

Day 3 - API Integration Report - AliCarGo

Author:
Ali Shahid

Roll no:
00230027

Day 3 of marketplace builder hackathon

2. API Integration Process

Overview

The goal of this task was to integrate external APIs into our marketplace application to facilitate data retrieval, manipulation, and display. The integration enhances our system's functionality by automating tasks and streamlining operations.

Steps Involved:

1. Identifying API Requirements

- The first step involved identifying the key functionalities required for the marketplace (e.g., fetching product data, processing transactions).
- Chose APIs based on data needs and ensured they met the marketplace's requirements.

2. Integrating APIs

- Installed necessary libraries (e.g., axios, fetch) to handle API requests in the application.
- Set up environment variables to securely store API keys and credentials.
- Developed functions to fetch data from the external API endpoints and map it to the marketplace data structure.

3. Handling Data in the Application

- The data fetched via the API was integrated into the existing components of the marketplace.
- Used state management (e.g., React's useState) to store API data and display it dynamically on the frontend.

4. Error Handling & Debugging

- Implemented error handling to manage failed API requests or missing data.
- Logs were set up to track API response status and pinpoint potential issues.

3. Adjustments Made to Schemas

Product Schema Updates

- Added New Fields for API Data

- Introduced new fields such as `image_url`, `tags`, and `price_per_day` to the product schema to accommodate the data fetched from the external API.
- Example:
 - javascript
 - CopyEdit
 - {
 - name: 'image_url',
 - type: 'string',
 - title: 'Product Image URL'
 - }
 - **Revised Existing Fields**
 - Updated the price field to handle different pricing models (e.g., hourly, daily).
- Example:
 - javascript
 - CopyEdit
 - {
 - name: 'price_per_day',
 - type: 'string',
 - title: 'Price per Day'
 - }

Custom Data Types

- Introduced custom fields to manage external product categories (tags), which were mapped to the API data for better classification and searchability.

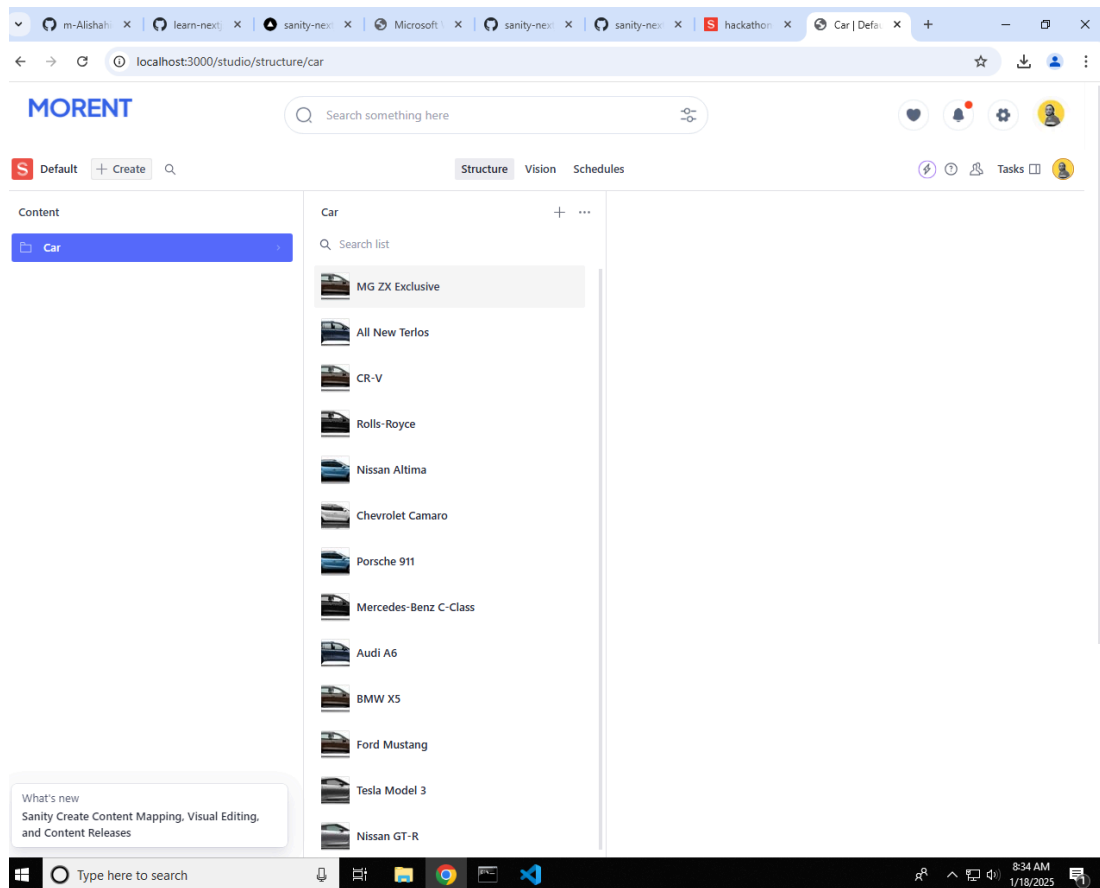
4. Migration Steps and Tools Used

Migration Process:

1. **Database Backup**
 - Prior to initiating the migration, a full backup of the existing database was performed to avoid data loss.
2. **Data Mapping and Transformation**
 - Data from the external API was mapped to the existing database schema using a script that matched API fields with the marketplace data fields.
3. **Data Insertion**
 - The transformed data was inserted into the marketplace's database, ensuring that the new fields were populated accurately.
 - For large data sets, batch processing was used to minimize performance issues.
4. **Validation**
 - After the migration, all product data was validated to ensure it was correctly formatted and visible on the frontend.

Code snippets for API integration and migration scripts

```
File Edit Selection View Go Run ... hackathon2
EXPLORER TS query.ts U page.tsx M TS fetchdata.ts U TS client.ts U JS importTemplate7Data.mjs 1, U X JS next.config.mjs
HACKATHON2
  > .next
  > node_modules
  > public
  > scripts
    JS importTemplate7Data.mjs 1, U
      src
      app
        > admin
        > categories
        > components
        > details
        > fonts
        > payment
        > studio
      # globals.css
      layout.tsx
      page.tsx
      components
      lib
      sanity
        lib
          TS client.ts
          TS fetchdata.ts
          TS image.ts
          TS live.ts
          TS query.ts
          > schemaTypes
          TS env.ts
          TS structure.ts
      .env.local
      .gitignore
      components.json
scripts > JS importTemplate7Data.mjs > client
1 import { createClient } from '@sanity/client';
2 import dotenv from 'dotenv';
3 import { fileURLToPath } from 'url';
4 import path from 'path';
5
6 const __filename = fileURLToPath(import.meta.url);
7 const __dirname = path.dirname(__filename);
8 dotenv.config({ path: path.resolve(__dirname, '../.env.local') });
9
10 const client = createClient({
11   projectId: process.env.SANITY_PROJECT_ID,
12   dataset: process.env.SANITY_DATASET,
13   useCdn: true,
14   token: process.env.SANITY_API_TOKEN,
15   apiVersion: '2023-10-10',
16 });
17
18 async function uploadImageToSanity(imageUrl) {
19   try {
20     console.log('Uploading image: ${imageUrl}');
21     const response = await axios.get(imageUrl, { responseType: 'arraybuffer' });
22     const buffer = Buffer.from(response.data);
23     const asset = await client.assets.upload('image', buffer, {
24       filename: imageUrl.split('/').pop()
25     });
26     console.log('Image uploaded successfully: ${asset._id}');
27     return asset._id;
28   } catch (error) {
29     console.error('Failed to upload image:', imageUrl, error);
30     return null;
31   }
32 }
33
34 async function importData() {
35   try {
36     console.log('Fetching car data from API...');
37     // Fetch car data from API
38   } catch (error) {
39     console.error('Failed to fetch car data:', error);
40   }
41 }
```



m-A x | lear x | sani x | Mic x | sani x | sani x | S hact x | Boc x | Che x | Boc x | N Ver x | + | - | X

localhost:3000 | ☆ | 👤 | ⋮


Popular Car

View All

Koenigsegg

♥

Sport



70L

Manual

4 People


\$99.00/day

Rent Now

NissanGT - R

♡

Sport



70L

Manual

4 People


\$99.00/day

Rent Now

Rolls-Royce

♥

Sedan



70L

Manual

4 People


\$99.00/day

Rent Now

NissanGT - R

♡

Sport



70L

Manual

4 People

\$99.00/day


Rent Now

Recommendation Car

All New Rush

♥

SUV



70L

Manual

4 People


\$99.00/day

Rent Now

CR - V

♡

SUV



70L

Manual

4 People


\$99.00/day

Rent Now

All New Terios

♥

SUV



70L

Manual

4 People


\$99.00/day

Rent Now

CR - V

♡

SUV



70L

Manual

4 People

\$99.00/day

Rent Now