

Day 3 - API Integration Report - [EliteBuy - Ecommerce]

API Integration Process:

1. Overview

In this project, I integrated APIs for handling products, orders, and payments using Next.js, Sanity CMS, and Stripe. The integration aimed to ensure seamless data flow between the frontend, backend, and database.

2. Steps Followed

- Configured Sanity CMS for product and order management.
- Integrated Stripe API for payment processing.
- Implemented API routes in Next.js for handling requests.
- Utilized ``getServerSideProps`` and ``fetch`` for data fetching.
- Secured API requests using authentication and environment variables.

3. Challenges & Solutions

- Issue: CORS error while fetching data.
- Solution: Configured CORS settings in API endpoints.

- Issue: Payment failure due to incorrect Stripe keys.
- Solution: Verified and updated environment variables.

Adjustments Made to Schemas:

1. Sanity Schema Modifications

- Product Schema:
 - Added `price`, `image`, and `stock` fields.
 - Defined references for categories.
- Order Schema:
 - Included `user`, `products`, `totalAmount`, and `paymentStatus` fields.
 - Integrated Stripe payment tracking.

2. Next.js API Route Adjustments

- Created API endpoints for fetching products (`/api/products`).
- Developed a checkout API (`/api/checkout`) that interacts with Stripe.
- Implemented order tracking API (`/api/orders`).

Migration Steps and Tools Used:

1. Data Migration Process

- Exported existing data from local JSON to Sanity CMS.
- Used `sanity import` CLI for batch data migration.
- Verified data integrity post-migration.

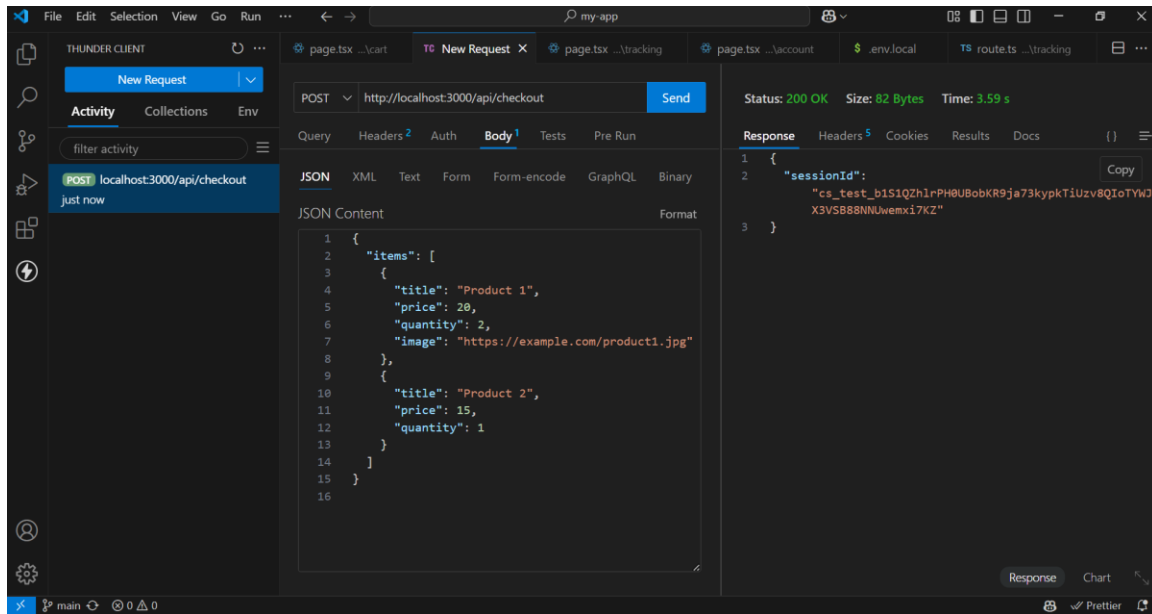
2. Tools Utilized

- **Sanity CLI** for schema modifications.
- **Postman** for API testing.
- **Stripe Dashboard** for transaction validation.
- ***Next.js API Routes*** for handling backend operations.

