HACKATHON DAY - 3

API INTEGRATION AND DATA MIGRATION (RENTAL E-COMMERCE)

Overview:

This document process the Integration of Sanity CMS API into a Next.js app for managing and displaying data.The main objective of this phase is to integratebthe Sanity CMS API for migrating product data as well as to adjust the schema to organizw the product information.The process includes setting up the connection between backend and frontend enabling efficient data retrieval and visualization.

Sanity API Integration:

Following steps used for seamless integration:

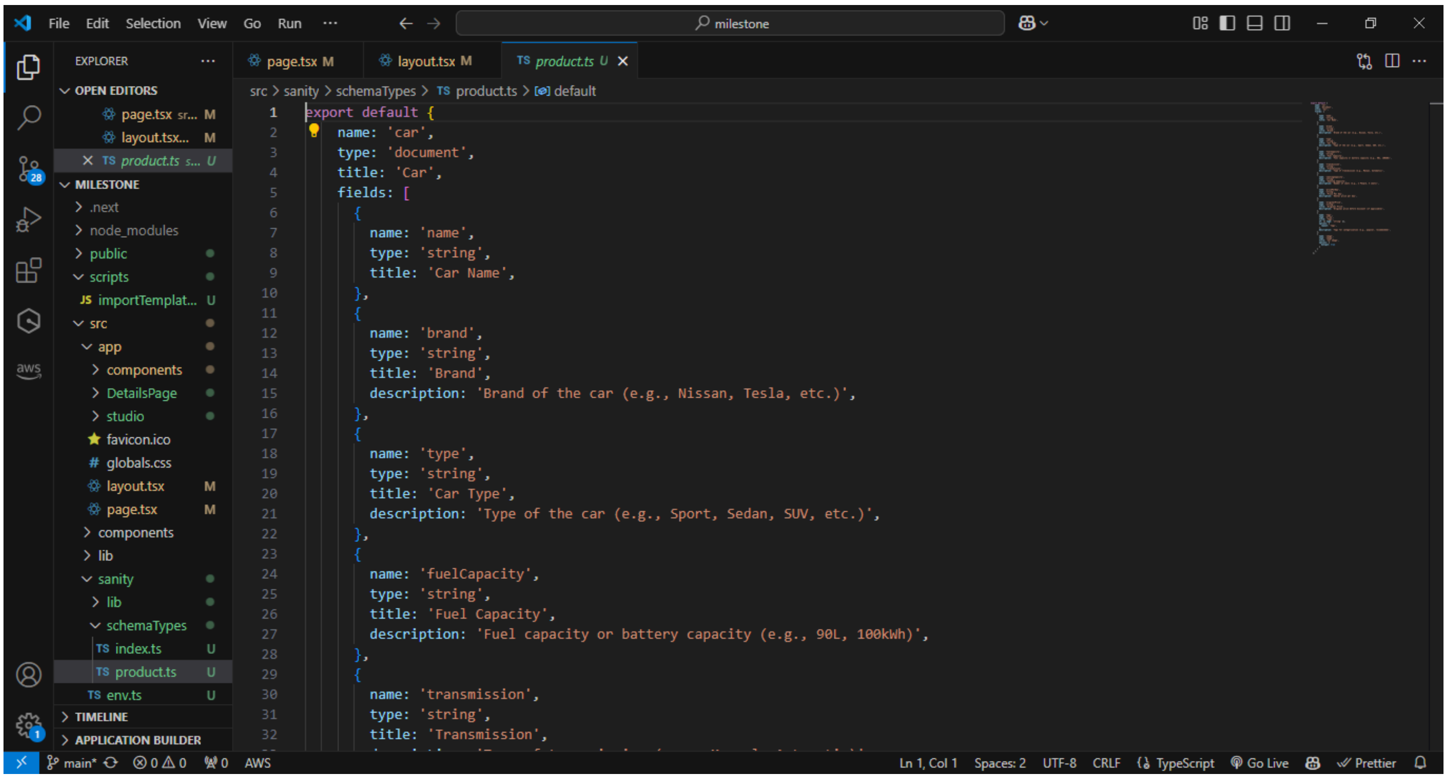
Fetching Data from External API:

- Made a custom importTemplate7Data.mjs to fetch product data.
- Access the following endpoints:sanity-nextjs-application.vercel.app/api/hackathon/template7

Schema Validation:

Now,make a schema where we describe all products data which manage specific data e.g.Product name,description,price,discounted price,image and category.

this schema ensures that product data is structured properly for easy querying and retrieval.

