Day 3 - API Integration Report - Khalil Luxury Rentals

Prepared by: KHALIL-UR-REHMAN

API Integration Process

Steps:

1. ***Environment Setup****:*
    - *Environment variables loaded using dotenv from .env.local.*

    I used dotenv to load environment variables from .env.local, ensuring sensitive data like API tokens are not hardcoded in the script. This helps in securely managing secrets such as project Id, dataset, and token.

    - *Sanity client configured with project ID, dataset, and API version.*

    I created a Sanity client using the create Client function from @sanity/client. Configured with projectId, dataset, token, use Cdn, and api Version. This setup is essential for authenticating and interacting with the Sanity CMS.

2. ***API Endpoint****:*
    - *Data fetched from https://sanity-nextjs- application. Vercel .app /api /hackathon /template7.*

    I integrated API to fetch car data. Each car's data is processed, and necessary transformations are applied (e.g., handling image references and default values for missing fields).

3. ***Data Processing****:*
    - *Images uploaded to Sanity using the upload Image To Sanity function.*
    - *Car data transformed and uploaded to Sanity as documents of type car.*
4. ***Error Handling****:*
    - *Try-catch blocks used to handle errors during data fetching and image uploading.*

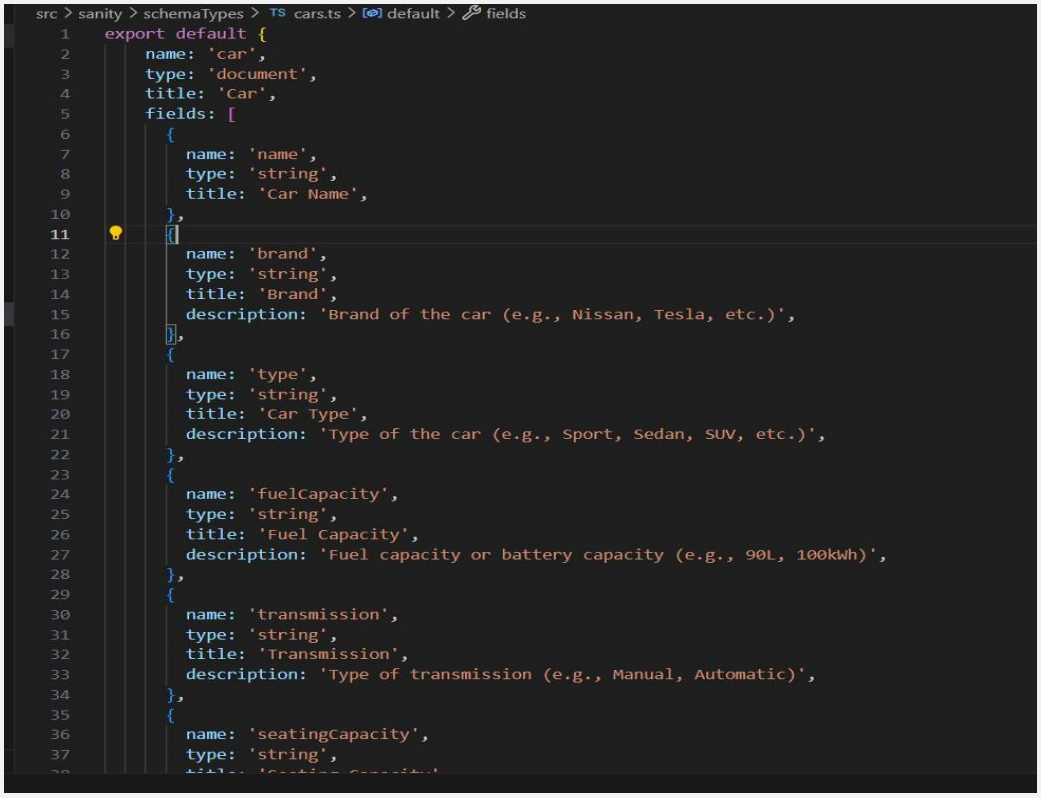
Adjustments Made to Schemas

- ***Car Schema****: Defined with fields such as name, brand, type, fuel Capacity, transmission, seating Capacity, price Per Day, original Price, tags, and image.*

Each car object is mapped to a corresponding Sanity document with defined fields such as name, brand, type, etc. I ensured that optional fields (like original Price, tags) are handled gracefully with default values

Schema Definition Example (cars.ts):

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



Migration Steps and Tools Used

1. ***Data Fetching****:*
    - *Data retrieved from the specified API endpoint using axios.*

2. ***Image Upload****:*
    - *Images downloaded as buffers and uploaded to Sanity using the assets.upload method.*

3. ***Data Upload****:*

- *Each car's data created in Sanity as a document using the client.create method.*

*Each processed car object is uploaded to the Sanity dataset using **client.create(sanityCar)**. This action integrates the fetched and processed data into your Sanity CMS, completing the migration step.*