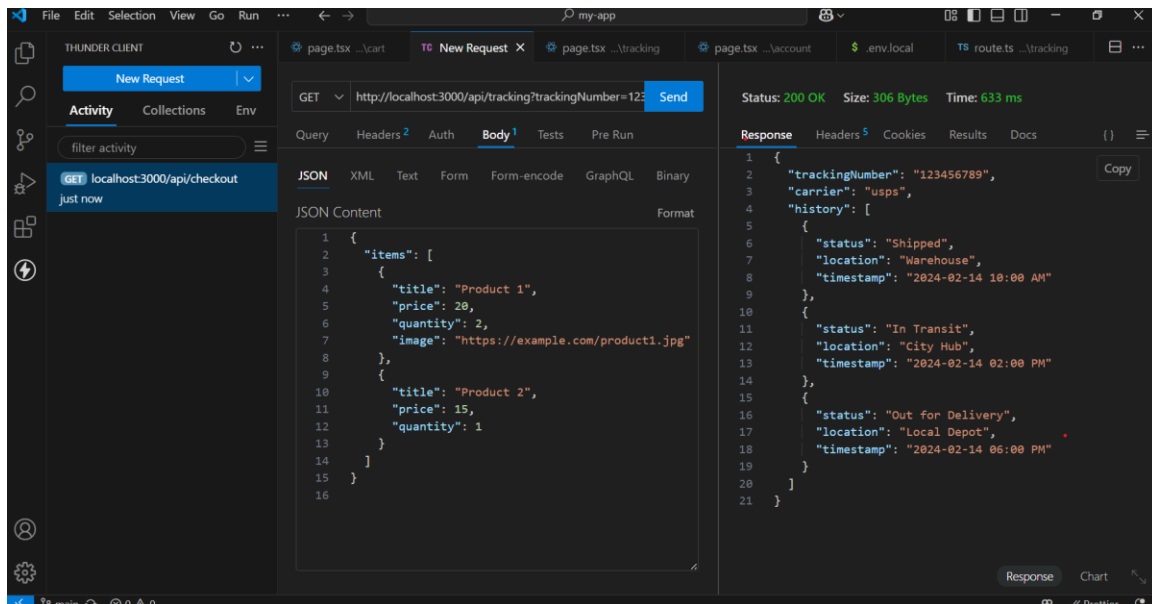
Screenshots:

1. API Calls (ThunderClient)

checkout api.



Order Tracking API.




2. Data Displayed in Frontend

EliteBuy

[Home](#)[Shop](#)[Contact](#)[Tracking](#)


Search products...

Our Products




iPhone 15 Pro Max
\$250
Brand: iPhone 15 Pro Max
Storage: 128 gb
RAM: 4 gb

[View Details](#)




iPhone XR
\$99
Brand: iPhone-Xr
Storage: 64 gb
RAM: 3

[View Details](#)



iPhone 13
\$130
Brand: iPhone 13
Storage: 128 gb
RAM: 4 gb

[View Details](#)



OnePlus 11
\$120
Brand: OnePlus 11
Storage: 128 gb
RAM: 8 gb


[View Details](#)

localhost:3000/product/oneplus-11

EliteBuy


[Home](#)[Shop](#)[Contact](#)[Tracking](#)

Search products...




Asus Zenfone 9
\$100
Brand: Asus Zenfone 9
Storage: 128 gb
RAM: 12

[View Details](#)




Xiaomi Redmi Note 12
\$120
Brand: Xiaomi Redmi Note 12
Storage: 512 gb
RAM: 8 gb

[View Details](#)




Google Pixel 8
\$140
Brand: Google Pixel 8
Storage: 128 gb
RAM: 8 gb

[View Details](#)



Motorola Edge+
\$110
Brand: Motorola Edge+
Storage: 64 gb
RAM: 4 gb


[View Details](#)



EliteBuy

[Home](#)[Shop](#)[Contact](#)[Tracking](#)

Search products...