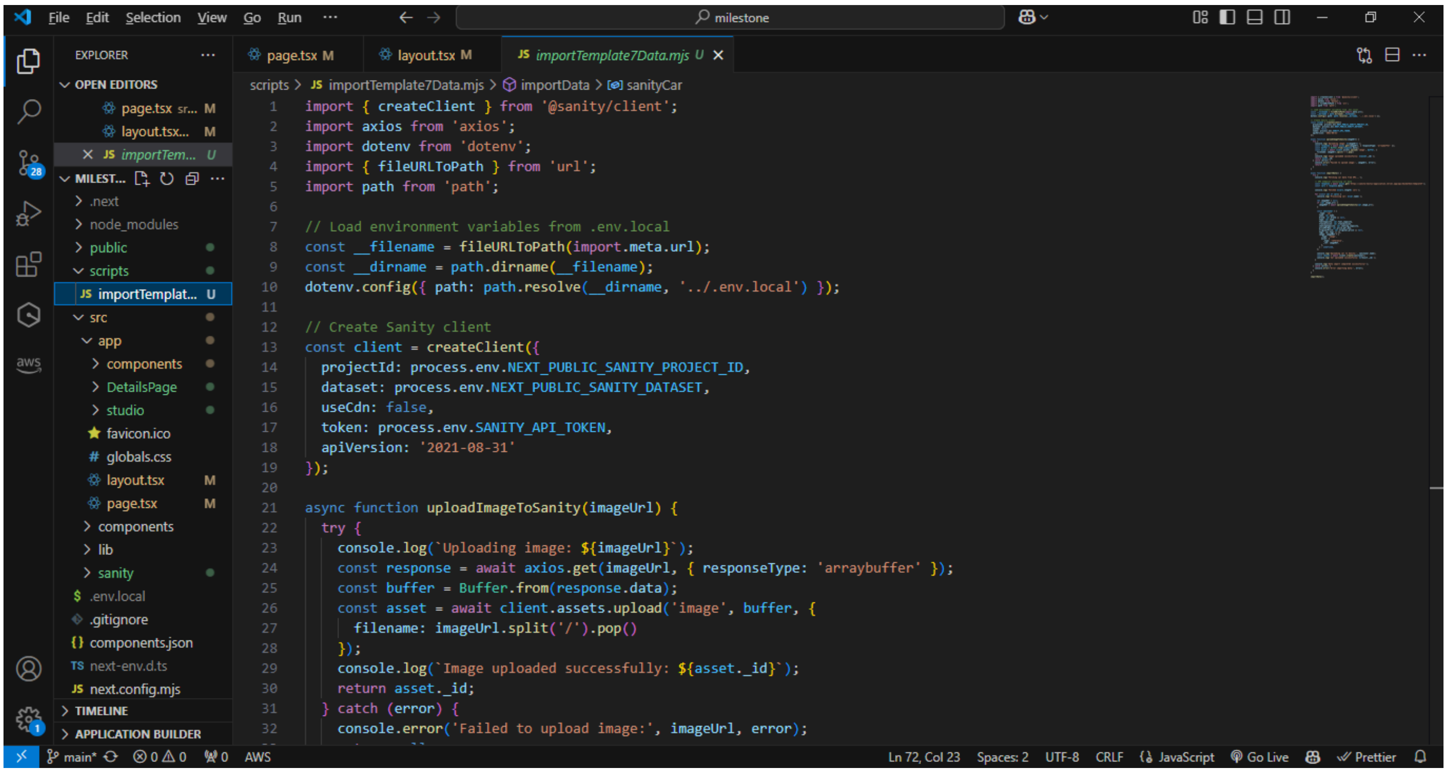


```
export default {
  name: 'car',
  type: 'document',
  title: 'Car',
  fields: [
    {
      name: 'name',
      type: 'string',
      title: 'Car Name',
    },
    {
      name: 'brand',
      type: 'string',
      title: 'Brand',
      description: 'Brand of the car (e.g., Nissan, Tesla, etc.)',
    },
    {
      name: 'type',
      type: 'string',
      title: 'Car Type',
      description: 'Type of the car (e.g., Sport, Sedan, SUV, etc.)',
    },
    {
      name: 'fuelCapacity',
      type: 'string',
      title: 'Fuel Capacity',
      description: 'Fuel capacity or battery capacity (e.g., 90L, 100kWh)',
    },
    {
      name: 'transmission',
      type: 'string',
      title: 'Transmission',
```

Data Migration:

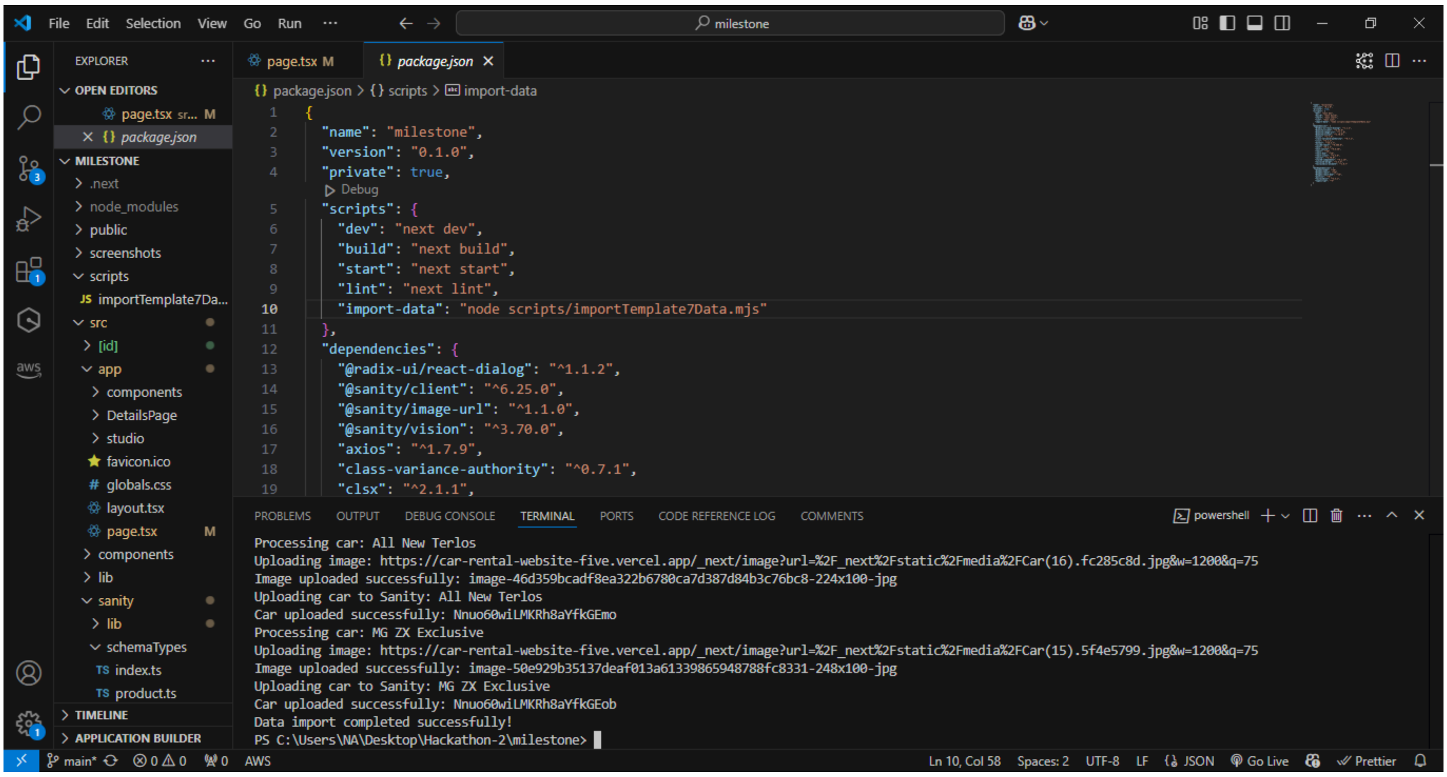
The next step is to make a migration script that would do the process of importing the fetch data into sanity.We did the following steps:

- Fetch data from API.
- Use the clienty to push the data into Sanity.
- Now import this scripts/importTemplate7Data.mjs in package.json file.
  import-data: “node scripts/importTemplate7Data.mjs.
- Run the following command to import data into Sanity;
  npm run import-data



```javascript
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);
```

Now,the data is successfully imported into Sanity,as we can see here;



