

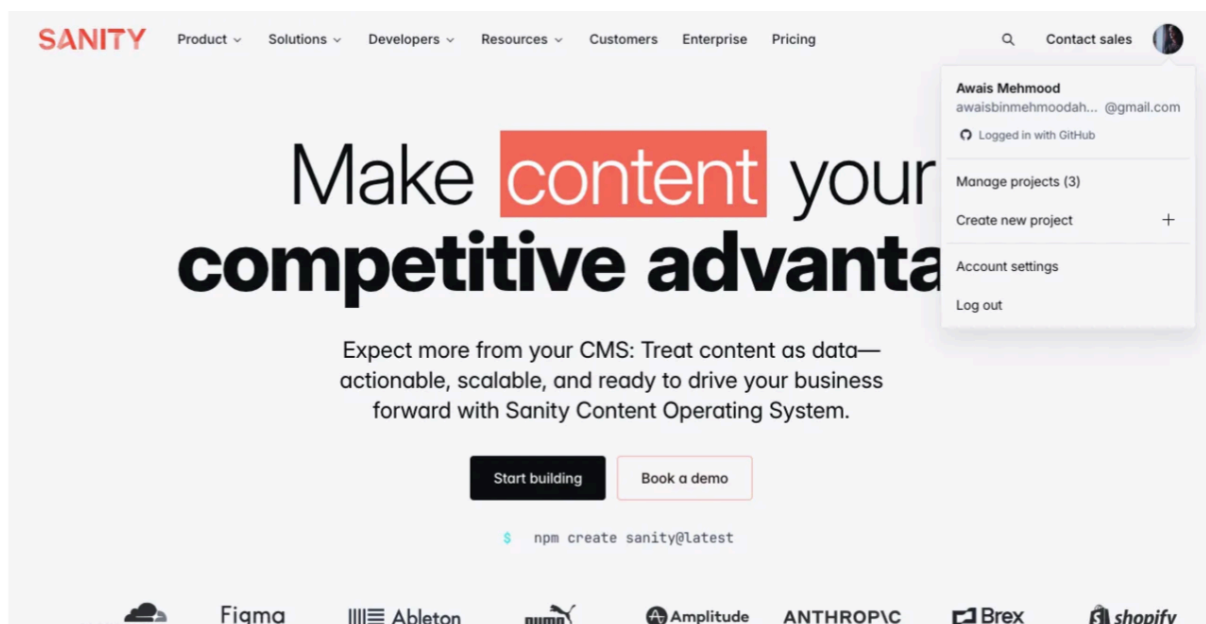
Day 3 - API Integration Report - G. Ecommerce [[Bandage](#)]


API Integration Process:

1. Setting Up Sanity Studio

First, open Sanity Studio and click on your **Profile** icon. Then, select **Manage Project**.

From the list of projects, choose one that you want to work with.





Awais Mehmood

Practice

24 days left in trial

PR

Awais Mehmood

Practice

PLAN

Growth Trial

STATUS

Active

PROJECT ID

25qpndph

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

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?

What's new

Sanity Create Content Mapping, Visual Editing, and Content Releases



Awais Mehmood

Practice

24 days left in trial

Getting started

Overview

Members

Studios

Datasets

Access

Activity

Usage

Plan

API

Settings

Webhooks

CORS origins

Tokens

Tokens

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

NAME

PERMISSIONS

CREATED

Name

Examples: "Employee import", "Website preview" or "PDF generator".

Permissions

Choose the access privileges for the token.

Contributor

☐ Read and write access to draft content within all datasets, with no access to project settings. (Tokens: read+write drafts)

Deploy Studio (Token only)

☐ Access to deploy Sanity Studio and GraphQL APIs to our hosted service.

Developer

☐ Read and write access to all datasets, with access to project settings for developers. (Tokens: read+write)

Editor

☒ Read and write access to all datasets, with limited access to project

What's new

Sanity Create Content Mapping, Visual Editing, and Content Releases

sanity/lib/client.ts

3. Configuring Environment Variables

Save the copied API token in your `.env.local` file like this:

```
SANITY_API="your_token_here"
```

4. Add in client folder

Add the api in your client folder or simply copy and paste this in

code in `sanity/lib/client.ts`

```
//sanity/lib/client.ts
```

```
import { createClient } from 'next-sanity'
```

```
import { apiVersion, dataset, projectId } from '../env'
```

```
export const client = createClient({
```

```
  projectId,
```

```
  dataset,
```

```
  apiVersion,
```

```
  token: process.env.SANITY_API,
```

```
  useCdn: true,
```

```
})
```

5. Defining the Product Schema

Create a file named `product.ts` in your schema folder and paste the following code or make your own: `sanity/schemaTypes//product.ts`

```
export default {
  name: 'product',
  title: 'Product',
  type: 'document',
  fields: [
    { name: 'name', type: 'string' },
    { name: 'subname', type: 'string' },
    { name: 'discount', type: 'number' },
    { name: 'price', type: 'number' },
    { name: 'description', type: 'text' },
    {
      name: 'image',
      type: 'image',
      options: {
        hotspot: true, // Optional:
        // This allows image cropping in
        // the Sanity Studio
      },
    },
  ],
};
```

