DAY 3 - API INTEGRATION AND DATA MIGRATION

Introduction

API integration and data migration are critical aspects of web application development. In the Next.js framework, efficient handling of API calls and seamless data migration are essential to ensuring optimal performance and user experience. This document outlines the best practices, tools, and methodologies for integrating APIs and migrating data in a Next.js application using Sanity as a headless CMS.

API Integration in Next.js with Sanity

Understanding API Integration

API integration involves connecting a Next.js application with external services or databases through APIs (Application Programming Interfaces). Sanity provides a powerful content API that enables dynamic data fetching and management.

Methods of API Integration with Sanity

1. **Setting Up Sanity Client**

Install Sanity client:
```
npm install @sanity/client
```

-

Initialize Sanity client:
```
import sanityClient from '@sanity/client';

const client = sanityClient({
  projectId: '1y5hoazk
',
  dataset: 'production',
  useCdn: true, // `false` if you want fresh data
  apiVersion: '2023-02-08',
});
```

-

2. **Fetching Data from Sanity**

Using GROQ query to fetch data:

```
async function fetchData() {
 const query = `*[_type == "post"]{title, id, body}`;
 const data = await client.fetch(query);
 return data;
}
```

-

3. **Fetching Data in Next.js Pages**

Using `getServerSideProps` for server-side fetching:

```
export async function getServerSideProps() {
 const query = `*[_type == "post"]{title, id, body}`;
 const data = await client.fetch(query);
 return { props: { data } };
}
```

-

4. **Using Next.js API Routes for Sanity Integration**

Creating a custom API route in `pages/api/posts.js`:

```
export default async function handler(req, res) {
 const query = `*[_type == "post"]{title, id, body}`;
 const data = await client.fetch(query);
 res.status(200).json(data);
}
```

-

Data Migration in Next.js with Sanity

Understanding Data Migration

Data migration involves transferring content from one system, database, or format to another. When working with Sanity, data migration may include moving data from an old CMS or local database to Sanity's content management system.

1. **Transforming Data for Sanity**
    - **Formatting the data structure to match Sanity's schema**
    -
2. **Importing Data into Sanity**

```
import { defineType } from "sanity"


export const product = defineType({
    name: "product",
    title: "Product",
    type: "document",
    fields: [
        {
            name: "title",
            title: "Title",
            validation: (rule) => rule.required(),
            type: "string"
        },
        {
            name:"description",
            type:"text",
            validation: (rule) => rule.required(),
            title:"Description",
        },
        {
            name: "productImage",
            type: "image",
```

```
            validation: (rule) => rule.required(),
            title: "Product Image"
        },
        {
            name: "price",
            type: "number",
            validation: (rule) => rule.required(),
            title: "Price",
        },
        {
            name: "tags",
            type: "array",
            title: "Tags",
            of: [{ type: "string" }]
        },
        {
            name:"dicountPercentage",
            type:"number",
            title:"Discount Percentage",
        },
        {
            name:"isNew",
```

