Day 3 - API Integration Report [www.WoodenAspire.com]

Objective.

The objective of Day 3 is to incorporate APIs and transfer data into Sanity CMS to develop a fully operational marketplace backend. This activity is designed to simulate real-world scenarios and equip participants with the skills needed to manage various client demands efficiently.

Significant Knowledge Outcomes.

1. Understand how to integrate APIs into a Next.js project.
2. Learn to migrate data into Sanity CMS.
3. Develop skills to validate schemas for seamless API integration.
4. Implement best practices in API error handling and schema adjustments.

Steps for Day 3:

1) API Integration Process.

API Endpoint Setup

● **API Endpoint:** https://template6-six.vercel.app/api/products.

● **Data Structure:** The API provided product data featuring the following fields:
Product name , description , category , price , colors , inventory , status , and Product image.

Error Handling: Ensure proper checks for errors like invalid parameters or missing data, and return clear messages to improve user experience.

2) Adjustment of Schema.

Field Updates:

● **Image:** Set up as a Sanity image field with hotspot functionality activated, allowing for cropping and scaling adjustments.

● **Tags:** Included as an optional array of strings to support multiple tag selections.

Sanity Schema.

```
export const product ={
    name: "product",
    title: "Product",
    type: "document",
    fields: [
        {
            name: "title",
            type: "string",
            title: "Title",
        },
        {
            name: "id",
            type: "number",
            title: "Id",
        },
        {
            name: "description",
            type: "text",
            title: "Description",
        },
        {
            name: "productImage",
```

```
            type: "image",
            title: "Product Image"
        },
        {
            name: "price",
            type: "number",
            title: "Price",
        },
        {
            name: "tags",
            type: "array",
            title: "Tags",
            of: [{ type: "string" }]
        },
        {
            name: "discount Percentage",
            type: "number",
            title: "Discount Percentage",
        },
        {
            name: "isNew",
            type: "boolean",
            title: "New Badge",
        }
    ]
}
```

3) Migration Steps.

1. Environment Setup:

○ Install dependencies: : @sanity/client,dotenv.

○ Create a .env.local file to store environment variables secure.

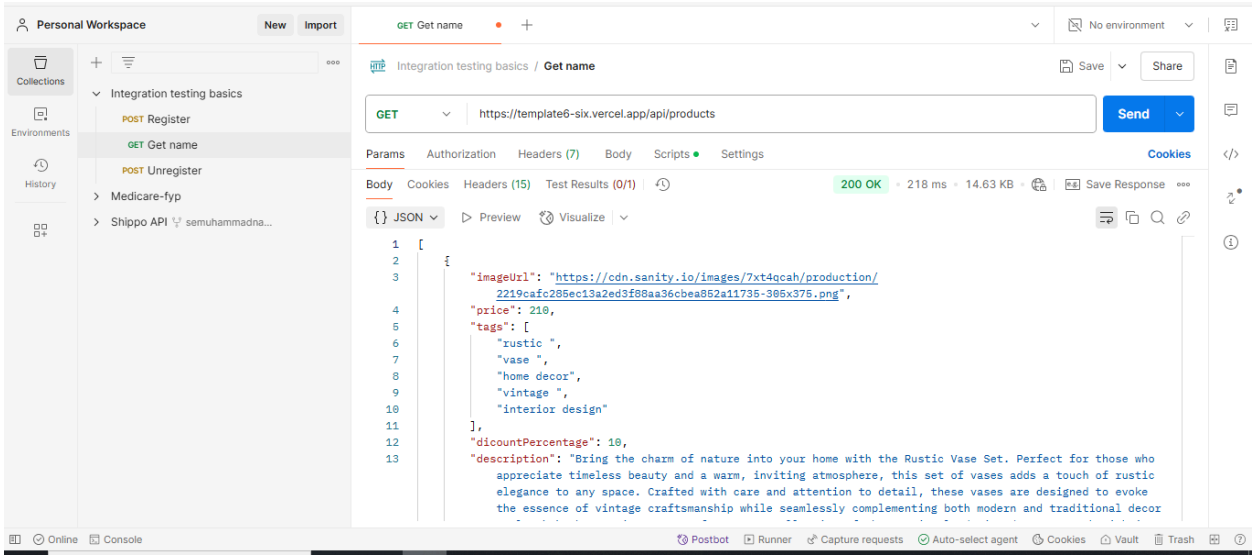
2) Data Fetching:

○ The `fetch()` function sends an HTTP GET request to the API endpoint to retrieve product data in JSON format.