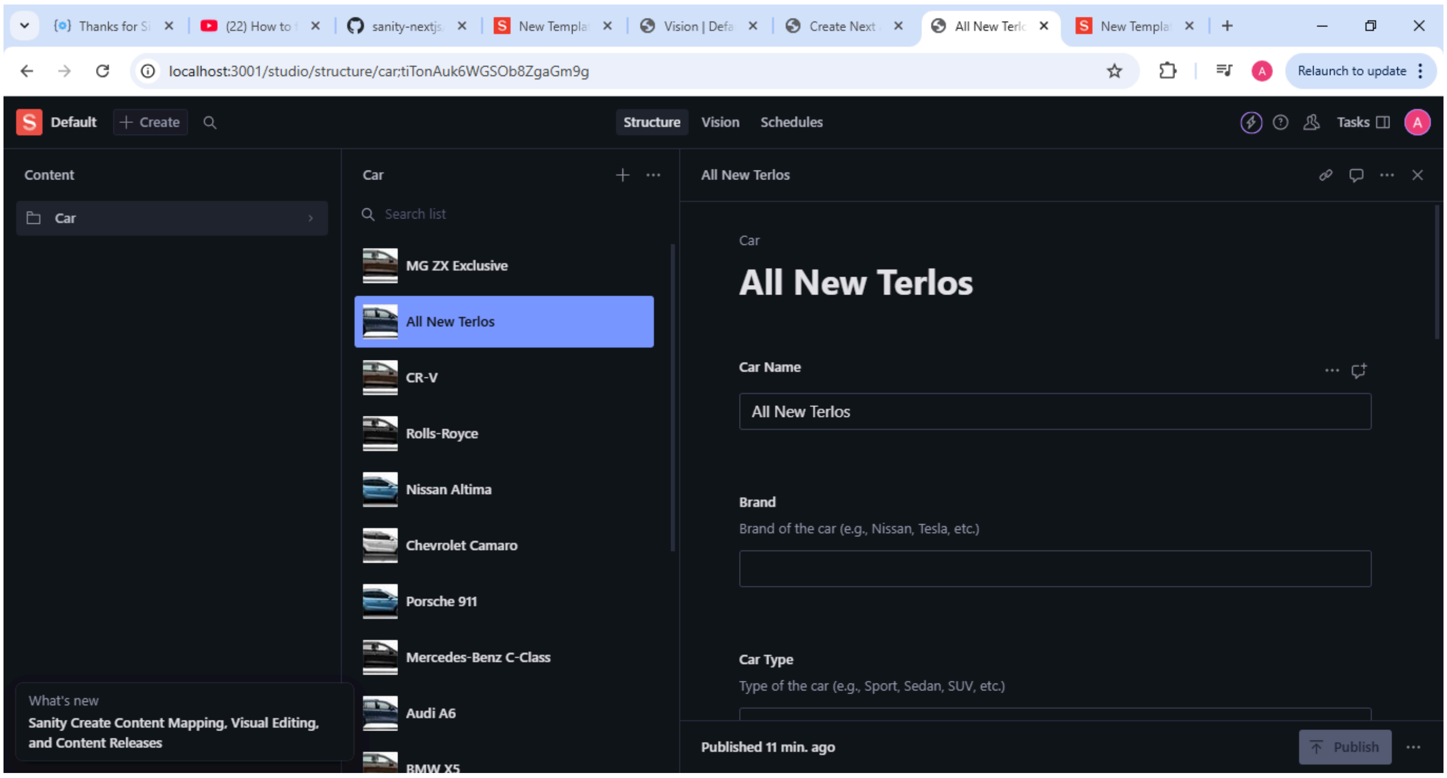
```json
{
  "name": "milestone",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint",
    "import-data": "node scripts/importTemplate7Data.mjs"
  },
  "dependencies": {
    "@radix-ui/react-dialog": "^1.1.2",
    "@sanity/client": "^6.25.0",
    "@sanity/image-url": "^1.1.0",
    "@sanity/vision": "^3.70.0",
    "axios": "^1.7.9",
    "class-variance-authority": "^0.7.1",
    "clsx": "^2.1.1",
```

```
Processing car: All New Terlos
Uploading image: https://car-rental-website-five.vercel.app/_next/image?url=%2F_next%2Fstatic%2Fmedia%2FCar(16).fc285c8d.jpg&w=1200&q=75
Image uploaded successfully: image-46d359bcadf8ea322b6780ca7d387d84b3c76bc8-224x100-jpg
Uploading car to Sanity: All New Terlos
Car uploaded successfully: Nnuo60wiLMKRh8aYfkGEmo
Processing car: MG ZX Exclusive
Uploading image: https://car-rental-website-five.vercel.app/_next/image?url=%2F_next%2Fstatic%2Fmedia%2FCar(15).5f4e5799.jpg&w=1200&q=75
Image uploaded successfully: image-50e929b35137deaf013a61339865948788fc8331-248x100-jpg
Uploading car to Sanity: MG ZX Exclusive
Car uploaded successfully: Nnuo60wiLMKRh8aYfkGEob
Data import completed successfully!
PS C:\Users\NA\Desktop\Hackathon-2\milestone>
```

Verification Of Migrated Data:

Now,the data was finally migrated,we take following steps to verify if the data is completely fetched or not.