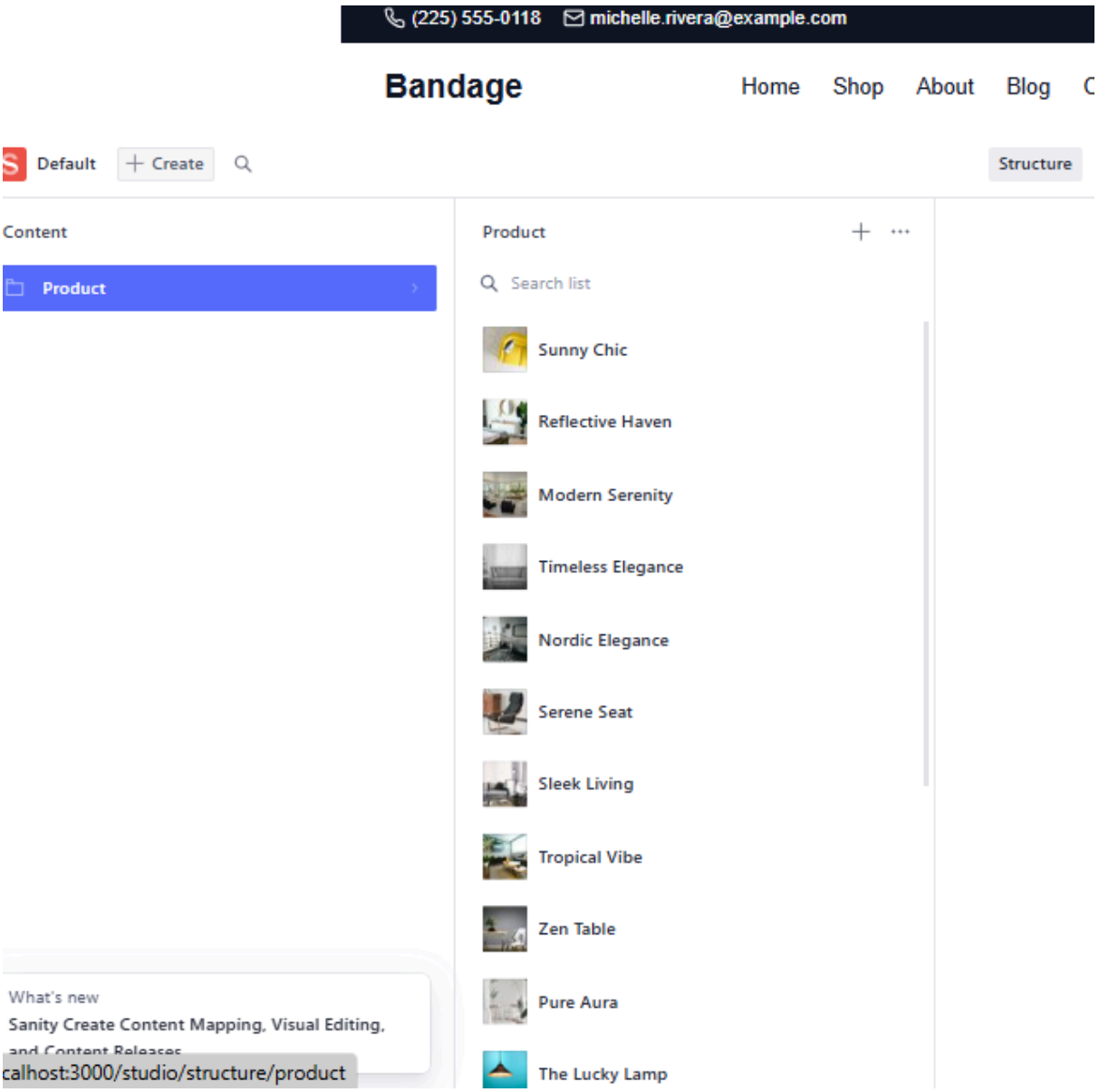
```
            type:"boolean",

            title:"New Badge",

        }

    ]

})
```

Use Sanity CLI to import data:
sanity dataset import data.json production

-

Alternatively, use Sanity's API:

```
async function importData() {
  const newData = {
    _type: 'post',
    title: 'New Post',
    slug: { _type: 'id', current: 'new-post' },
    body: 'This is a new post imported into Sanity.'
  };
  await client.create(newData);
  console.log('Data imported successfully!');
}
importData();
```



Use Environment Variables

Store Sanity project ID and API key securely in `.env.local`.

NEXT_PUBLIC_SANITY_PROJECT_ID=your_project_id

SANITY_API_TOKEN=your_api_token

-

Conclusion

API integration and data migration with Sanity in Next.js enable efficient content management and data handling. By leveraging Sanity's API, GROQ queries, and automated data migration scripts, developers can build dynamic, scalable applications while ensuring data integrity and performance.