Day-03 API Integration and Data Migration Report

Introduction

Project Overview

This project involves integrating a car rental API into a frontend application, utilizing Sanity CMS for managing car data. The primary goal is to dynamically upload car details, including images, to Sanity via an API and use this structured data to create an interactive and user-friendly frontend experience. This seamless integration enhances data management while ensuring a reliable, scalable backend for the car rental marketplace.

API Integration:

API Used

The data is fetched from the following API endpoint:

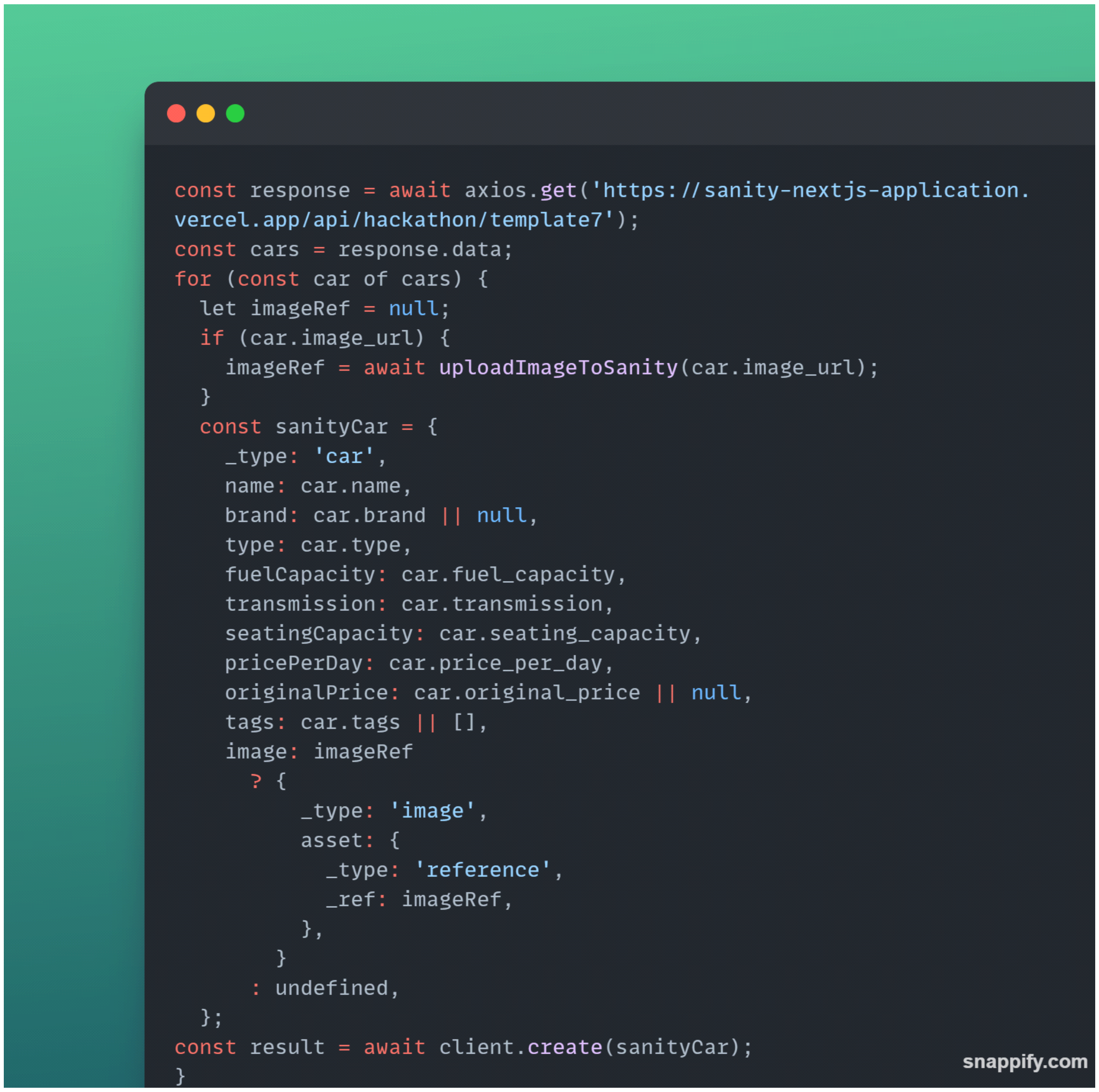
URL: https://sanity-nextjs-application.vercel.app/api/hackathon/template7

Provided Data:
Car Name
Brand
Type
Fuel Capacity
Transmission
Seating Capacity
Price Per Day
Original Price
Tags
Image URL

Fetching and Uploading Data:

Data is fetched using Axios and processed iteratively to upload car details to Sanity CMS. This involves uploading the car images and referencing them appropriately within the document structure.

