

# DAY 3 - API INTEGRATION REPORT-[General E-commerce]

## a) API Integration Process Report

**[Muhammad Kamran]**

**[General E-commerce]**

**[Date]**

### 1. Introduction

This report summarizes the API integration process, covering key steps, challenges, and solutions.

### 2. Integration Process

#### 2.1 Key Steps

- Identify API requirements, endpoints, and authentication methods.
- Obtain and securely configure API access.
- Implement API requests using proper HTTP methods.
- Handle errors, rate limits, and logging.
- Optimize performance with caching and pagination.
- Secure API usage with HTTPS and authentication mechanisms.
- Test using Postman/curl and deploy with monitoring.

#### 2.2 Migration & Tools

- Assess and map old and new schemas.
- Develop and test migration scripts.
- Use tools like Postman, Swagger, AWS DMS, and database migration tools.

### 3. Challenges & Solutions

## DAY 3 - API INTEGRATION REPORT-[General E-commerce]

Challenge	Solution
Authentication Issues	Use correct API keys and guidelines.
Rate Limiting	Implement request throttling and caching.
Data Format Mismatch	Validate and adjust schemas as needed.
Downtime & Failures	Use retries, fallbacks, and monitoring.
Migration Complexity	Automate with scripts and thorough testing.

### 4. Conclusion

Effective API integration requires planning, security, and optimization. Schema adjustments and migration strategies ensure smooth transitions and data integrity.

### **b) Screenshots of:**

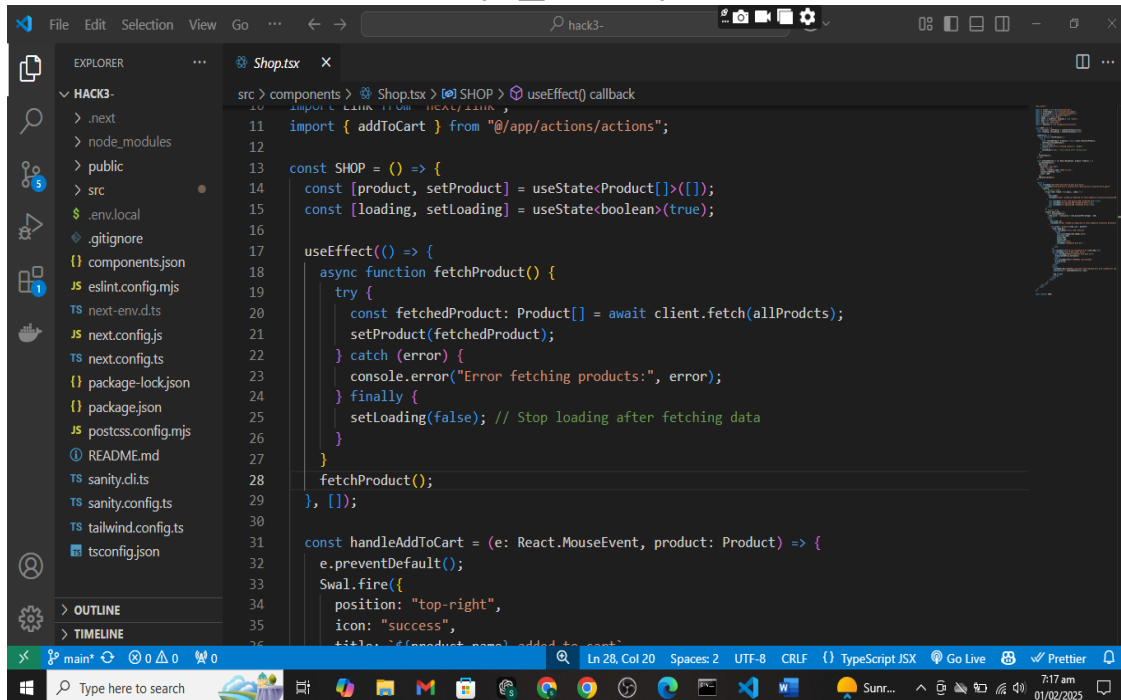
- . Api calls.
- . Data successfully displayed in Frontend
- . Populated Sanity CMS fields.

