# Day 3 - API Integration Report for Morent

**1. Understanding the Provided API**

**1.1 API Documentation Review**

* The provided API contains car rental data, including fields such as:
  + id: Unique identifier for each car.
  + name: Name of the car (e.g., Koenigsegg, Nissan GT-R).
  + type: Type of car (e.g., Sport).
  + fuel\_capacity: Fuel capacity of the car (e.g., 90L).
  + transmission: Transmission type (e.g., Manual).
  + seating\_capacity: Number of seats (e.g., 2 People).
  + price\_per\_day: Rental price per day (e.g., $99.00).
  + image\_url: URL of the car image.
  + tags: Tags for categorization (e.g., "popular").

**1.2 Key Endpoints**

* **Product Listings**: /api/hackathon/template7 (to fetch all cars).
* **Categories**: Not explicitly provided, but car types (e.g., Sport, SUV) can be derived from the type field.
* **Order History**: Not applicable for this API.

Endpoints overview has been provided in the documentation of api specifications.

**2. Validating and Adjusting the Schema**

**2.1 Comparison of API Data and Sanity Schema**

* **API Fields**:
* {
* "id": 1,
* "name": "Koenigsegg",
* "type": "Sport",
* "fuel\_capacity": "90L",
* "transmission": "Manual",
* "seating\_capacity": "2 People",
* "price\_per\_day": "$99.00",
* "image\_url": "https://example.com/car.jpg",
* "tags": ["popular"]

}

* **Sanity Schema Fields**:
* export default {
* name: 'car',
* type: 'document',
* fields: [
* { name: 'name', type: 'string', title: 'Name' },
* { name: 'type', type: 'string', title: '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', of: [{ type: 'string' }], title: 'Tags' },
* { name: 'image', type: 'image', title: 'Car Image' },
* ],

};

**2.2 Schema Adjustments**

* Adjusted the schema to match the API data:
  + name: Mapped directly.
  + type: Mapped directly.
  + fuel\_capacity: Mapped to fuelCapacity.
  + transmission: Mapped directly.
  + seating\_capacity: Mapped to seatingCapacity.
  + price\_per\_day: Mapped to pricePerDay.
  + image\_url: Mapped to image after uploading to Sanity.
  + tags: Mapped directly.

**3. Data Migration**

**3.1 Migration Script**

* Used the provided migration script to fetch data from the API and upload it to Sanity CMS.

**3.2 Migration Steps**

1. **Set Up Environment Variables**:
   * Created a .env.local file with the following variables:

NEXT\_PUBLIC\_SANITY\_PROJECT\_ID=your-project-id

NEXT\_PUBLIC\_SANITY\_DATASET=your-dataset

SANITY\_API\_TOKEN=your-api-token

1. **Run the Migration Script**:
   * Compiled the TypeScript file using:

tsc

* + Executed the script using:

node importData.js

1. **Verify Data in Sanity CMS**:
   * Checked the Sanity Studio to ensure all car data was imported correctly.
   * Verified that images were uploaded and linked to the respective car entries.

**3.3 Tools Used**

* **Sanity CLI**: For data import.
* **Axios**: For fetching data from the API.
* **Node.js**: For running the migration script.

**4. API Integration in Next.js**

**4.1 Utility Functions**

* Created utility functions to fetch data from the API and Sanity CMS.

**4.2 Rendering Data**

* Displayed data in components such as product listings and car details.

