Hackathon

Day 3 - Report

Prepared By

ABDUL RAHMAN MOIN

Day-3

API Integration Report

Introduction

Project Overview:

In this project, I have successfully integrated a car rental API with a frontend application, leveraging Sanity CMS to streamline car rental management. By dynamically uploading car details, including specifications like capacity, transmission, and fuel type, via the API, I've ensured seamless storage and accessibility within Sanity. This integration not only optimizes data handling but also powers a responsive and intuitive frontend, complete with advanced filtering, sorting, and categorization capabilities to enhance the user experience.

API Integration:

API Used:

The information is sourced from a specialized API endpoint designed to deliver detailed data on a wide range of cars. The API URL is as follows:

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

This API offers comprehensive details, including:

- Car's ID
- Car's Name
- Car's Rent
- Car's Image
- Car's Category
- Car's Fuel Capacity
- Car's Seating Capacity
- Car's Transmission

Migration Tools Used:

I referred to Sir Ali Jawwad's guidance blog for migrating data from an API to Sanity. Here's the link to the blog:

https://alijawwad001.atlassian.net/wiki/external/OWVkZTc1NjExMjU1NDljNmFkMGJhZTUxYmVmMWNmMzc

API Calls SnapShot:

CarouselSection.tsx:

```tsx
const [isError, setIsError] = useState('')
 const [carouselCardData, setCarouselCardData] = useState([])

 useEffect(() => {
     const fetchingCars = async () => {
         try {
             const data = await client.fetch(`*[_type == "car" && "popular" in tags]`)
             setCarouselCardData(data)
         } catch (err: any) {
             console.log("error: ", err);
             setIsError(err.message)
         }
     }

     fetchingCars()
 }, [])
```

RecommandedCars.tsx

```tsx

const [isError, setIsError] = useState('')
const [RecommandedCarsData, setRecommandedCarsData] = useState([])

useEffect(() => {
    const fetchingCars = async () => {
        try {
            const data = await client.fetch(`*[_type == "car" && "recommended" in tags]`)
            setRecommandedCarsData(data)
        } catch (err: any) {
            console.log("error: ", err);
            setIsError(err.message)
        }
    }

    fetchingCars()
}, [])
```

app/car-detail/[id]/page.tsx:

```tsx
const carId = props.params.id

const [isError, setIsError] = useState('')
const [carData, setCarData] = useState([])

useEffect(() => {
    const fetchingCar = async () => {
        try {
            const data = await client.fetch(`*[_type == "car" && _id == $carId]`, { carId })
            setCarData(data)
        } catch (err: any) {
            console.log("error: ", err);
            setIsError(err.message)
        }
    }

    fetchingCar()
}, [])
```

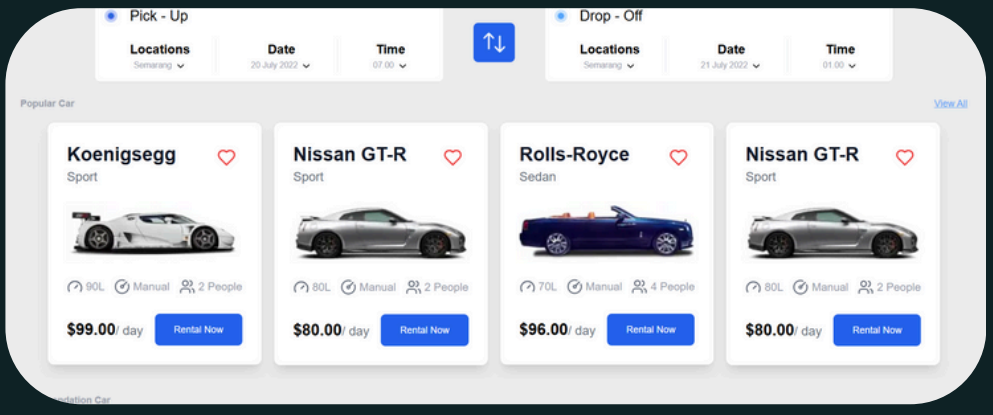
app/category/page.tsx:

```tsx
const [isError, setIsError] = useState('')
const [RecommandedCarsData, setRecommandedCarsData] = useState([])

useEffect(() => {
    const fetchingCars = async () => {
        try {
            const data = await client.fetch(`*[_type == "car" && "recommended" in tags]`)
            setRecommandedCarsData(data)
        } catch (err: any) {
            console.log("error: ", err);
            setIsError(err.message)
        }
    }

    fetchingCars()
}, [])
```

Frontend Page:

The frontend page displays the car rental details fetched from Sanity CMS. Below is an example of how the page appears:

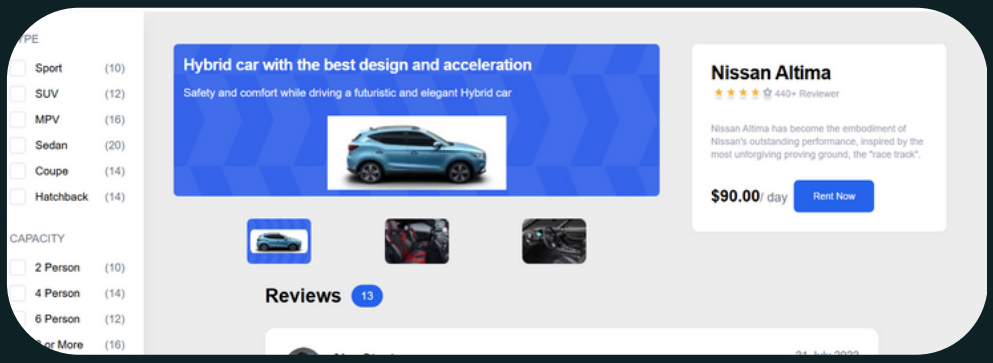
In this reference, we can clearly see the car's image, name, rent, and other essential details like fuel capacity, seating capacity, and transmission type.



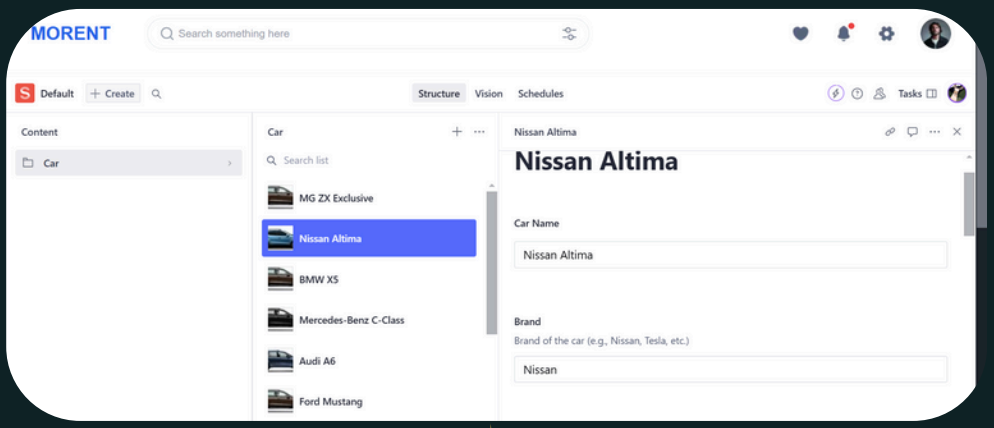
Daynamic Page:

The frontend page displays the car rental details fetched from Sanity CMS. Below is an example of how the page appears:

In this reference, we can clearly see the car's image, name, rent, and other essential details like fuel capacity, seating capacity, and transmission type.



Populated Sanity CMS Fields:



Code Snippets:

API Integration:

```jsx
<div
                    className="carousel carousel-start bg-transparent rounded-box
                max-w-full lg:flex lg:justify-center space-x-4 p-4"
                >
                    {carouselCardData.map((item: any) => (
                        <Link href={`/car-detail/${item._id}`}
                        >
                            <Card
                                key={item._id} carId={item._id}
                                name={item.name} category={item.type}
                                image={urlFor(item.image).url()} fuelCapacity={item.fuelCapacity}
                                transmission={item.transmission} capacity={item.seatingCapacity}
                                price={item.pricePerDay} />
                        </Link>

                    ))}
                </div>
```

Migration Script:

```javascript
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;

      // Upload image and get asset reference if it exists
      if (car.image_url) {
        imageRef = await uploadImageToSanity(car.image_url);
      }

      const sanityCar = {
        _id: `car-${car.id}`, // Prefix the ID to ensure validity
        _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,
      };

      // Log the product before attempting to upload it to Sanity
      console.log('Uploading car object to Sanity:', sanityCar);

      // Import data into Sanity
      const result = await client.createOrReplace(sanityCar);
      console.log(`✅ Imported car: ${sanityCar.name}`);
      console.log(`Car uploaded successfully: ${result._id}`);
    }

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

Conclusion:

This project successfully demonstrates the integration of an API with Sanity CMS for managing car rental data. The process involved fetching data from the API, uploading images, and displaying the details dynamically on the frontend. This streamlined approach ensures data accuracy, enhances the user interface, and leverages the robust features of Sanity CMS to meet project requirements.