# DAY 3 - API INTEGRATION REPORT-[General E-commerce]

Screenshots of:

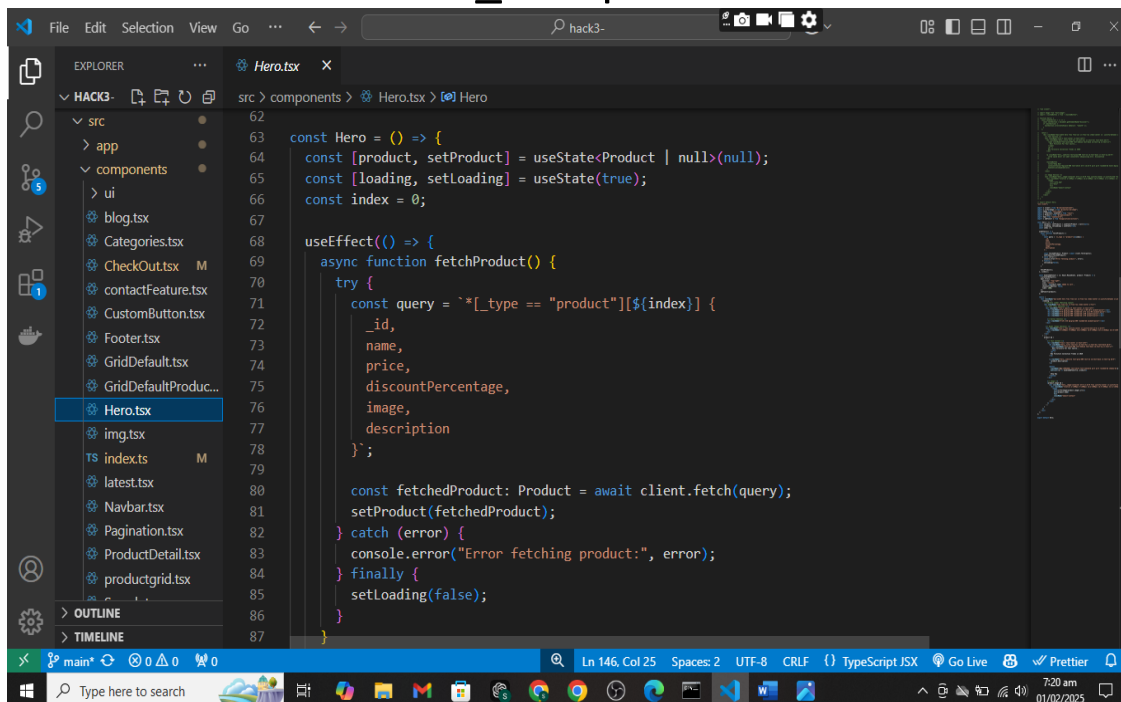
Api calls

## Shop\_ Component



```
src > components > Shop.tsx > [SHOP] > useEffect() callback
10 import Link from 'next/link';
11 import { addToCart } from '@app/actions/actions';
12
13 const SHOP = () => {
14   const [product, setProduct] = useState<Product[]>([]);
15   const [loading, setLoading] = useState<boolean>(true);
16
17   useEffect(() => {
18     async function fetchProduct() {
19       try {
20         const fetchedProduct: Product[] = await client.fetch(allProducts);
21         setProduct(fetchedProduct);
22       } catch (error) {
23         console.error("Error fetching products:", error);
24       } finally {
25         setLoading(false); // Stop loading after fetching data
26       }
27     }
28     fetchProduct();
29   }, []);
30
31   const handleAddToCart = (e: React.MouseEvent, product: Product) => {
32     e.preventDefault();
33     Swal.fire({
34       position: "top-right",
35       icon: "success",
36       title: `Product added to cart!`,
37     });
38   };
39 }
```

