DAY 3 - API INTEGRATION AND DATA MIGRATION

Introduction

This documentation outlines the process of integrating an API for a Car Rental Project into a Sanity CMS. It details the following steps

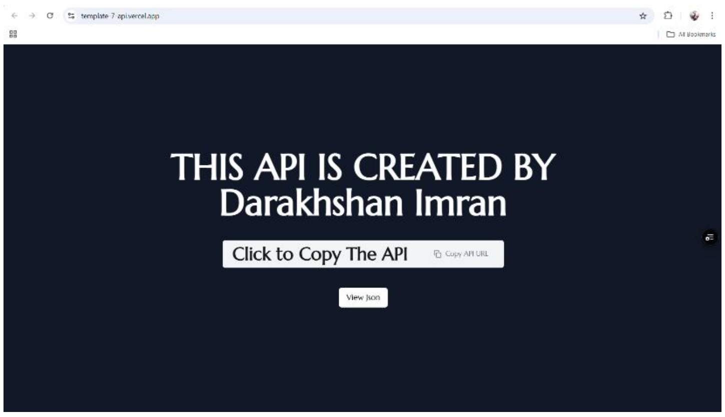
1. Creating an API
2. Setup up a clean Sanity project.
3. Defining a schema.
4. Importing API data.
5. Fetching it using GROQ queries for local development.

A. Creating an API

Define Your Data Structure:

Create an API that represents your rental car data. Each car object should include attributes like id, make, model, year, price, transmission, mileage, seating capacity and any other relevant details.

https://template-7-api.vercel.app/api/cars



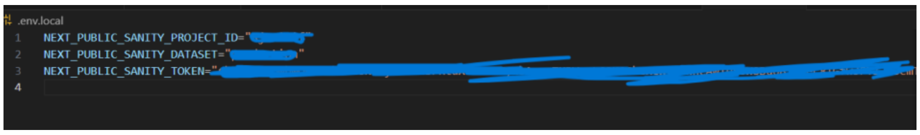
Test the API:

Ensure your API endpoint is working correctly and returning the expected data structure.

B. Setting Up Environment Variables

Configuring Environment Variables

Begin by setting up your environment variables in ***.env.local*** file doesn't already exist in your project's root directory, create one. Then, add the following variables:



```
.env.local
NEXT_PUBLIC_SANITY_PROJECT_ID="[redacted]"
NEXT_PUBLIC_SANITY_DATASET="[redacted]"
NEXT_PUBLIC_SANITY_TOKEN=[redacted]
```

C. Obtaining Sanity Project ID and API Token

Create a Sanity Project:

- Go to Sanity.io and create a new project.

npx sanity@latest init --create-project "YOUR PROJECT NAME" --dataset production

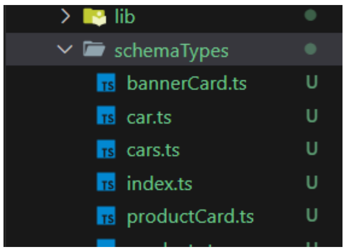
- Note down your Project ID.

Generate an API Token:

- Navigate to Settings > API in your Sanity project dashboard
- Generate a new token with read and write permissions.
- Add this token to your .env.local file as shown above.

