sanity"22 export const product = defineType({34 name: "product",5 titles: "Product",6 type: "document",7 fields: {89 {10 name: "title",11 title: "Title",12 validation: (rule) => rule.required(),13 type: "string",14},15 {16 name: "description",17 title: "Description",18 type: "text",19 validation: (rule) => rule.required(),20 title: "Description",21},22 {23 name: "productImage",24 type: "image",25 validation: (rule) => rule.required(),26 title: "Product Image",27},28 {29 name: "price",30 type: "number",31 validation: (rule) => rule.required(),32 title: "Price",33},34 {35 name: "tags",36 type: "array",37 title: "Tags",38 of: [{ type: "string" }]39}40}41}42}43}44}45}46}47}48}49}50}51}52}53}54}55}56}57}58}59}60}61}62}63}64}65}66}67}68}69}70}71}72}73}74}75}76}77}78}79}80}81}82}83}84}85}86}87}88}89}90}91}92}93}94}95}96}97}98}99}100}

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

✓ You are logged in as starxamxam@gmail.com using GitHub
✓ Fetching existing projects
? Create a new project or select an existing one Create new project
? Your project name: hackyHops
Your content will be stored in a dataset that can be public or private, depending on

my-app > package.json > ...

1 {2 "name": "my-app",3 "version": "0.1.0",4 "private": true,5 "type": "module",67 "scripts": {8 "dev": "next dev",9 "build": "next build",10 "start": "next start",11 "lint": "next lint",12 "import-data": "node script/importData.js"13 },14 "dependencies": {