Then, update your `sanity/schemaTypes/index.ts` file to include the new product schema:

```
import { type SchemaTypeDefinition } from 'sanity'

import product from './product'

export const schema: { types: SchemaTypeDefinition[] } = {
  types: [product],
}
```

Creating a Database | Mock Data

Next, create a `db.ts` file to create or fetch mock data:

```
// db.ts
export const fetchData = async () => {
  try {
    const response = await fetch(
      "https://677fff420476123f76a91d26.mockapi.io/eceommeceweb"
    );
    if (!response.ok) {
      throw new Error(`Failed to fetch data: ${response.statusText}`);
    }
    const data = await response.json();
    return data;
  } catch (error) {
    console.error("Error fetching data:", error);
    return [];
  }
};

// Fetch data and store it in a constant
export const demoData = await fetchData();
```

7. Building the Insert Page

```
async function uploadImageToSanity(imageUrl: string) {
  const res = await fetch(imageUrl);
  const blob = await res.blob();

  const imageAsset = await client.assets.upload('image', blob, {
    filename: 'image.jpg',
  });

  return imageAsset._id;
}

function Fetch() {
  async function insertData() {
    try {
      const result = await Promise.all(
        demoData.map(async (item: any) => {
          const imageAssetId = await uploadImageToSanity(item.image);

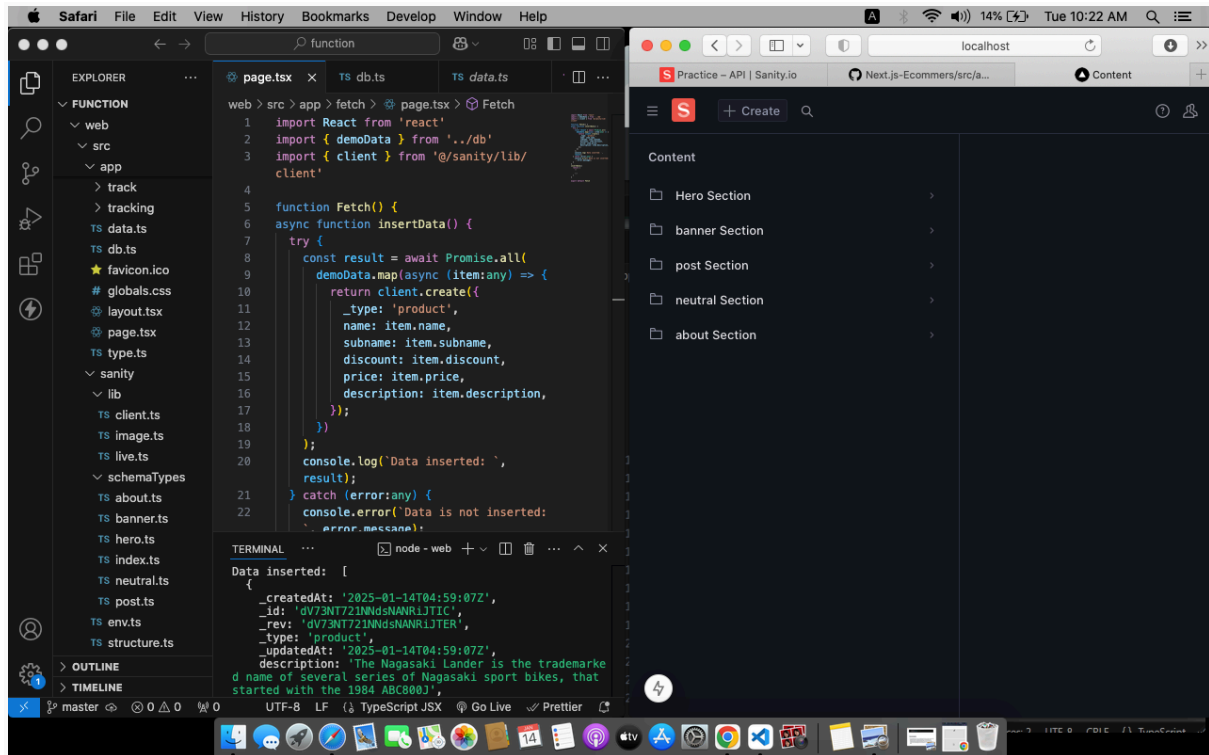
          return client.create({
            _type: 'product',
            name: item.name,
            subname: item.subname,
            discount: item.discount,
            price: item.price,
            description: item.description,
            image: {
              _type: 'image',
              asset: {
                _ref: imageAssetId,
                _type: 'reference',
              },
            },
          });
        })
      );
      console.log('Data inserted: ', result);
    } catch (error: any) {
      console.error('Data is not inserted: ', error.message);
    }
  }
  // insertData();

  return (
    <>
    | { /* Your JSX here */ }
    </>
  );
}

export default Fetch;
```

7. Testing and Verifying the Data

1. Run the development server using the terminal



Data Inserted Successfully!