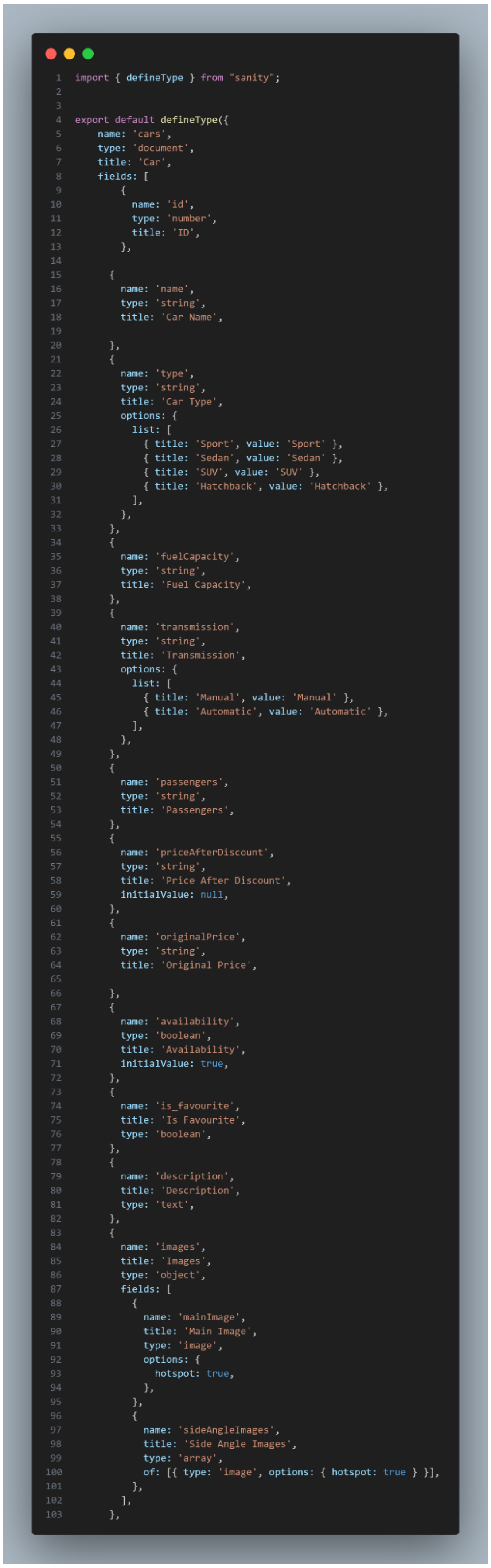
D. Creating the Sanity Schema

1. Create schema file "filename.ts" inside SanityType folder:

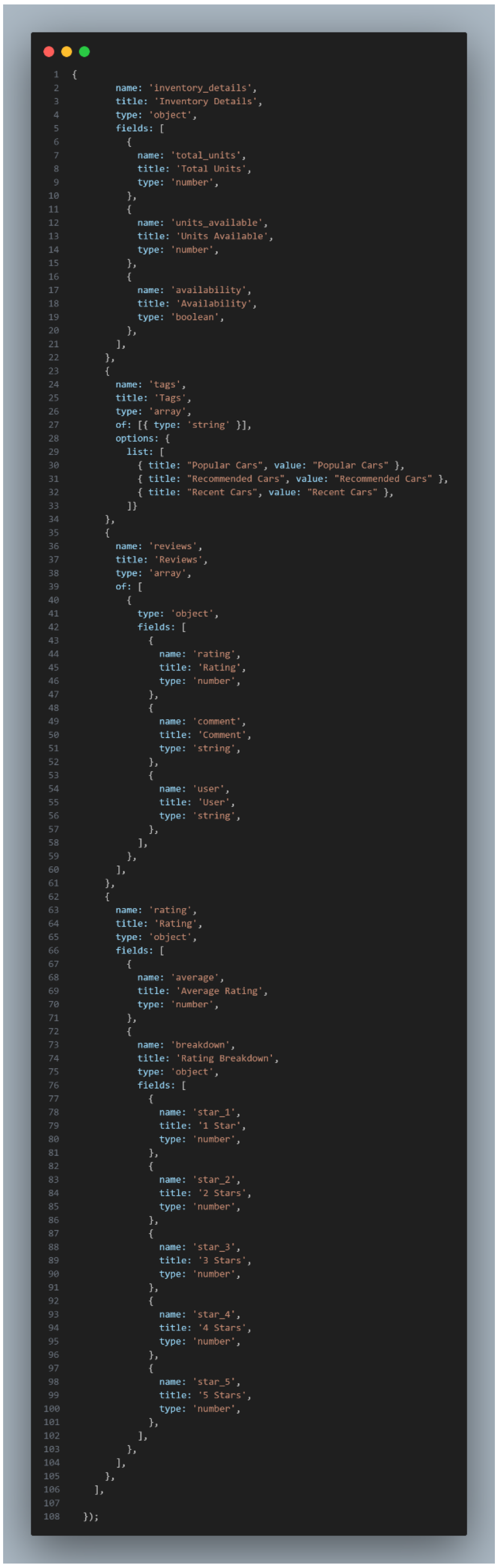


```
> lib
v schemaTypes
    bannerCard.ts     U
    car.ts            U
    cars.ts           U
    index.ts          U
    productCard.ts    U
```