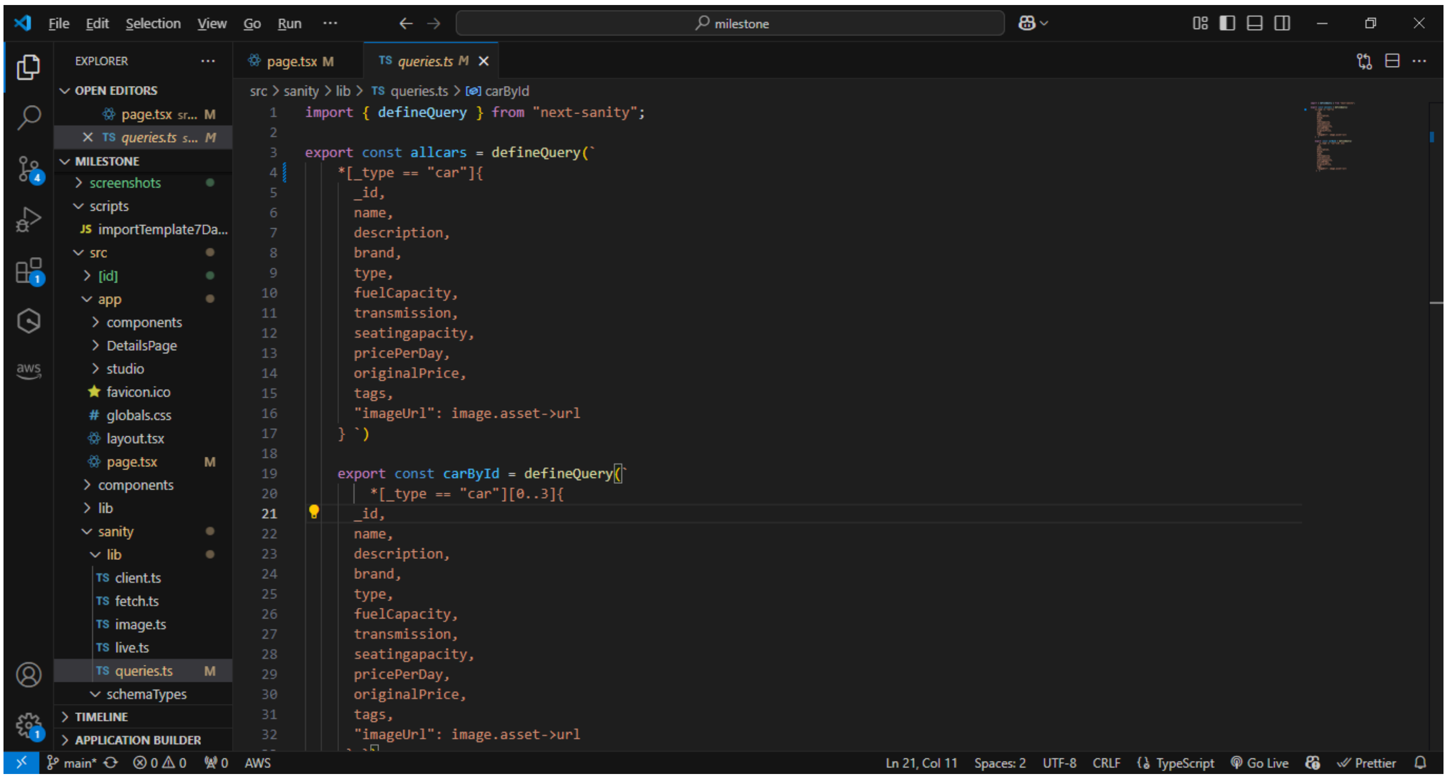
- Run the localhost:3000/studio/structure to access your sanity project and see the migrated data displayed in Sanity.



Sanity Data To FrontEnd:

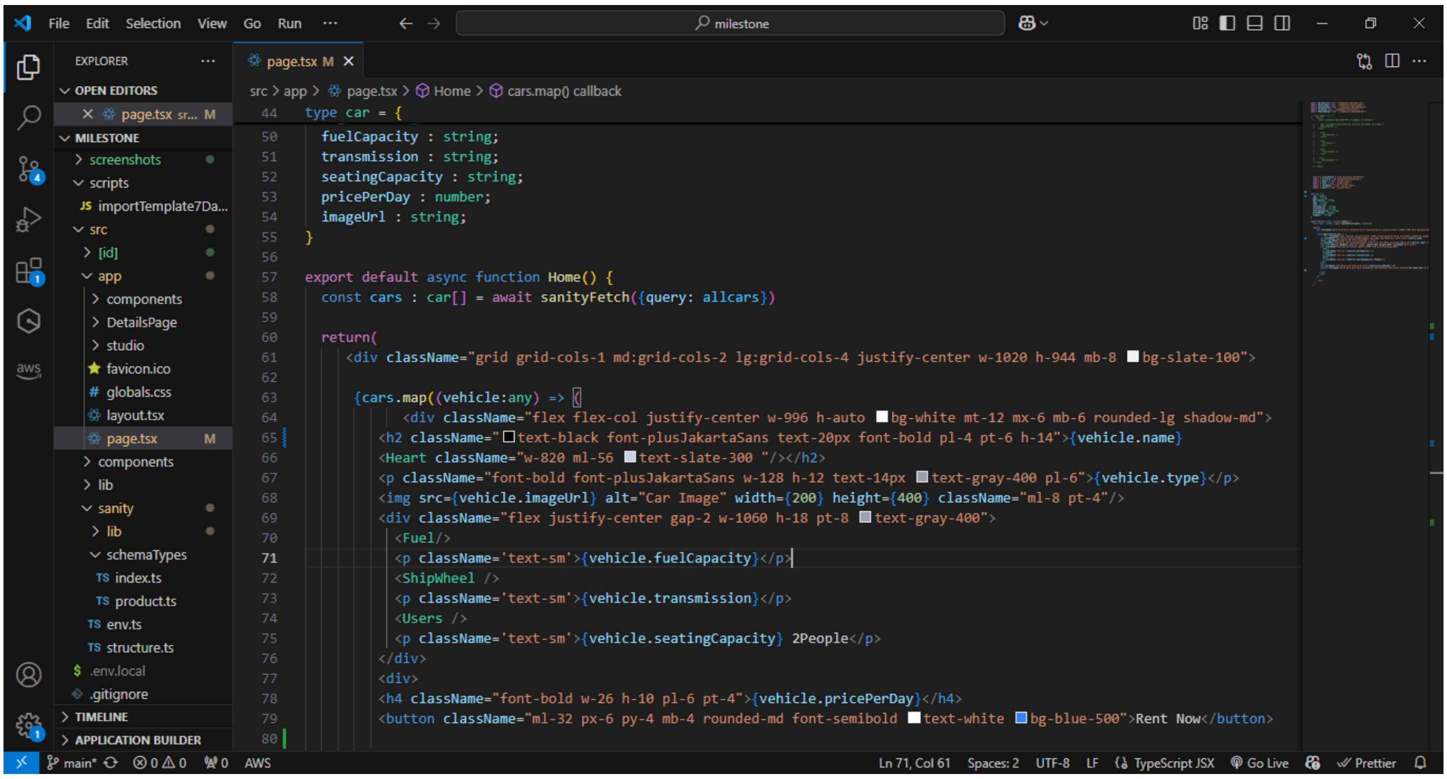
After the migration process we will shown this data on our Frontend by the help of GROQ Query.Steps taken:

- First in the lib folder of sanity folder we make queries.ts file and in this file we call our product data which we want with the help of GROQ Query as shown below:



