DAY 3 - API INTEGRATION AND DATA MIGRATION

Hackathon Day 3: API Integration and Data Migration

Objective

Today’s focus was on integrating an external API and migrating its data into Sanity CMS. The goal was to create a robust and dynamic backend for a marketplace, ensuring seamless API integration, data transformation, and rendering on the frontend.

Steps I Performed

1. **Understanding the API**
   - **API URL:** https://hackathon-apis.vercel.app/api/products
   - The API provided details about products, categories, and images.
   - I tested all endpoints using Postman to ensure the responses were correct and structured for use.
2. **Setting Up the Backend**
   - **Tools I used:**
     - `axios`: For fetching data from the API.
     - `next-sanity`: To interact with Sanity CMS.
     - `slugify`: To create clean and unique slugs for each product.
3. **Sanity Schema Adjustments**
   - I updated the Product schema by adding fields like tags, dimensions, and categories.
   - Created a new Category schema for storing product categories.
   - Established a reference between Product and Category schemas to link them effectively.
4. **Data Migration**
   - **Here’s the process I followed:**
     - **Fetching Data:** Used axios to pull data from the API.
     - **Image Uploads:** Uploaded the product images to Sanity CMS using its Asset Manager.
     - **Data Transformation:** Formatted the API data to align with my Sanity schema.
     - **Importing to Sanity:** Used the createOrReplace method to insert data efficiently.
   - **Error Handling:**
     - I wrapped API calls and migrations in try-catch blocks.
     - Logged errors for easier debugging in case of failures.
5. **Frontend Integration**
   - I created reusable backend functions to fetch data from Sanity:
     - `getAllProducts`: Fetches all available products.
     - `getPopularProducts`: Fetches featured products.
     - `getProductBySlug`: Retrieves specific product details using a slug.
   - The imported data was then rendered in the frontend, ensuring it displayed accurately in the marketplace UI.

Challenges and Solutions

1. **Large Image Files**
   - **Challenge:** Uploading large files caused delays.
   - **Solution:** Used the Sanity Asset Manager with buffer support to handle images.

2. **Schema Mismatches**
   - **Challenge:** Field discrepancies between API data and Sanity schema.
   - **Solution:** Adjusted schemas and applied a transformation script to map the fields properly.
3. **Data Integrity**
   - **Challenge:** Migrating data accurately without any losses.
   - **Solution:** Added validation checks to ensure all required fields were present and correctly formatted.

Performance Optimizations

- Batched API requests to avoid hitting rate limits and improved migration speed.
- Cached frequently accessed API responses to minimize redundant network calls.
- Limited query fields to fetch only the necessary data, reducing response size.

Results

- Successfully imported all products and categories into Sanity CMS.
- Verified data accuracy in the Sanity CMS Studio.
- Displayed the imported data seamlessly in the frontend marketplace.

Future Improvements

- Automate schema validation to detect mismatches before migration.
- Add detailed logging to track potential issues effectively.
- Implement real-time syncing with external APIs using Sanity's webhooks.

Environment Setup

Setting up environment variables in `.env.local`:

```
NEXT_PUBLIC_SANITY_PROJECT_ID=your_project_id
NEXT_PUBLIC_SANITY_DATASET=production
SANITY_API_TOKEN=your_sanity_token
```

Generating Sanity API Token:

1. Log in at Sanity Dashboard.
2. Select the project and navigate to the "API" tab.
3. Create a new API token with Editor permissions and copy it to the `.env.local` file.

Script for Data Migration

Here's the basic migration script:

1. **Install Necessary Packages:**

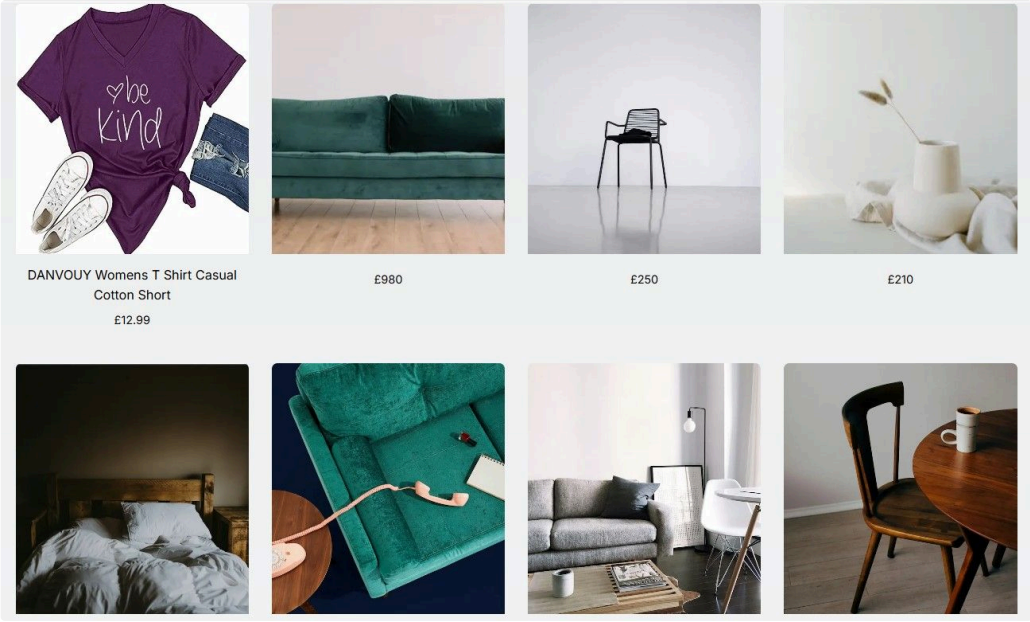
```
npm install @sanity/client axios dotenv
```

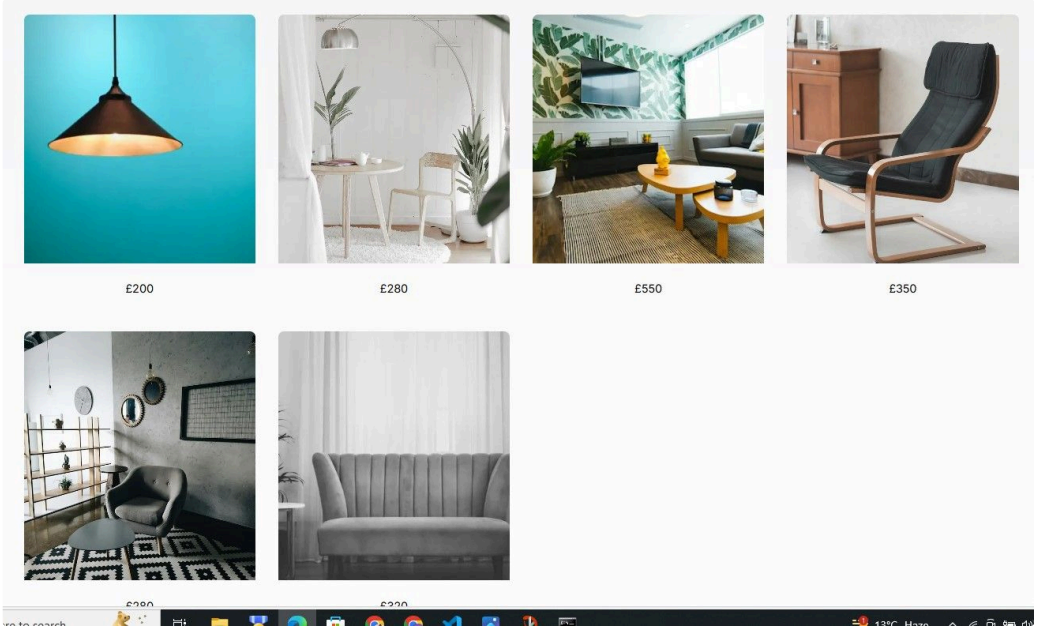
2. **Create** `scripts/importSanityData.mjs` file.
3. **Update** `package.json` Script Section:

```
"import-data": "node scripts/importSanityData.mjs"
```

4. **Run the script using:**

```
npm run import-data
```





The imported data was retrieved from an external API and ingested into Sanity CMS to form the backend structure for a dynamic marketplace. This process involved mapping, transforming, and validating the data to ensure compatibility with the predefined schema.

Final Words

Day 3 was both challenging and rewarding! I learned how to integrate an external API, modify schemas dynamically, and seamlessly manage data migration. Looking forward to using these techniques in future projects!

#Hackathon #Day3 #SanityCMS #APIIntegration #DataMigration