## Hero\_Component



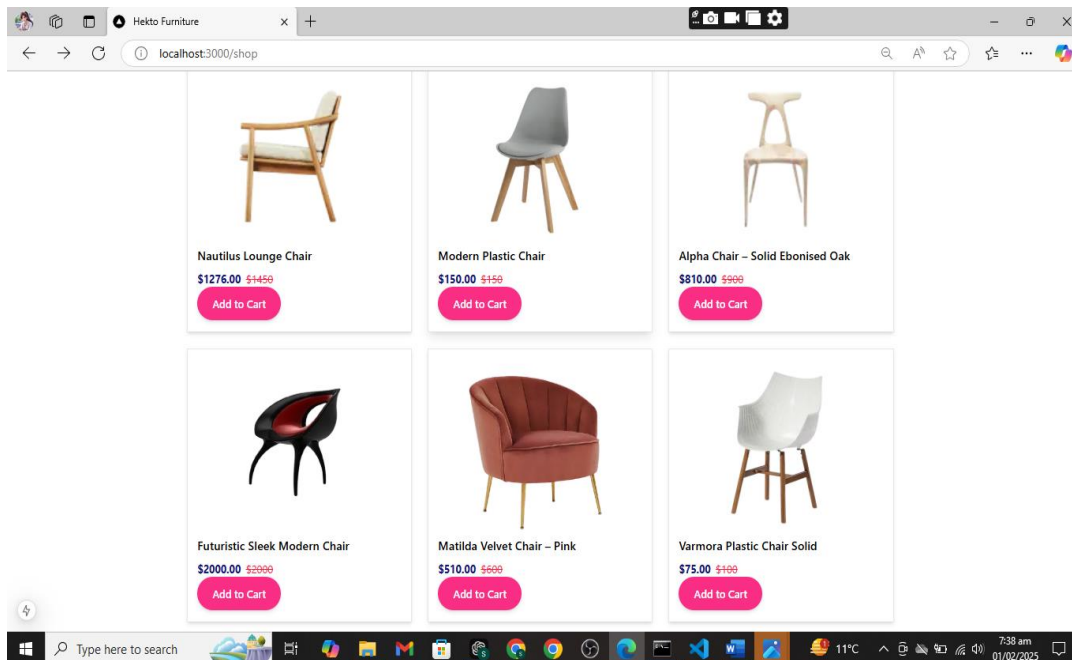
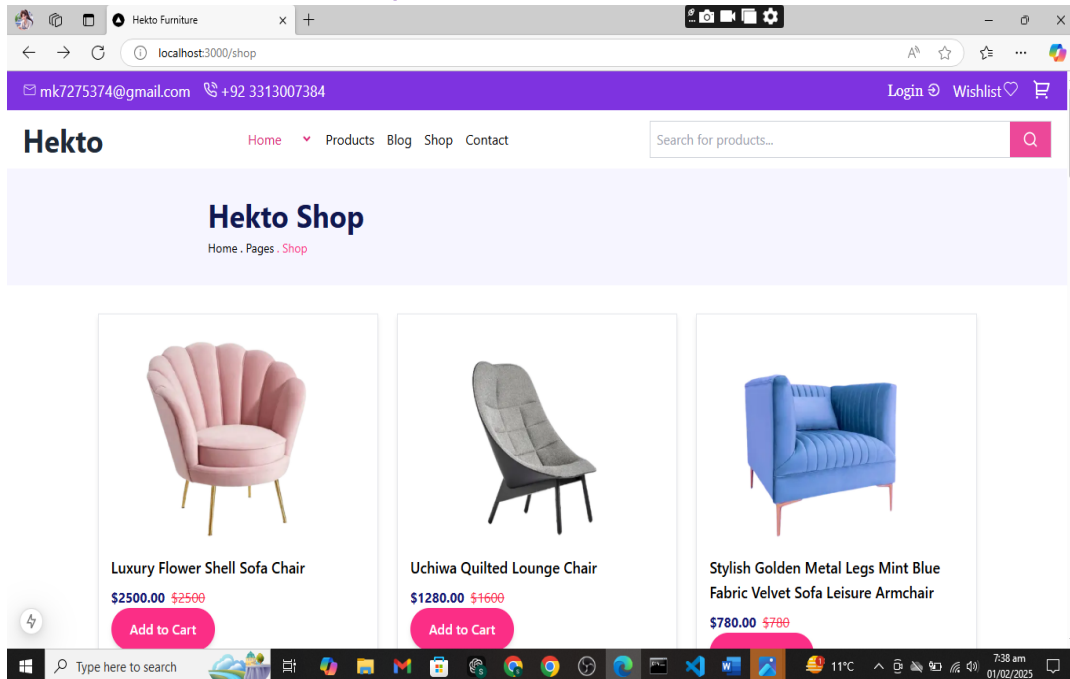
```
src > components > Hero.tsx > [Hero]
62
63 const Hero = () => {
64   const [product, setProduct] = useState<Product | null>(null);
65   const [loading, setLoading] = useState(true);
66   const index = 0;
67
68   useEffect(() => {
69     async function fetchProduct() {
70       try {
71         const query = `*[_type == "product"]${index} {
72           _id,
73           name,
74           price,
75           discountPercentage,
76           image,
77           description
78         }`;
79
80         const fetchedProduct: Product = await client.fetch(query);
81         setProduct(fetchedProduct);
82       } catch (error) {
83         console.error("Error fetching product:", error);
84       } finally {
85         setLoading(false);
86       }
87     }
88     fetchProduct();
89   }, [index]);
90 }
```