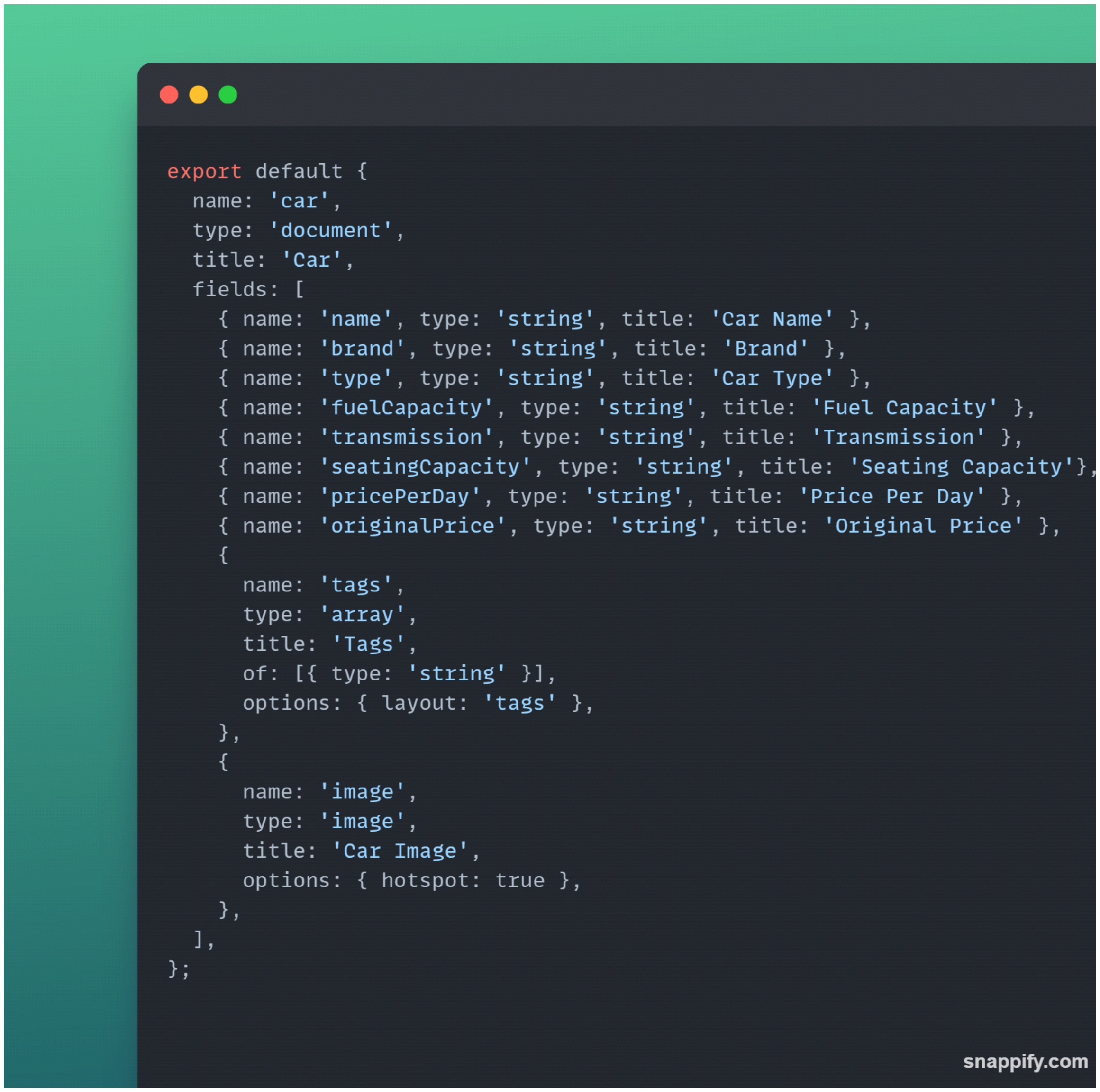
Code Snippet (Data Migration Script):



```
const response = await axios.get('https://sanity-nextjs-application.
vercel.app/api/hackathon/template7');
const cars = response.data;
for (const car of cars) {
  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,
  };
const result = await client.create(sanityCar);
}
```