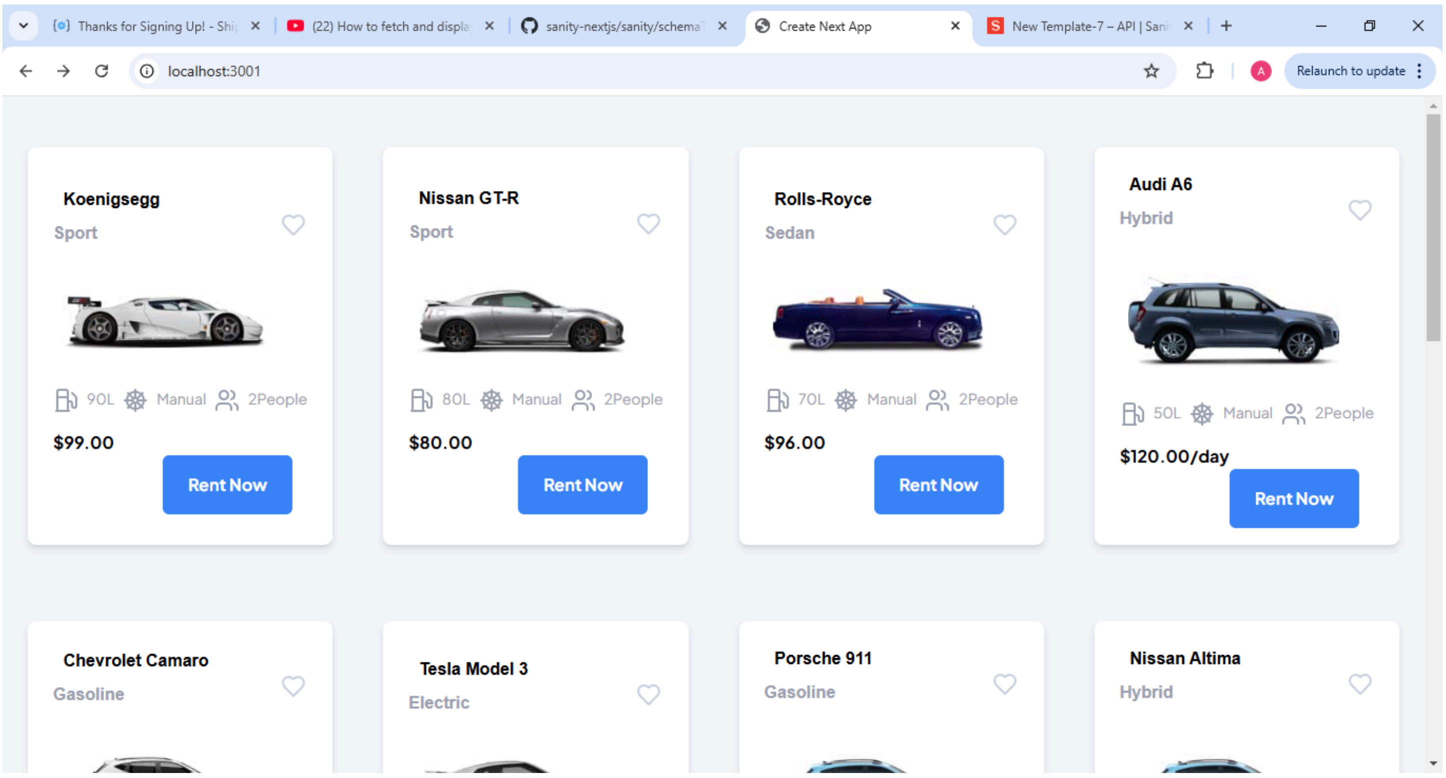
```ts
import { defineQuery } from "next-sanity";

export const allcars = defineQuery(`
  *[_type == "car"]{
    _id,
    name,
    description,
    brand,
    type,
    fuelCapacity,
    transmission,
    seatingapacity,
    pricePerDay,
    originalPrice,
    tags,
    "imageUrl": image.asset->url
  } `)

  export const carById = defineQuery(`
    *[_type == "car"][0..3]{
    _id,
    name,
    description,
    brand,
    type,
    fuelCapacity,
    transmission,
    seatingapacity,
    pricePerDay,
    originalPrice,
    tags,
    "imageUrl": image.asset->url
```

