Day 3 - API Integration Report for Car Rental-E-Commerce Website

Overview

On Day 3, the focus was to integrate APIs into the Car-rental-e-commerce project and populate Sanity CMS with product data sourced from a local API. This report documents the API integration process, schema adjustments, data migration steps, and the tools utilized. Screenshots and code snippets are included to provide a comprehensive understanding.

API Integration Process

1. **Data Source**
   Data was fetched from the API endpoint:
   https://sanity-nextjs-application.vercel.app/api/hackathon/template7
   - **Fields included:** Product Name, description, price, images, inventory, categories, and status.
2. **Integration Steps**
   - **Schema Design**: Created a custom schema for Rental Car products in Sanity CMS to match the data structure.
   - **Import Script**: Developed a script to fetch product data, process it, and upload it to Sanity CMS.
   - **Image Handling**: Implemented logic to handle multiple product images and upload them as unique assets in Sanity CMS.
   - **Data Validation**: Ensured all fields like slug, price, and stock level adhered to validation rules.
   - **API Endpoints**: Created endpoints to fetch data from Sanity CMS for frontend usage.

Schema Adjustments

Schema adjustments refer to modifications made to the structure of a database or data model to enhance functionality, improve efficiency, or accommodate new requirements. For the Shop.co clothing brand website, schema adjustments may involve refining product categories, optimizing inventory management, integrating customer preferences, or improving order processing. These changes ensure seamless data flow, enhance user experience, and support business growth by adapting to evolving market needs.

Schema Example

```
export default {
    name: 'car',
    type: 'document',
    title: 'Car',
    fields: [
      {
        name: 'name',
        type: 'string',
        title: 'Car Name',
      },
      {
        name: 'brand',
        type: 'string',
        title: 'Brand',
        description: 'Brand of the car (e.g., Nissan, Tesla, etc.)',
      },
      {
        name: 'type',
        type: 'string',
        title: 'Car Type',
        description: 'Type of the car (e.g., Sport, Sedan, SUV, etc.)',
      },
      {
        name: 'fuelCapacity',
        type: 'string',
        title: 'Fuel Capacity',
        description: 'Fuel capacity or battery capacity (e.g., 90L, 100kWh)',
      },
      {
        name: 'transmission',
        type: 'string',
        title: 'Transmission',
        description: 'Type of transmission (e.g., Manual, Automatic)',
      },
      {
        name: 'seatingCapacity',
        type: 'string',
        title: 'Seating Capacity',
        description: 'Number of seats (e.g., 2 People, 4 seats)',
      },
      {
        name: 'pricePerDay',
        type: 'string',
        title: 'Price Per Day',
        description: 'Rental price per day',
      },
      {
        name: 'originalPrice',
        type: 'string',
        title: 'Original Price',
        description: 'Original price before discount (if applicable)',
      },
      {
        name: 'tags',
        type: 'array',
        title: 'Tags',
        of: [{ type: 'string' }],
        options: {
          layout: 'tags',
        },
        description: 'Tags for categorization (e.g., popular, recommended)',
      },
      {
        name: 'image',
        type: 'image',
        title: 'Car Image',
        options: {
          hotspot: true
        }
      }
    ],
  };
```