2. In Sanity project's schemas directory, create a file named cars.ts.
3. Schema files are shown below



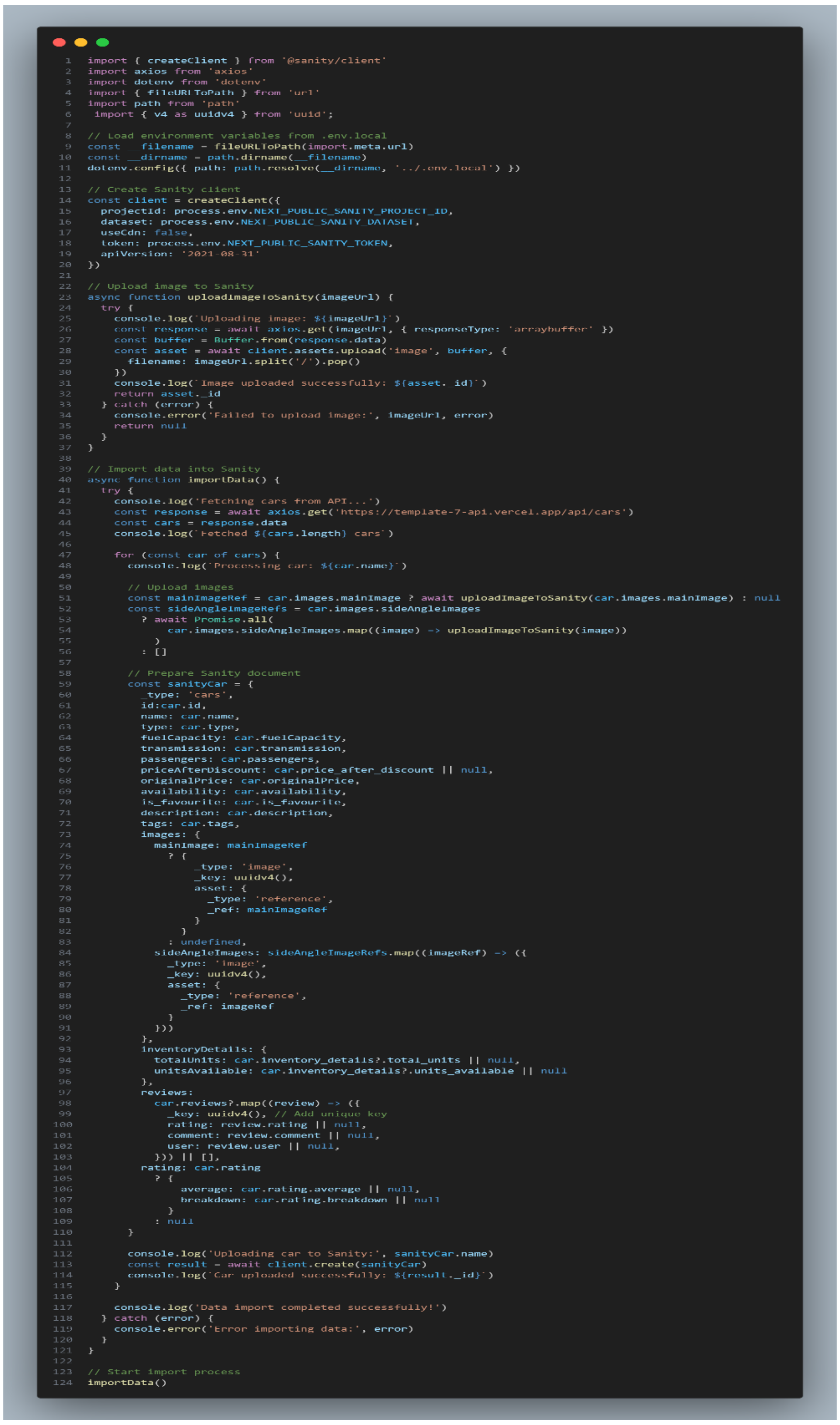
```
import { defineType } from "sanity";


export default defineType({
    name: 'cars',
    type: 'document',
    title: 'Car',
    fields: [
        {
          name: 'id',
          type: 'number',
          title: 'ID',
        },

      {
        name: 'name',
        type: 'string',
        title: 'Car Name',

      },
      {
        name: 'type',
        type: 'string',
        title: 'Car Type',
        options: {
          list: [
            { title: 'Sport', value: 'Sport' },
            { title: 'Sedan', value: 'Sedan' },
            { title: 'SUV', value: 'SUV' },
            { title: 'Hatchback', value: 'Hatchback' },
          ],
        },
      },
      {
        name: 'fuelCapacity',
        type: 'string',
        title: 'Fuel Capacity',
      },
      {
        name: 'transmission',
        type: 'string',
        title: 'Transmission',
        options: {
          list: [
            { title: 'Manual', value: 'Manual' },
            { title: 'Automatic', value: 'Automatic' },
          ],
        },
      },
      {
        name: 'passengers',
        type: 'string',
        title: 'Passengers',
      },
      {
        name: 'priceAfterDiscount',
        type: 'string',
        title: 'Price After Discount',
        initialValue: null,
      },
      {
        name: 'originalPrice',
        type: 'string',
        title: 'Original Price',

      },
      {
        name: 'availability',
        type: 'boolean',
        title: 'Availability',
        initialValue: true,
      },
      {
        name: 'is_favourite',
        title: 'Is Favourite',
        type: 'boolean',
      },
      {
        name: 'description',
        title: 'Description',
        type: 'text',
      },
      {
        name: 'images',
        title: 'Images',
        type: 'object',
        fields: [
          {
            name: 'mainImage',
            title: 'Main Image',
            type: 'image',
            options: {
              hotspot: true,
            },
          },
          {
            name: 'sideAngleImages',
            title: 'Side Angle Images',
            type: 'array',
            of: [{ type: 'image', options: { hotspot: true } }],
          },
        ],
      },
```



```
{
        name: 'inventory_details',
        title: 'Inventory Details',