**4.3 Testing**

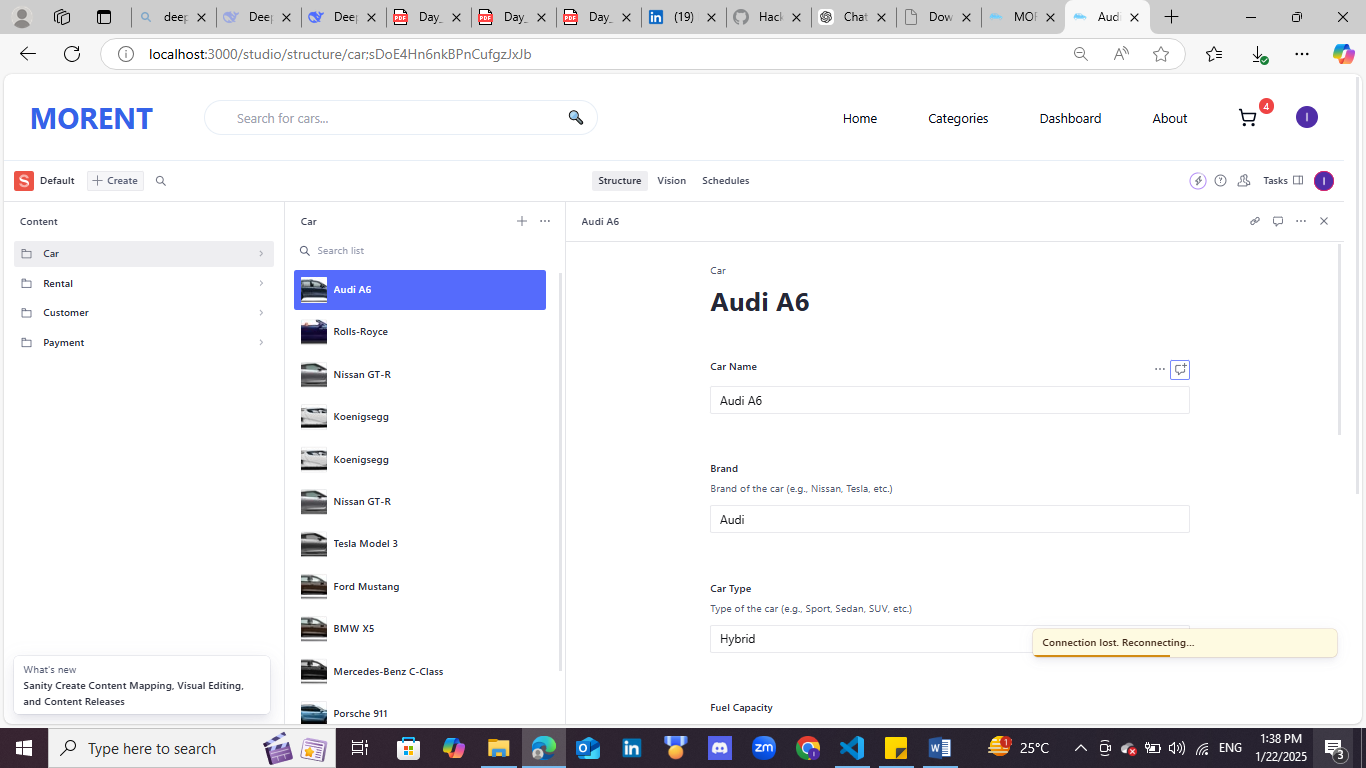
* Tested API endpoints using Postman and browser developer tools.

**5. Expected Output**

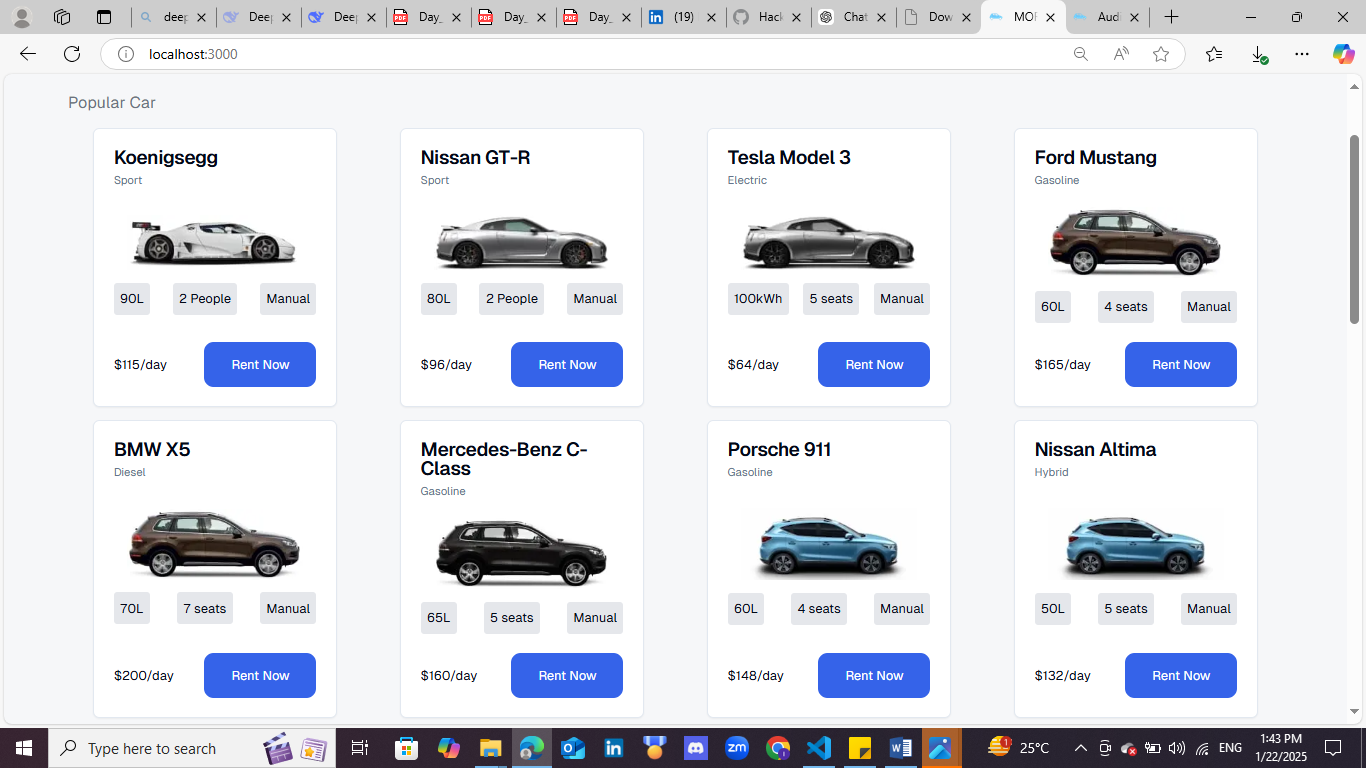
* **Sanity CMS**: Successfully populated with car data.
* **Next.js Frontend**: Functional integration displaying product listings with accurate data.