Migration Steps

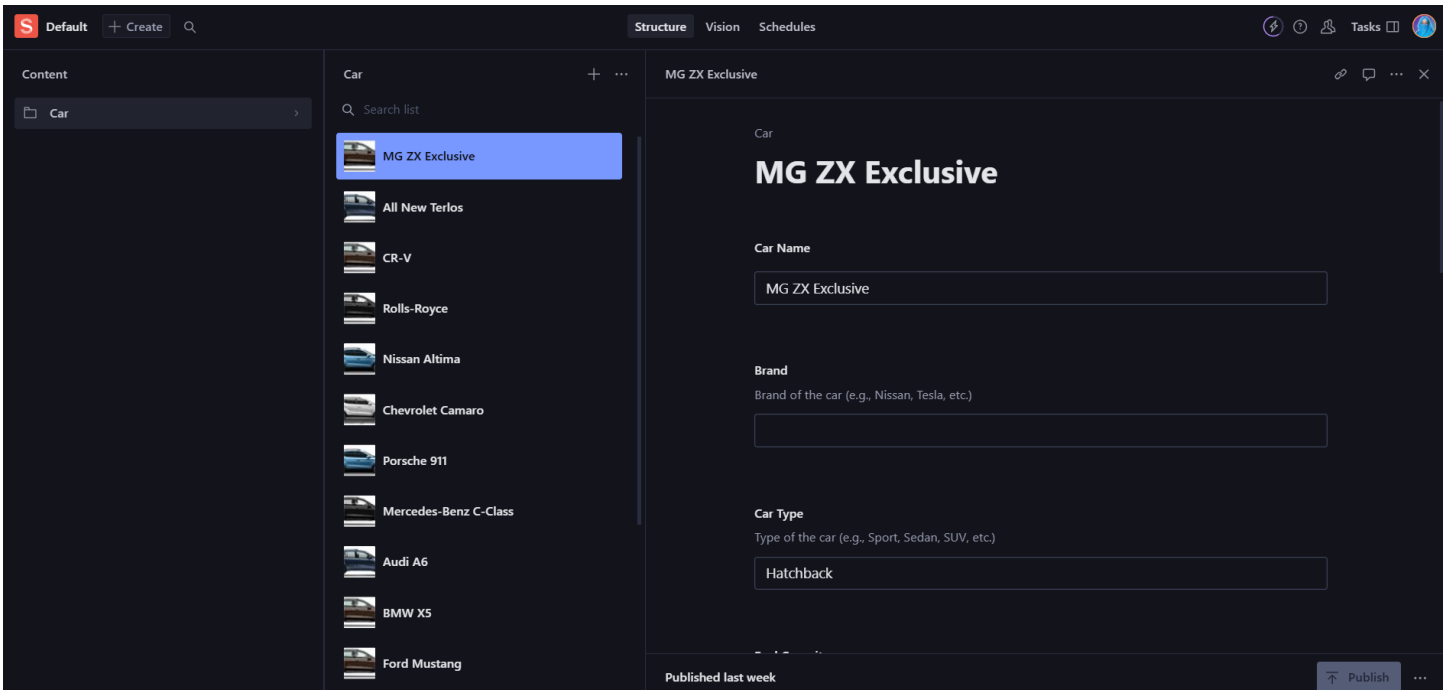
1. **Tools Used**
    - **Sanity Client**: For uploading data to Sanity CMS.
    - **Axios**: For making API calls.
    - **.env**: To secure API keys and environment variables.

Migration Script Example

```
import { createClient } from '@sanity/client';
import axios from 'axios';
import dotenv from 'dotenv';
import { fileURLToPath } from 'url';
import path from 'path';

// Load environment variables from .env.local
const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);
dotenv.config({ path: path.resolve(__dirname, '../.env.local') });

// Create Sanity client
const client = createClient({
  projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
  dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
  useCdn: false,
  token: process.env.SANITY_API_TOKEN,
  apiVersion: '2021-08-31'
});

async function uploadImageToSanity(imageUrl) {
  try {
    console.log(`Uploading image: ${imageUrl}`);
    const response = await axios.get(imageUrl, { responseType: 'arraybuffer' });
    const buffer = Buffer.from(response.data);
    const asset = await client.assets.upload('image', buffer, {
      filename: imageUrl.split('/').pop()
    });
    console.log(`Image uploaded successfully: ${asset._id}`);
    return asset._id;
  } catch (error) {
    console.error('Failed to upload image:', imageUrl, error);
    return null;
  }
}

async function importData() {
  try {
    console.log('Fetching car data from API...');

    // API endpoint containing car data
    const response = await axios.get('https://sanity-nextjs-application.vercel.app/api/hackathon/template7');
    const cars = response.data;

    console.log(`Fetched ${cars.length} cars`);

    for (const car of cars) {
      console.log(`Processing car: ${car.name}`);

      let imageRef = null;
      if (car.image_url) {
        imageRef = await uploadImageToSanity(car.image_url);
      }

      const sanityCar = {
        _type: 'car',
        name: car.name,
        brand: car.brand || null,
        type: car.type,
        fuelCapacity: car.fuel_capacity,
        transmission: car.transmission,
        seatingCapacity: car.seating_capacity,
        pricePerDay: car.price_per_day,
        originalPrice: car.original_price || null,
        tags: car.tags || [],
        image: imageRef ? {
          _type: 'image',
          asset: {
            _type: 'reference',
            _ref: imageRef,
          },
        } : undefined,
      };

      console.log('Uploading car to Sanity:', sanityCar.name);
      const result = await client.create(sanityCar);
      console.log(`Car uploaded successfully: ${result._id}`);
    }

    console.log('Data import completed successfully!');
  } catch (error) {
    console.error('Error importing data:', error);
  }
}

importData();
```