3. Image Uploads:
○ Download images from API and upload them to Sanity Asset Manager using the Sanity client.

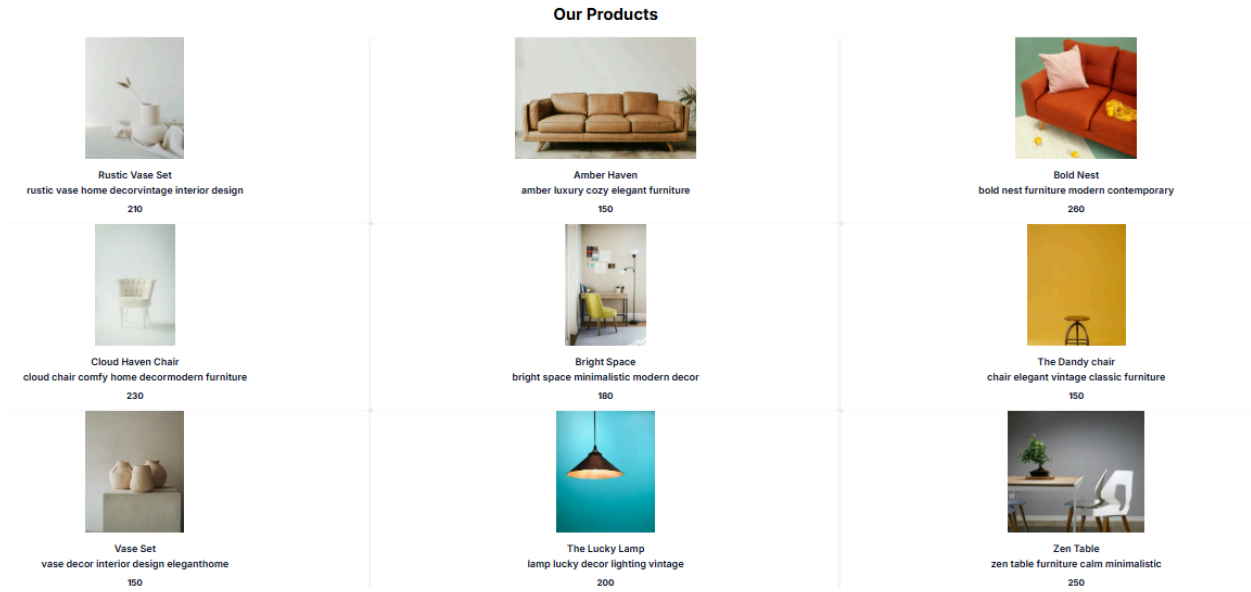
API Call with Postman.

To test the API call using Postman, follow these steps:
1. Open Postman and create a new request.
2. Set the request type to GET.
3. Enter the API Endpoint:
https://template6-six.vercel.app/api/products.
4. Click Send to retrieve the product data.
5. Verify the response payload structure and log details for confirmation.