        type: 'object',
        fields: [
          {
            name: 'total_units',
            title: 'Total Units',
            type: 'number',
          },
          {
            name: 'units_available',
            title: 'Units Available',
            type: 'number',
          },
          {
            name: 'availability',
            title: 'Availability',
            type: 'boolean',
          },
        ],
      },
      {
        name: 'tags',
        title: 'Tags',
        type: 'array',
        of: [{ type: 'string' }],
        options: {
          list: [
            { title: "Popular Cars", value: "Popular Cars" },
            { title: "Recommended Cars", value: "Recommended Cars" },
            { title: "Recent Cars", value: "Recent Cars" },
          ]}
      },
      {
        name: 'reviews',
        title: 'Reviews',
        type: 'array',
        of: [
          {
            type: 'object',
            fields: [
              {
                name: 'rating',
                title: 'Rating',
                type: 'number',
              },
              {
                name: 'comment',
                title: 'Comment',
                type: 'string',
              },
              {
                name: 'user',
                title: 'User',
                type: 'string',
              },
            ],
          },
        ],
      },
      {
        name: 'rating',
        title: 'Rating',
        type: 'object',
        fields: [
          {
            name: 'average',
            title: 'Average Rating',
            type: 'number',
          },
          {
            name: 'breakdown',
            title: 'Rating Breakdown',
            type: 'object',
            fields: [
              {
                name: 'star_1',
                title: '1 Star',
                type: 'number',
              },
              {
                name: 'star_2',
                title: '2 Stars',
                type: 'number',
              },
              {
                name: 'star_3',
                title: '3 Stars',
                type: 'number',
              },
              {
                name: 'star_4',
                title: '4 Stars',
                type: 'number',
              },
              {
                name: 'star_5',
                title: '5 Stars',
                type: 'number',
              },
            ],
          },
        ],
      },
    ],

  });
```