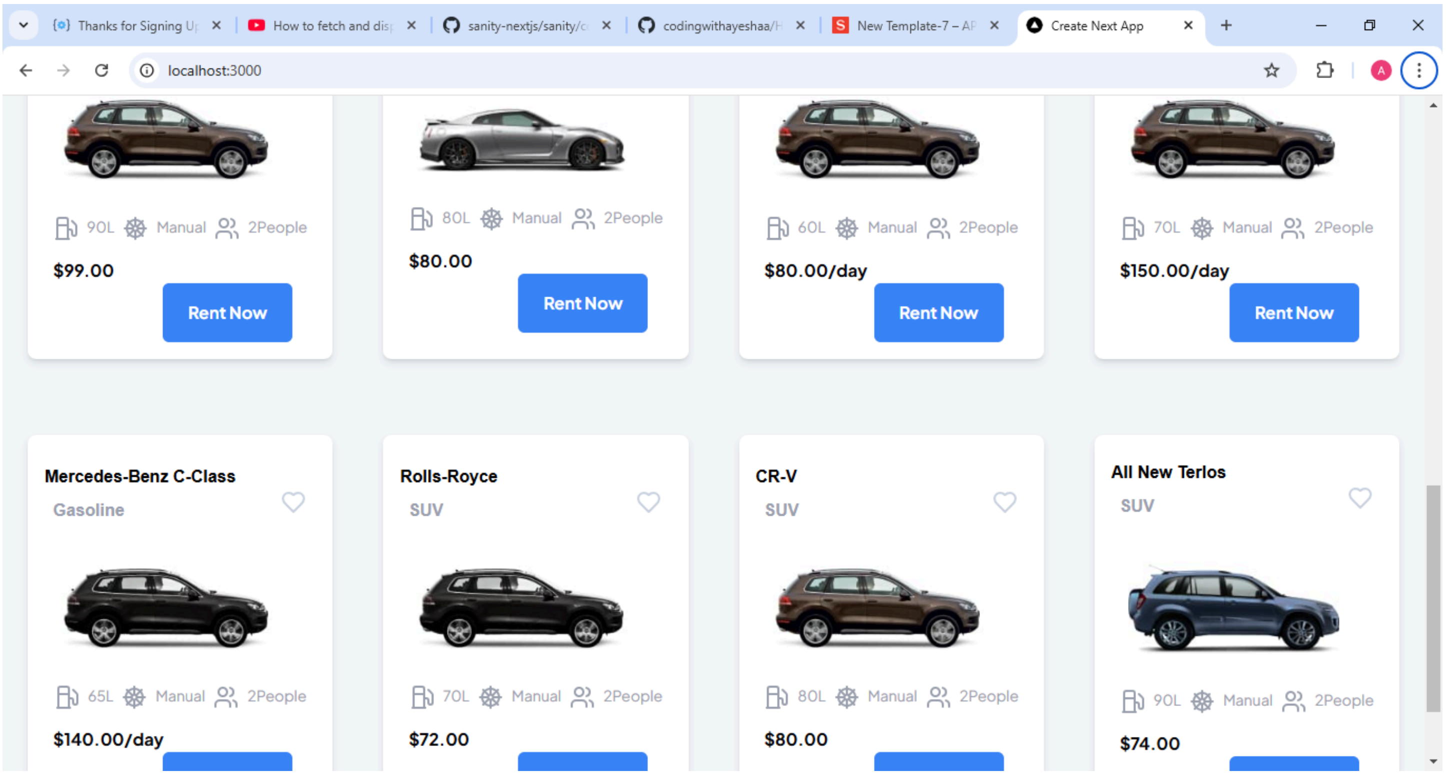
- Second step is to do some coding to make your frontend attractive .Here in my main page.tsx file we will import this data with the help of GROQ query we made above.
- Now Our FrontEnd will look like this.
- Successfully we imported the migrated data froom Sanity to Frontend.



```tsx
type car = {
  fuelCapacity : string;
  transmission : string;
  seatingCapacity : string;
  pricePerDay : number;
  imageUrl : string;
}

export default async function Home() {
  const cars : car[] = await sanityFetch({query: allcars})

  return(
    <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 justify-center w-1020 h-944 mb-8 bg-slate-100">

     {cars.map((vehicle:any) => (
          <div className="flex flex-col justify-center w-996 h-auto bg-white mt-12 mx-6 mb-6 rounded-lg shadow-md">
       <h2 className="text-black font-plusJakartaSans text-20px font-bold pl-4 pt-6 h-14">{vehicle.name}
       <Heart className="w-820 ml-56 text-slate-300 "/></h2>
       <p className="font-bold font-plusJakartaSans w-128 h-12 text-14px text-gray-400 pl-6">{vehicle.type}</p>
       <img src={vehicle.imageUrl} alt="Car Image" width={200} height={400} className="ml-8 pt-4"/>
       <div className="flex justify-center gap-2 w-1060 h-18 pt-8 text-gray-400">
         <Fuel/>
         <p className='text-sm'>{vehicle.fuelCapacity}</p>
         <ShipWheel />
         <p className='text-sm'>{vehicle.transmission}</p>
         <Users />
         <p className='text-sm'>{vehicle.seatingCapacity} 2People</p>
       </div>
       <div>
       <h4 className="font-bold w-26 h-10 pl-6 pt-4">{vehicle.pricePerDay}</h4>
       <button className="ml-32 px-6 py-4 mb-4 rounded-md font-semibold text-white bg-blue-500">Rent Now</button>
```

Data Successfully Displayed On FrontEnd:





Conclusion:

This concludes our understanding of API Integration and its migration into Sanity and then from Sanity to Next.js through GROQ Query which is a challenging task.

Checklist For Day-1:

- [x] ~~Business Goals:~~
- [x] ~~Market Research:~~
- [x] ~~Data Schema Draft:~~
- [x] ~~Submission from Day 1:~~

Checklist For Day-2:

- [x] ~~Technical Plan:~~
- [x] ~~WorkFlows:~~
- [x] ~~API Requirements:~~
- [x] ~~Sanity Schema:~~
- [x] ~~Submission from Day 2:~~

Checklist For Day-3:

- [x] ~~API Understanding:~~
- [x] ~~Schema Validation:~~
- [x] ~~Data Migration:~~
- [x] ~~API Integration In Next.js:~~
- [x] ~~Submission from Day 3:~~