Day 3- API Integration Report for Morent Car Rental Service

This report documents the API integration process for our luxury car rental marketplace, including schema adjustments, API testing, frontend integration, and database migrations. Additionally, screenshots of API calls, frontend data display, and populated Sanity CMS fields are provided.

1) API integration process.

a) Understanding the API

- Reviewed the provided API documentation for the assigned template # 7.
- Identified key endpoints:
- **Cars Endpoint:** /template7
- Assigned template has provided following detail, see picture below
  - **Id ➔ type number**
  - **Name ➔ type string ➔** (name of a car)
  - **Type ➔ type string ➔** (name of car type)
  - **Fuel capacity ➔ type string ➔** (fuel car can hold at max)
  - **Transmission ➔ type string ➔** (what's responsible for driving power from the engine to the wheels)
  - **Seating capacity in car ➔ type string ➔** (no of peoples can sit in car)
  - **Price per day ➔ type string ➔** (discounted rent per day)
  - **Original Price ➔ type string ➔** (original rent per day)
  - **Image ➔ type image ➔** (image of a car)
  - **Tags ➔ type array ➔** (tags for categorization)

Below is the image of API which provided for template # 07


```
{
  "id": 2,
  "name": "Nissan GT-R",
  "type": "Sport",
  "fuel_capacity": "80L",
  "transmission": "Manual",
  "seating_capacity": "2 People",
  "price_per_day": "$80.00",
  "original_price": "$100.00",
  "image_url": "https://car-rental-website-five.vercel.app/_next/image?url=%2F_next%2Fstatic%2Fmedia%2Fcar(1).cab606a9.jpg&w=640&q=75",
  "tags": [
    "popular"
  ]
},
```


b) Adjustment made in Schema

Based on our business model, the following adjustments were made to the car schema:

- Added **slug** property in the schema which type is slug
- Added **brand** key in schema to give brand name to cars as well
- Added **rating** and **rating count** key whose type is number in order to give insights about the progress of cars
- Modified **type** (of car) to **car type** to give it more descriptive perspective
- Modified type of **price per day** to **number** from string

- Modified type of **original price** to **number** from string
- Modified type of **images** as **array** because we have to show more pictures in specific car section.
- Modified type of **tags** to **string** from array to align it with my requirement
- Modified few cars **images** and add those images which are relevant to brand name, for example if car name is Nissan GTR Skyline, so I used Skyline picture for that car.
- Overall, I have made a fake API from https://mockapi.io/ to make said changes in API and used that schema to fetch data from API endpoint. Below is the API endpoint from which I fetched data https://678aeec7dd587da7ac2bbedb.mockapi.io/cars

Below is the picture of car schema which I made for sanity CMS.

c) Migration steps and tools used

- Method Used: **Script-Based Migration**
- Wrote a script in JavaScript to fetch and transform data from the API.
- This script fetches the data from API endpoint and create client in order to send it on Sanity CMS, see below pic

- Fetched car data from the API endpoint https://678aeec7dd587da7ac2bbedb.mockapi.io/cars using **axios**.
- Uploaded car images to Sanity CMS using **client.assets.upload**.
- Transformed the API data to match the Sanity schema

2) Screenshots

1) API calls.

scripts > importSanityData.mjs > uploadImageToSanity > asset > filename

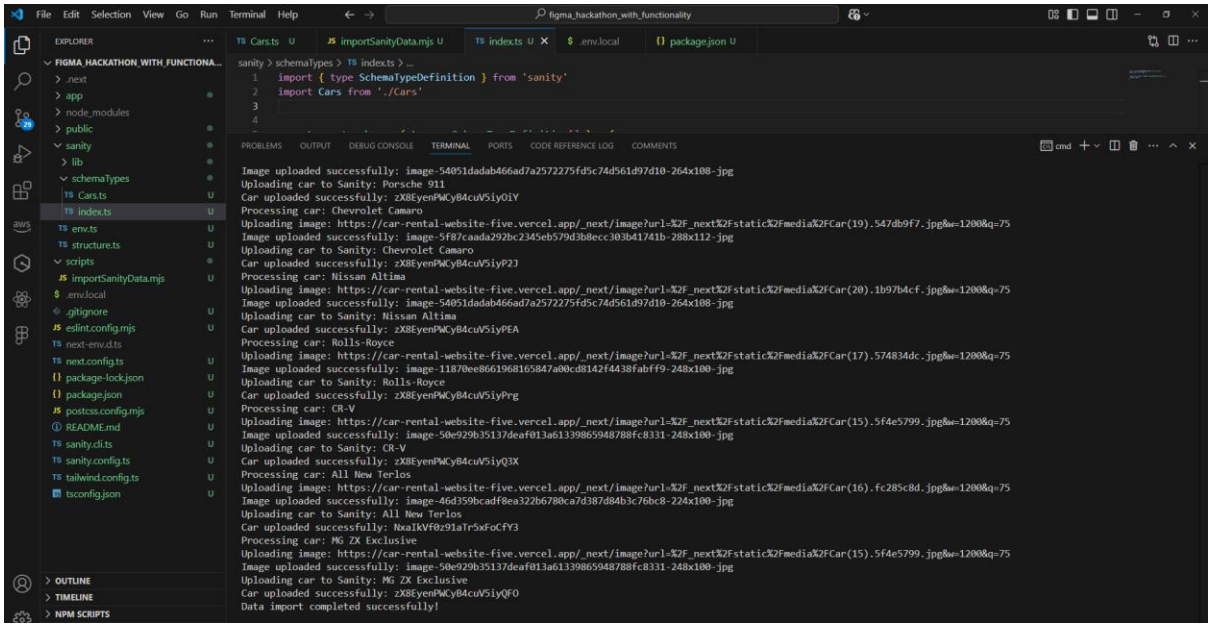
```
import { createClient } from '@sanity/client';
import axios from 'axios';
import dotenv from 'dotenv';
import { fileURLToPath } from 'url';
import path from 'path';
import { v4 as uuidv4 } from 'uuid'; // Install uuid with `npm install uuid`


// 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.NEXT_PUBLIC_SANITY_TOKEN,
  apiVersion: '2025-01-18'
});

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://678aeec7dd587da7ac2bbedb.mockapi.io/cars');
    const cars = response.data;
```

```
    // API endpoint containing car data
    const response = await axios.get('https://678aeec7dd587da7ac2bbedb.mockapi.io/cars');
    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,
        carType: car.type,
        gasoline: car.fuel_capacity,
        steering: car.transmission,
        personCapacity: car.seating_capacity,
        rent: car.price_per_day,
        previousRent: car.original_price || null,

        tags: car.tags || "",
        images: imageRef
        ? [
            {
              _type: 'image',
              _key: uuidv4(), // Generate a unique key for the image
              asset: {
                _type: 'reference',
                _ref: imageRef,
              },
            },
          ]
        : [],
      };

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