# DAY 3 - API INTEGRATION REPORT-[General E-commerce]

Screenshots of:

Data successfully Displayed in the frontend.

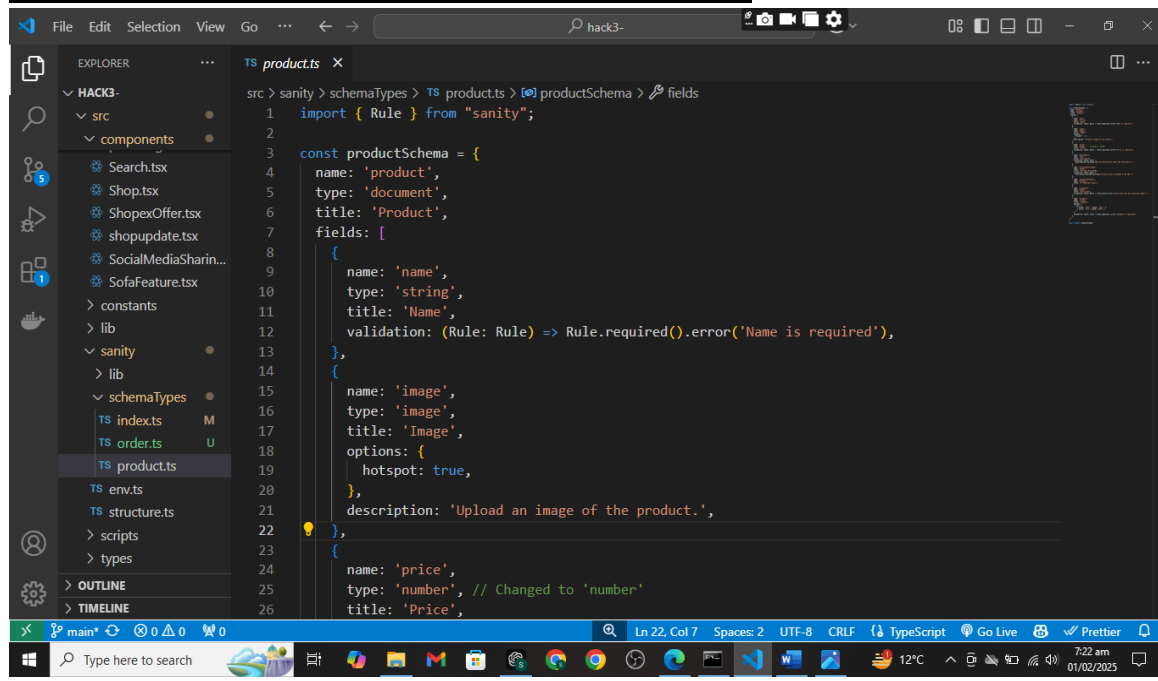
## Shop



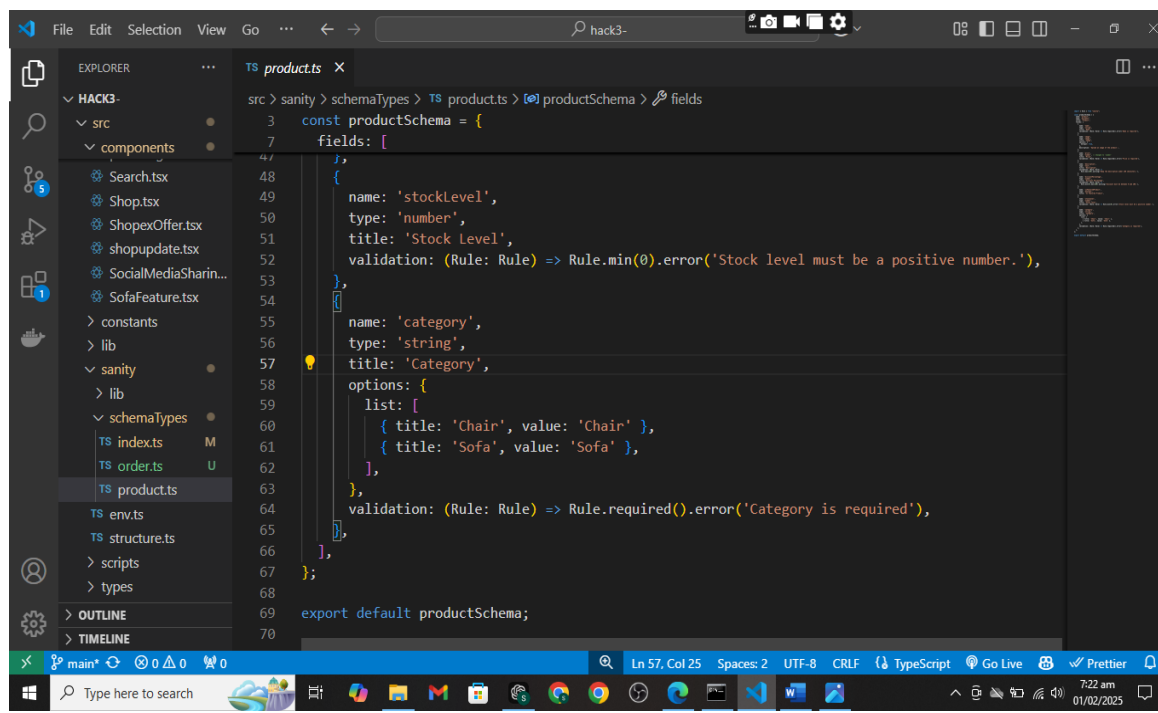
# DAY 3 - API INTEGRATION REPORT-[General E-commerce]

Screenshots of;

Populated Sanity CMS fields.



```
src > sanity > schemaTypes > TS product.ts > productSchema > fields
1  import { Rule } from "sanity";
2
3  const productSchema = {
4    name: 'product',
5    type: 'document',
6    title: 'Product',
7    fields: [
8      {
9        name: 'name',
10       type: 'string',
11       title: 'Name',
12       validation: (Rule: Rule) => Rule.required().error('Name is required'),
13     },
14     {
15       name: 'image',
16       type: 'image',
17       title: 'Image',
18       options: {
19         hotspot: true,
20       },
21       description: 'Upload an image of the product.',
22     },
23     {
24       name: 'price',
25       type: 'number', // Changed to 'number'
26       title: 'Price',
```



```
3  const productSchema = {
4    fields: [
48     {
49       name: 'stockLevel',
50       type: 'number',
51       title: 'Stock Level',
52       validation: (Rule: Rule) => Rule.min(0).error('Stock level must be a positive number.'),
53     },
54     {
55       name: 'category',
56       type: 'string',
57       title: 'Category',
58       options: {
59         list: [
60           { title: 'Chair', value: 'Chair' },
61           { title: 'Sofa', value: 'Sofa' },
62         ],
63       },
64       validation: (Rule: Rule) => Rule.required().error('Category is required'),
65     },
66   ],
67 };
68
69 export default productSchema;
```

# DAY 3 - API INTEGRATION REPORT-[General E-commerce]

## Data migration.

## *Data Successfully migrated to Sanity Studio*