**Submission Attachments**

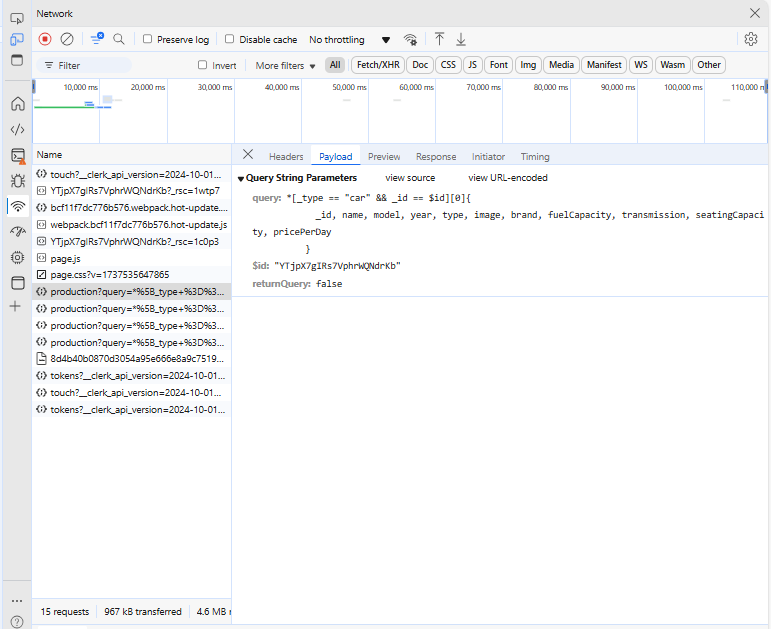
## Screenshot of populated schema:



## Screenshot of data successfully displayed in frontend:



## Screenshot of API calls:



## Code snippets for API integration:

export default function Home() {

  const [cars, setCars] = useState<Car[]>([]);

  const searchParams = useSearchParams();

  const searchQuery = searchParams.get('search') || '';

  useEffect(() => {

    const fetchCars = async () => {

      let query = '\*[\_type == "car"]{\_id, name, type, image, pricePerDay, fuelCapacity, transmission, seatingCapacity}';

      const data = await client.fetch(query);

      setCars(data);

    };

    fetchCars();

  }, []);

   const filteredCars = cars.filter((car) =>

    car.name.toLowerCase().includes(searchQuery.toLowerCase())

  );

import { NextResponse } from 'next/server';

import { client } from '../../../sanity/lib/client';

import { groq } from 'next-sanity';

const query = groq`\*[\_type == "car"]{

  \_id,

  name,

  type,

  image,

  pricePerDay,

  fuelCapacity,

  transmission,

  seatingCapacity

}`;

export async function GET() {

  try {

    const cars = await client.fetch(query);

    return NextResponse.json(cars);

  } catch (error) {

    return NextResponse.json({ message: 'Error fetching data' }, { status: 500 });

  }

}

## Migration Scripts:

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();