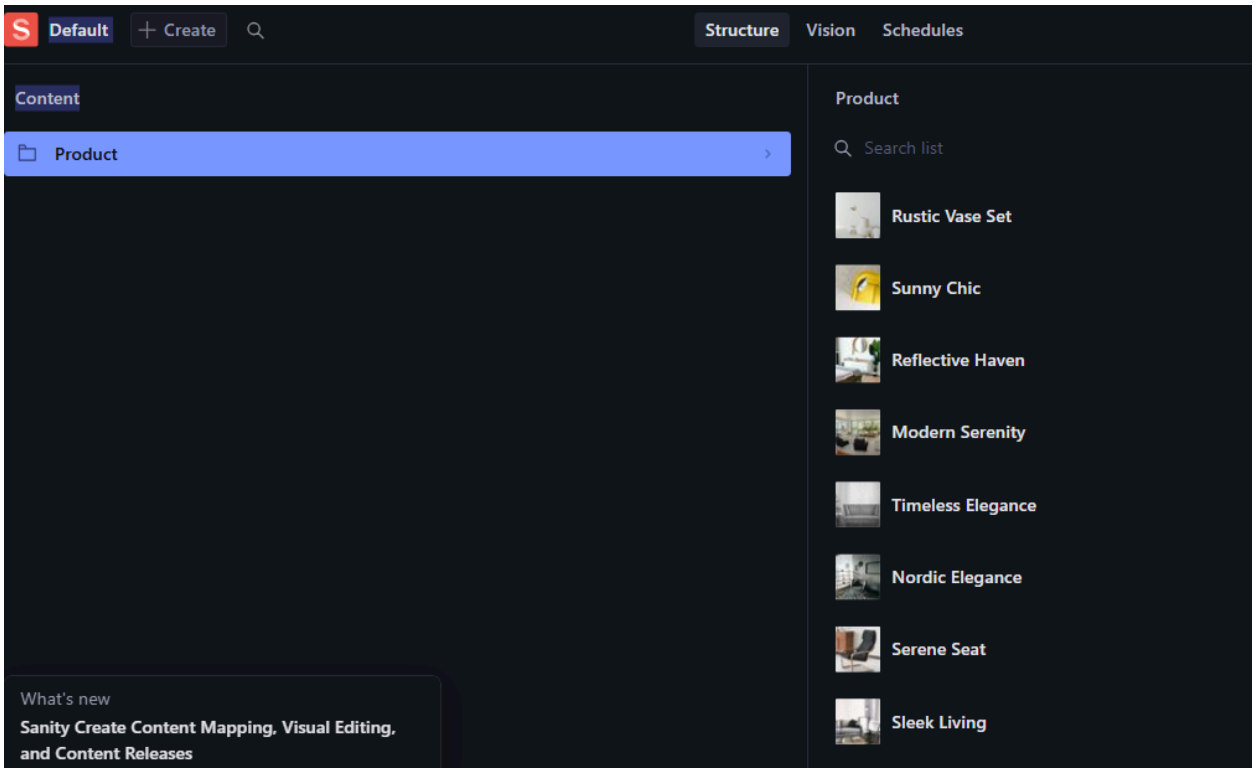


Screenshots:

FrontEnd Display.



Sanity CMS Fields.



API Integration.

```
    const [products, setProducts] = useState([]);
    const [loading, setLoading] = useState(true);

    useEffect(() => {
        const getProducts = async () => {
            setLoading(true); // Start loading when fetching starts
            try {
                const data = await fetchProducts(); // Call the utility
function
                setProducts(data); // Set products in the state
            } catch (error) {
                console.error("Error fetching products:", error);
            } finally {
                setLoading(false); // End loading
            }
        };

        getProducts();
    }, []);
```

Fetch.ts:

```
import { client } from '../sanity/lib/client'

export async function fetchProducts() {
    const query = `
      *[_type == "product"]{
        _id,
        _type,
        title,
        price,
       "productImage": productImage.asset->url,
        tags,
```

```
        discountPercentage,
        description,
        isNew
      }
    `
    const products = await client.fetch(query)
    return products
}
```

Code Snippet for Data Migration:

```
import { createClient } from '@sanity/client';

export const client = createClient({
    projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
    dataset: 'production',
    useCdn: true,
    apiVersion: '2025-01-17',
    token: process.env.SANITY_TOKEN,
});

async function uploadImageToSanity(imageUrl) {
    try {
        console.log(`Uploading image: ${imageUrl}`);

        const response = await fetch(imageUrl);
        if (!response.ok) {
            throw new Error(`Failed to fetch image: ${imageUrl}`);
        }

        const buffer = await response.arrayBuffer();
        const bufferImage = Buffer.from(buffer);

        const asset = await client.assets.upload('image', bufferImage, {
```

```
            filename: imageUrl.split('/').pop(),
        });

        console.log(`Image uploaded successfully: ${asset._id}`);
        return asset._id;
    } catch (error) {
        console.error('Failed to upload image:', imageUrl, error);
        return null;
    }
}

async function uploadProduct(product) {
    try {
        const imageId = await uploadImageToSanity(product.imageUrl);

        if (imageId) {
            const document = {
                _type: 'product',
                title: product.title,
                price: product.price,
                productImage: {
                    _type: 'image',
                    asset: {
                        _ref: imageId,
                    },
                },
                tags: product.tags,
                discount Percentage: product.dicountPercentage, // Typo in
field name: discount Percentage -> discountPercentage
                description: product.description,
                isNew: product.isNew,
            };

            const createdProduct = await client.create(document);
            console.log(`Product ${product.title} uploaded successfully:`,
createdProduct);
        } else {
            console.log(`Product ${product.title} skipped due to image
upload failure.`);
        }
```

```
    } catch (error) {
        console.error('Error uploading product:', error);
    }
}

async function importProducts() {
    try {
        const response = await
fetch('https://template6-six.vercel.app/api/products');

        if (!response.ok) {
            throw new Error(`HTTP error! Status: ${response.status}`);
        }

        const products = await response.json();

        for (const product of products) {
            await uploadProduct(product);
        }
    } catch (error) {
        console.error('Error fetching products:', error);
    }
}

importProducts();
```

Self-Validation Checklist:

	Task	Status (✔ or ✘)
0	API Endpoint Setup	✔
1	Schema Adjustments	✔
2	Data Migration	✔
3	Frontend Display	✔
4	Populated Sanity CMS Fields	✔

Prepared by: Muhammad Nehal Nadeem.