Sanity Schema:

The schema for the "Car" document in Sanity CMS ensures flexible and dynamic content management. Below is the schema used:



```
export default {
  name: 'car',
  type: 'document',
  title: 'Car',
  fields: [
    { name: 'name', type: 'string', title: 'Car Name' },
    { name: 'brand', type: 'string', title: 'Brand' },
    { name: 'type', type: 'string', title: 'Car Type' },
    { name: 'fuelCapacity', type: 'string', title: 'Fuel Capacity' },
    { name: 'transmission', type: 'string', title: 'Transmission' },
    { name: 'seatingCapacity', type: 'string', title: 'Seating Capacity'},
    { name: 'pricePerDay', type: 'string', title: 'Price Per Day' },
    { name: 'originalPrice', type: 'string', title: 'Original Price' },
    {
      name: 'tags',
      type: 'array',
      title: 'Tags',
      of: [{ type: 'string' }],
      options: { layout: 'tags' },
    },
    {
      name: 'image',
      type: 'image',
      title: 'Car Image',
      options: { hotspot: true },
    },
  ],
};
```

This schema allows for managing car details, including metadata and associated tags, with flexibility to update dynamically.

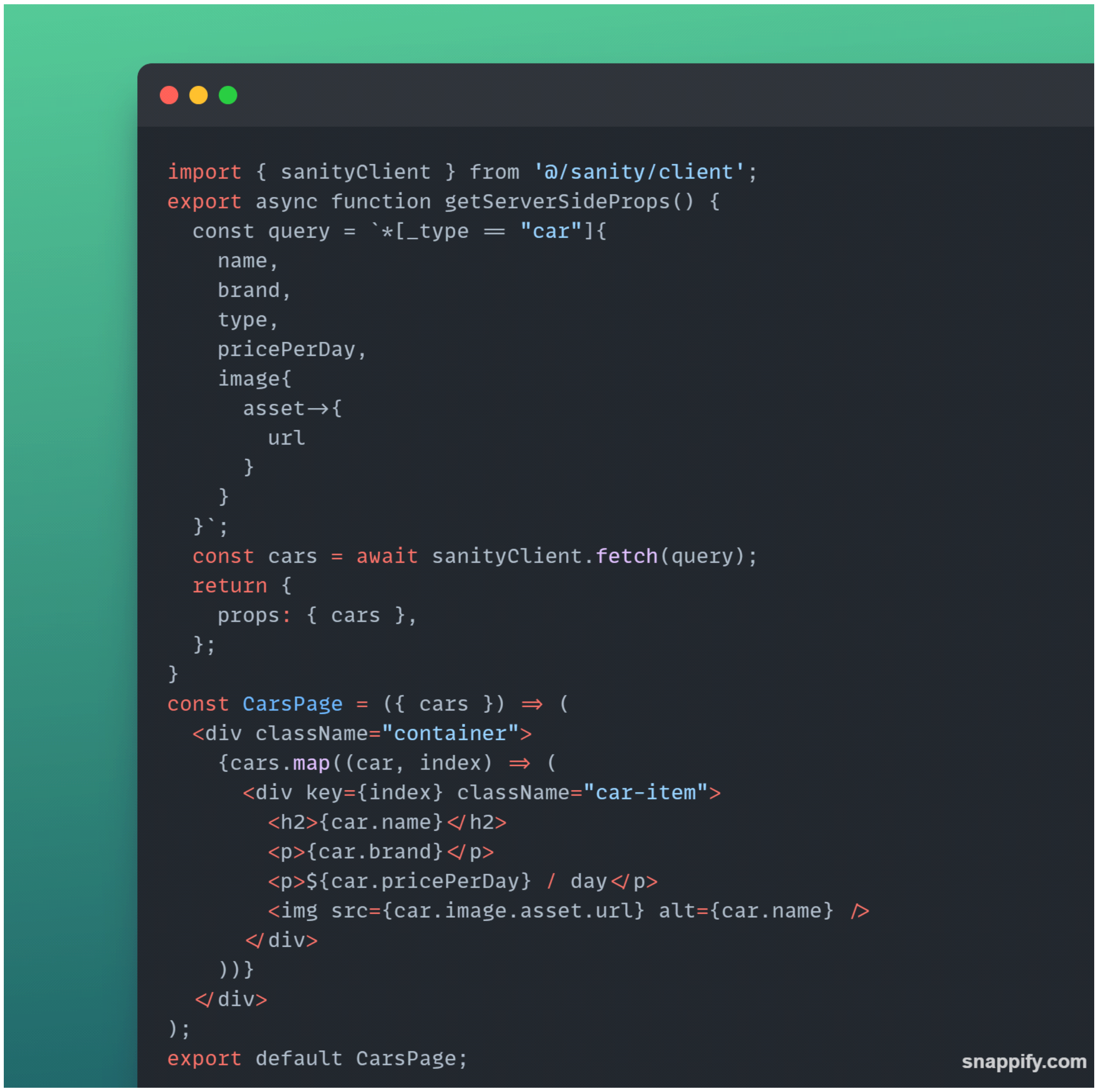
Frontend Integration:

Rendering Data

The car data from Sanity CMS is dynamically rendered on the frontend using Next.js.
Here is an example of how the This schema allows for managing car details, including metadata and associated tags, with flexibility to update dynamically.

data is fetched and displayed:

Code Snippet (Frontend Integration):



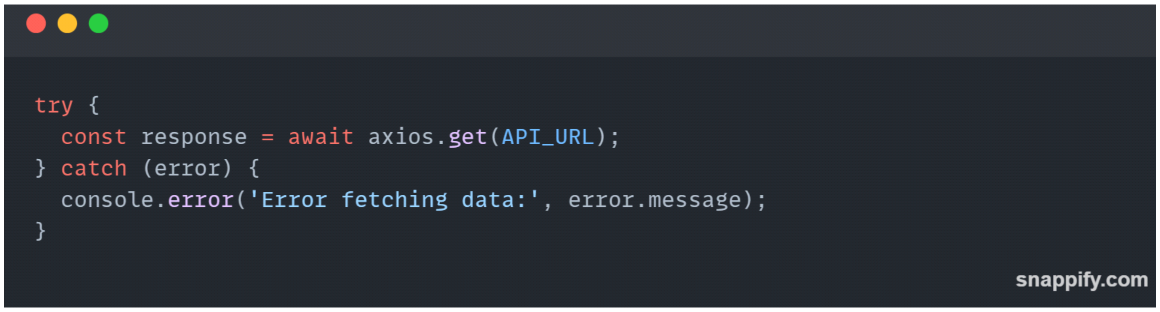
```
import { sanityClient } from '@/sanity/client';
export async function getServerSideProps() {
  const query = `*[_type == "car"]{
    name,
    brand,
    type,
    pricePerDay,
    image{
      asset->{
        url
      }
    }
  }`;
  const cars = await sanityClient.fetch(query);
  return {
    props: { cars },
  };
}
const CarsPage = ({ cars }) => (
  <div className="container">
    {cars.map((car, index) => (
      <div key={index} className="car-item">
        <h2>{car.name}</h2>
        <p>{car.brand}</p>
        <p>${car.pricePerDay} / day</p>
        <img src={car.image.asset.url} alt={car.name} />
      </div>
    ))}
  </div>
);
export default CarsPage;
```

snappify.com

Error Handling:

To ensure smooth user experience:

Log API errors


```
try {
  const response = await axios.get(API_URL);
} catch (error) {
  console.error('Error fetching data:', error.message);
}
```

Use fallback UI components like skeleton loaders for data loading states.

Frontend Features:

Car Information: Name, brand, type, price, and other details dynamically displayed.

Image Rendering: Car images fetched from Sanity are shown using optimized Next.js Image components.

Responsive Design: The layout adapts seamlessly across devices.

Conclusion:

The integration of the car rental API with Sanity CMS and the frontend application was successfully completed. Data is dynamically fetched, uploaded, and displayed with ease, providing a flexible content management system and an engaging user interface. Future improvements could include advanced filtering options and enhanced error-handling mechanisms.