```
    const sanityCar = {
      _type: 'car',
      name: car.name,
      brand: car.brand || null,
      carType: car.type,
      gasoline: car.fuel_capacity,
      steering: car.transmission,
      personCapacity: car.seating_capacity,
      rent: car.price_per_day,
      previousRent: car.original_price || null,

      tags: car.tags || "",
      images: imageRef
      ? [
        {
          _type: 'image',
          _key: uuidv4(), // Generate a unique key for the image
          asset: {
            _type: 'reference',
            _ref: imageRef,
          },
        },
      ]
      : [],
    };

    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();
```

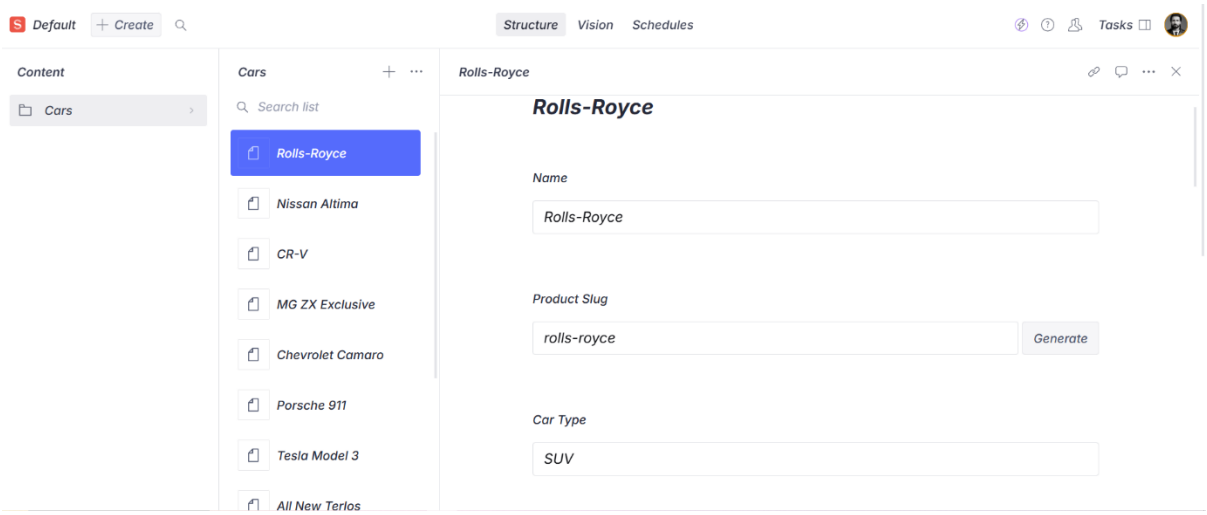
2) Populated Sanity CMS fields

- Added that script file in package.json file, under script section then I ran the **npm run import data** command to trigger the import process and inject the data into Sanity, see below picture data is sending to Sanity CMS.



- After integrating the new schema, the Sanity CMS dashboard reflects the updated fields:

  ✅ Newly added cars are properly structured in the CMS.
- Below is the picture when I got data from API using Axois into sanity.



3) Data successfully displayed in the frontend

- To get the car data from Sanity and show it on the frontend, I used **groq query** to fetch data from sanity and display it on **Nextjs** frontend. I pulled the data from Sanity and made it available on the homepage. The data is then displayed as car cards on the homepage, showing information like the car's name, brand, fuel capacity, transmission, and more as shown in the picture below.