Day 3 - API Integration and Data Migration Report - [Car Rental Marketplace]

Introduction

In this report, I document the process I followed for integrating APIs into my Next.js project and migrating data into Sanity CMS. This process involved several steps, including setting up the API integration, adjusting the schema in Sanity CMS, and migrating the data from the external source to Sanity. The end result is a functional marketplace backend, with data displayed dynamically in the frontend.

Step 1: Logging into Sanity

1. Install Sanity CLI

To start working with Sanity CMS, you first need to install the Sanity CLI (Command Line Interface) on your system. If you don't have it installed, follow the steps below:

- Open your terminal and run the following command to install the Sanity CLI globally:

```bash
npm install -g @sanity/cli
```

2. Log into Sanity

- Visit Sanity.io and log into your account.
- Once logged in, navigate to your Project Dashboard.
- Create the project by writing name of project and choose account or Select the existing project you are working on.

3. Creating an API Token

To securely interact with the Sanity CMS API, you need an **API token**.

- From the left-hand sidebar, click on **Settings**.
- Under **Settings**, select **API**.

- In the **API** section, click on the **Tokens** tab.
- Click **Add New Token** to create a new API token.
- Name the token (e.g., "Next.js Marketplace Token").
- Choose Developer mode.
- After creating the token, copy it and store it securely.

4. Storing the API Token Securely

Store your API token in a .env.local file to keep it secure. Do not expose it in your public code.

1. Create a .env.local file in the root of your Next.js project.
2. Add the following lines to store your token and project details:

```bash
SANITY_PROJECT_ID=your-sanity-project-id
SANITY_DATASET=production
SANITY_TOKEN=your-sanity-auth-token (if needed for private dataset access)
```

You can get the SANITY_PROJECT_ID and SANITY_TOKEN from your Sanity project's settings (under **API**).

3. Add env.local to your **.gitignore** file to prevent it from being committed to version control.

Step 2: API Integration

API Integration Overview

To integrate external data into the project, I used the API Link [https:// sanity-nextjs-application.vercel.app/api/hackathon/template7]provided for this template. The API provided endpoints to fetch data related to products, categories, and other essential information for the marketplace.

Steps Taken for API Integration

1. Installed Sanity Client:
    - I installed the Sanity client to interact with the Sanity CMS.

```bash
npm install @sanity/client
```

2. Set up the Sanity Client:

    - I created a configuration file (sanity/lib) for the Sanity client to connect to the Sanity project.

```javascript
import sanityClient from '@sanity/client';

export const client = sanityClient({
  projectId: process.env.SANITY_PROJECT_ID, // Your Sanity project ID
  dataset: process.env.SANITY_DATASET,     // Your dataset name
  token: process.env.SANITY_TOKEN,         // API token (if necessary)
  useCdn: true,                            // Use CDN for fast reads
});
```

Step 3: Schema Adjustments

Schema Overview

The Sanity CMS schema defines the structure of content in the marketplace. During this project, I adjusted the schema to align with the data provided by the API, ensuring that the imported data could be stored and rendered properly.

Schema Adjustments Process

1. **Original Schema**:
    - The initial schema for the "Car" document type is written in **SanitySchema.pdf** file in Images folder.

3. **Mapping API Data to Schema**:

    When migrating data from the API, I mapped the API fields to the corresponding Sanity fields. For instance:

- API field **car_title** → Sanity field **name**
- API field **car_image** → Sanity field **image**

Step 4: Data Migration

Migration Steps and Tools Used

For data migration, I followed these steps:

1. **Fetching Data from the API**:
   - I used the provided API endpoint (/api/cars) to fetch car data.
2. **Migration Script**:
   - I wrote a migration script to fetch data from the API and insert it into Sanity CMS. The script I used is written in Migration.pdf file in Images folder.

Step 5: Using GROQ Queries

1. **What is GROQ?**
   GROQ (Graph-Relational Object Query) is a query language used to query Sanity's content. It is similar to SQL or GraphQL and allows you to filter, project, sort, and join data easily.
2. Use GROQ queries to fetch data from Sanity and display it in your Next.js pages. You can find groq query which I had written in groq query.pdf file in Image folder.

Step 6: Using the Sanity Client in Next.js Pages

Now that you have configured the Sanity client, you can start using it to fetch data in your Next.js pages. Example you can find in Page.pdf file in Images folder.

Step: 7: Testing API Integration

After setting up the API token and integrating it into your Next.js project:

1. Run your Next.js app locally with the following command:
2. Visit **http://localhost:3000** in your browser to verify that data from Sanity is correctly displayed on the frontend.

```bash
npm run dev
```

Best Practices

- Store API tokens securely in .env.local.
- Use GROQ efficiently to filter, sort, and fetch only the necessary data.
- Validate data during migration and adjust schemas to ensure compatibility.
- Test thoroughly: Test API integrations and ensure data is displayed correctly.
- Back up your project before performing large data migrations.

Conclusion

By following the above steps, I successfully integrated the API, adjusted the Sanity schema, and migrated the data to Sanity CMS. The data is now displayed dynamically on the frontend of the marketplace, and the Sanity CMS is fully populated with product information.