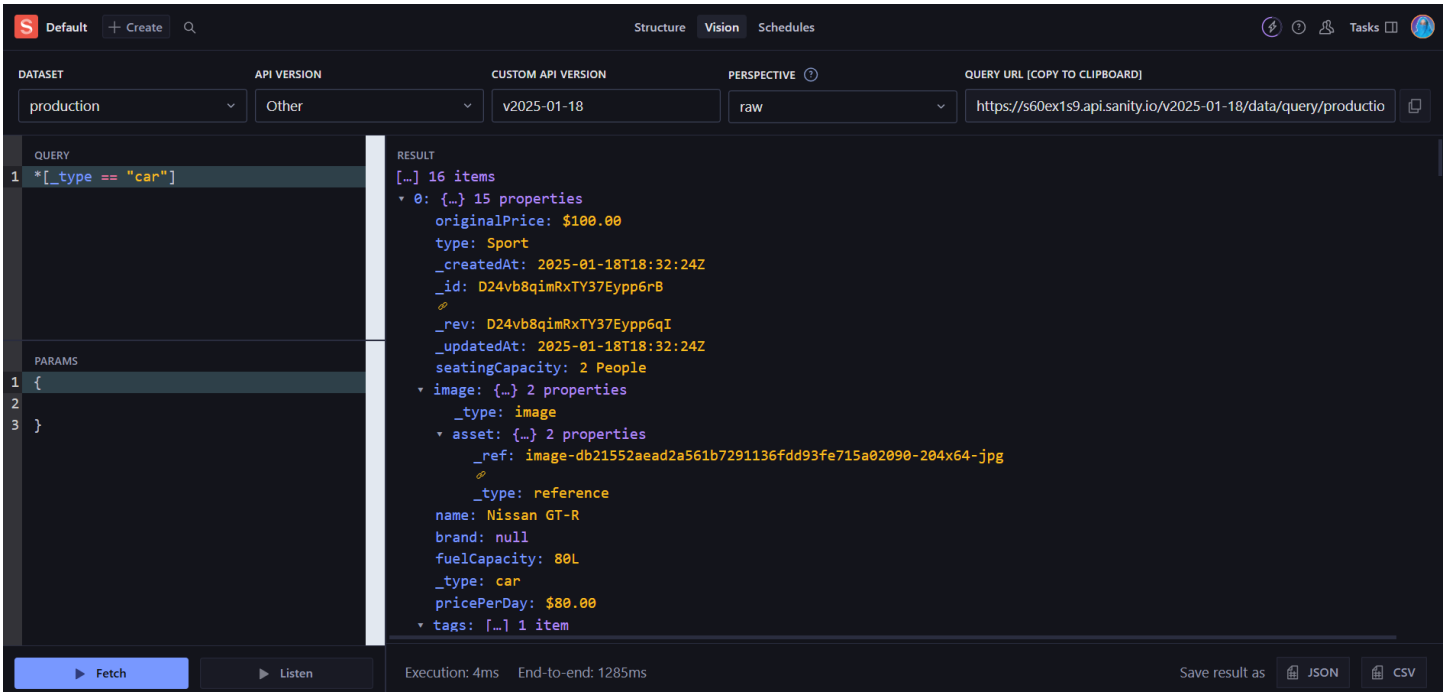
API Calls

Fetch All Products



API Response Validation

- Screenshot of API responses tested in Postman or browser.



Conclusion

The integration established a seamless process for importing, validating, and displaying Reatal Car data from the local API into Sanity CMS. Custom schemas and migration scripts ensured data consistency and project alignment. Future steps include enhancing API queries and implementing advanced features like search and filtering.