Google Pixel 8

Brand: Google Pixel 8

Storage: 128 gb

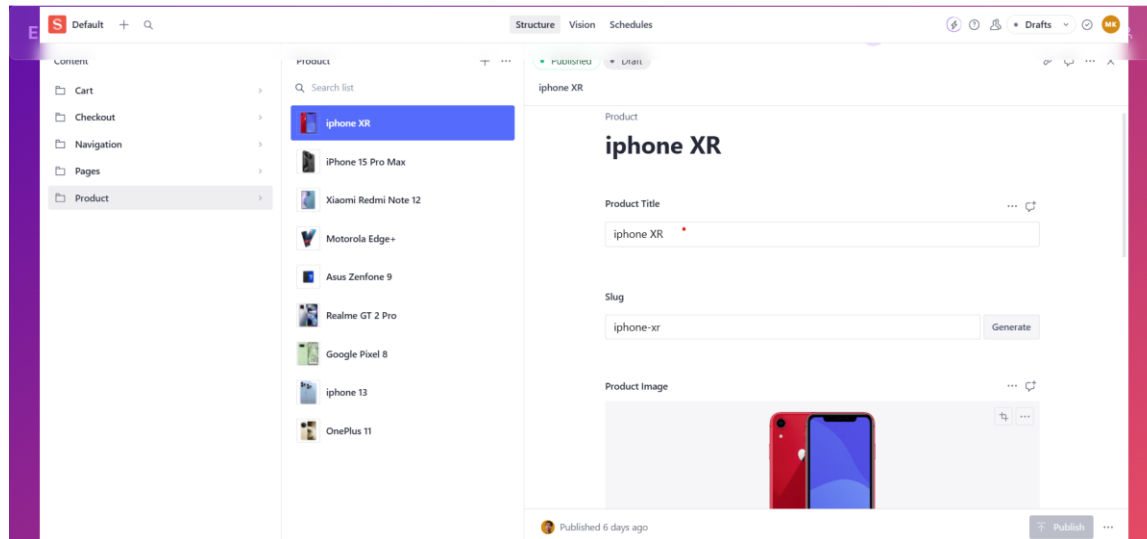
RAM: 8 gb

\$140.00

Category: Electronics

[Add to Cart](#)

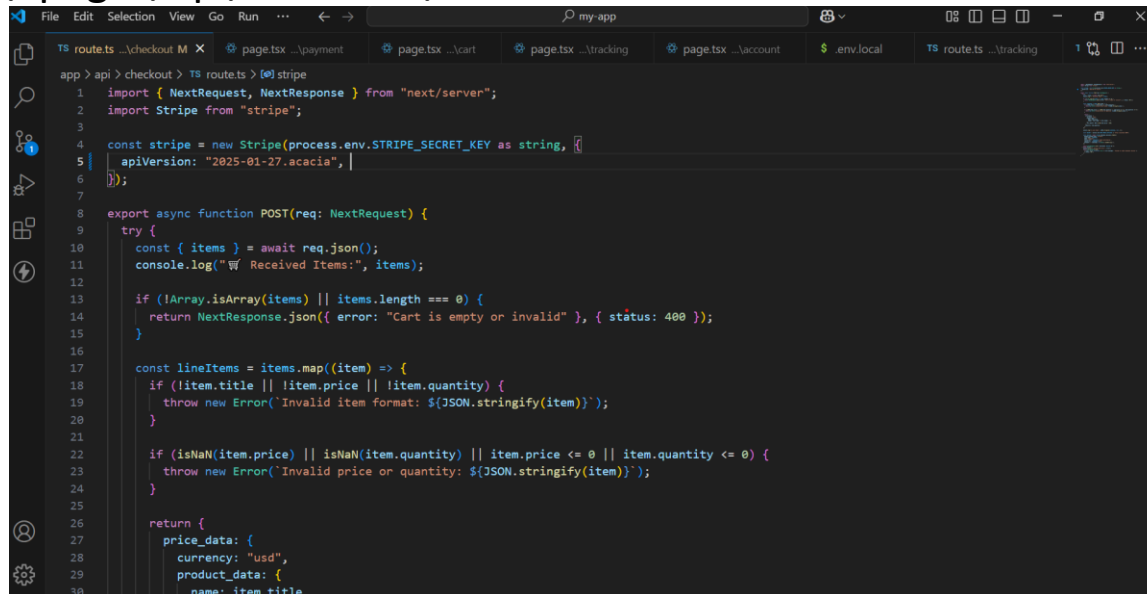
3. Populated Sanity CMS Fields



Code Snippets:

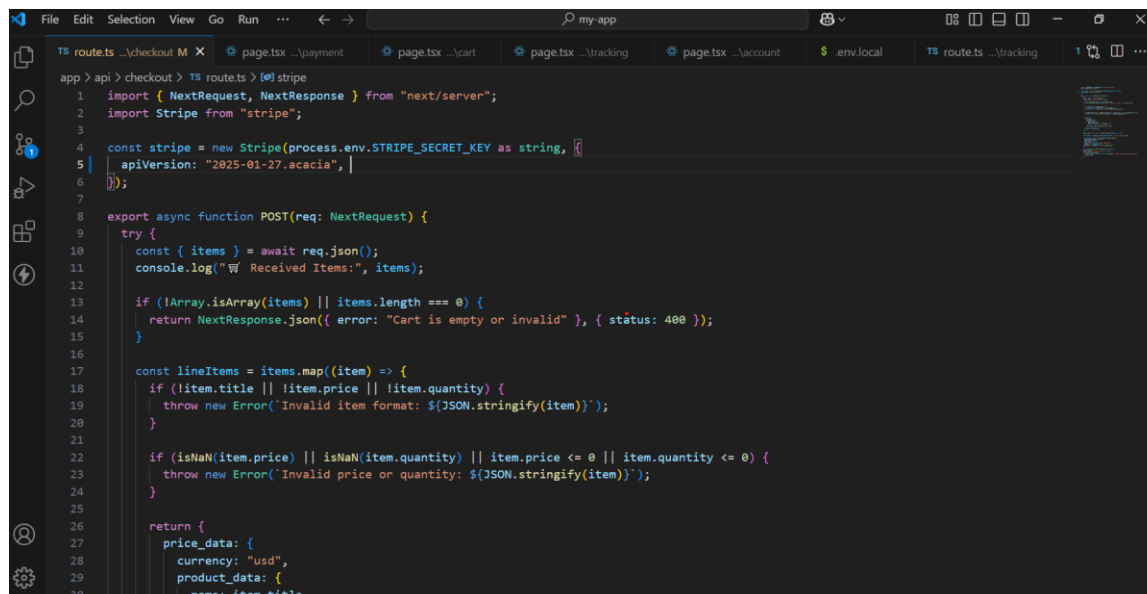
1. API Integration

/pages/api/checkout/route.ts



```
1 import { NextRequest, NextResponse } from "next/server";
2 import Stripe from "stripe";
3
4 const stripe = new Stripe(process.env.STRIPE_SECRET_KEY as string, {
5   apiVersion: "2025-01-27.acacia",
6 });
7
8 export async function POST(req: NextRequest) {
9   try {
10     const { items } = await req.json();
11     console.log("Received Items:", items);
12
13     if (!Array.isArray(items) || items.length === 0) {
14       return NextResponse.json({ error: "Cart is empty or invalid" }, { status: 400 });
15     }
16
17     const lineItems = items.map((item) => {
18       if (!item.title || !item.price || !item.quantity) {
19         throw new Error(`Invalid item format: ${JSON.stringify(item)}`);
20       }
21
22       if (isNaN(item.price) || isNaN(item.quantity) || item.price <= 0 || item.quantity <= 0) {
23         throw new Error(`Invalid price or quantity: ${JSON.stringify(item)}`);
24       }
25
26       return {
27         price_data: {
28           currency: "usd",
29           product_data: {
30             name: item.title,
```

2. Stripe Payment Integration



```
1 import { NextRequest, NextResponse } from "next/server";
2 import Stripe from "stripe";
3
4 const stripe = new Stripe(process.env.STRIPE_SECRET_KEY as string, {
5   apiVersion: "2025-01-27.acacia",
6 });
7
8 export async function POST(req: NextRequest) {
9   try {
10     const { items } = await req.json();
11     console.log("Received Items:", items);
12
13     if (!Array.isArray(items) || items.length === 0) {
14       return NextResponse.json({ error: "Cart is empty or invalid" }, { status: 400 });
15     }
16
17     const lineItems = items.map((item) => {
18       if (!item.title || !item.price || !item.quantity) {
19         throw new Error(`Invalid item format: ${JSON.stringify(item)}`);
20       }
21
22       if (isNaN(item.price) || isNaN(item.quantity) || item.price <= 0 || item.quantity <= 0) {
23         throw new Error(`Invalid price or quantity: ${JSON.stringify(item)}`);
24       }
25
26       return {
27         price_data: {
28           currency: "usd",
29           product_data: {
30             name: item.title,
```

Conclusion:

The API integration process was successfully completed, ensuring smooth communication between Sanity CMS, Next.js, and Stripe. The migration was performed efficiently, and the

website now dynamically handles products, orders, and payments.