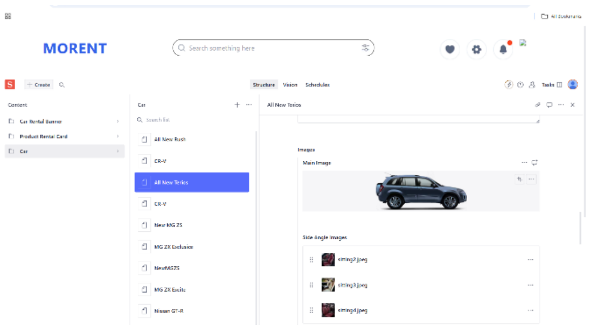
E. Import the Schema into Sanity

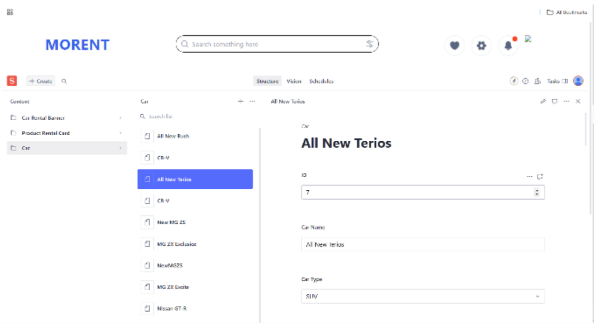
- In the root of your project, create a folder scripts/ migrationData.mjs.
- Import and add the schema:
- Create an Import Script:
- Create a file named migrationData.mjs in your project root and add the following:



```javascript
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';

// 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: '2021-08-31'
})

// Upload image to Sanity
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
  }
}

// Import data into Sanity
async function importData() {
  try {
    console.log('Fetching cars from API...')
    const response = await axios.get('https://template-7-api.vercel.app/api/cars')
    const cars = response.data
    console.log(`Fetched ${cars.length} cars`)

    for (const car of cars) {
      console.log(`Processing car: ${car.name}`)

      // Upload images
      const mainImageRef = car.images.mainImage ? await uploadImageToSanity(car.images.mainImage) : null
      const sideAngleImageRefs = car.images.sideAngleImages
        ? await Promise.all(
            car.images.sideAngleImages.map((image) => uploadImageToSanity(image))
          )
        : []

      // Prepare Sanity document
      const sanityCar = {
        _type: 'cars',
        id:car.id,
        name: car.name,
        type: car.type,
        fuelCapacity: car.fuelCapacity,
        transmission: car.transmission,
        passengers: car.passengers,
        priceAfterDiscount: car.price_after_discount || null,
        originalPrice: car.originalPrice,
        availability: car.availability,
        is_favourite: car.is_favourite,
        description: car.description,
        tags: car.tags,
        images: {
          mainImage: mainImageRef
            ? {
                _type: 'image',
                _key: uuidv4(),
                asset: {
                  _type: 'reference',
                  _ref: mainImageRef
                }
              }
            : undefined,
          sideAngleImages: sideAngleImageRefs.map((imageRef) => ({
            _type: 'image',
            _key: uuidv4(),
            asset: {
              _type: 'reference',
              _ref: imageRef
            }
          }))
        },
        inventoryDetails: {
          totalUnits: car.inventory_details?.total_units || null,
          unitsAvailable: car.inventory_details?.units_available || null
        },
        reviews:
          car.reviews?.map((review) => ({
            _key: uuidv4(), // Add unique key
            rating: review.rating || null,
            comment: review.comment || null,
            user: review.user || null,
          })) || [],
        rating: car.rating
          ? {
              average: car.rating.average || null,
              breakdown: car.rating.breakdown || null
            }
          : null
      }

      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)
  }
}

// Start import process
importData()
```

F. Data Imported in Sanity





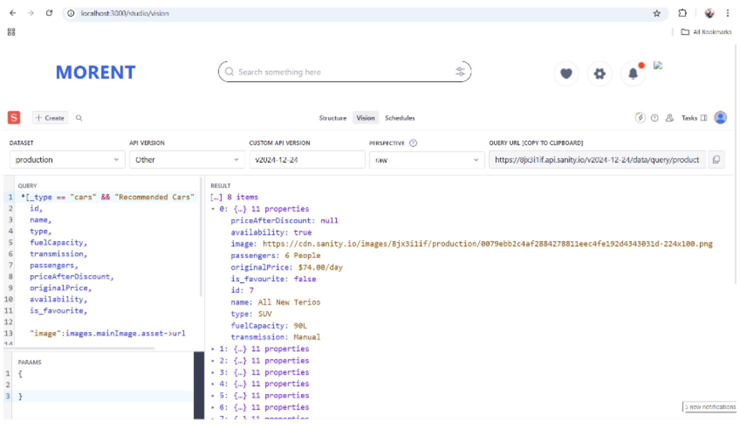
All New Terios

ID: 7

Car Name: All New Terios

Car Type: SUV

G. GROQ in Sanity



DATASET: production
API VERSION: Other
CUSTOM API VERSION: v2024-12-24
PERSPECTIVE: raw
QUERY URL [COPY TO CLIPBOARD]: https://8jx3i1if.api.sanity.io/v2024-12-24/data/query/product

QUERY
```
*[_type == "cars" && "Recommended Cars"
  id,
  name,
  type,
  fuelCapacity,
  transmission,
  passengers,
  priceAfterDiscount,
  originalPrice,
  availability,
  is_favourite,

  "image":images.mainImage.asset->url
```

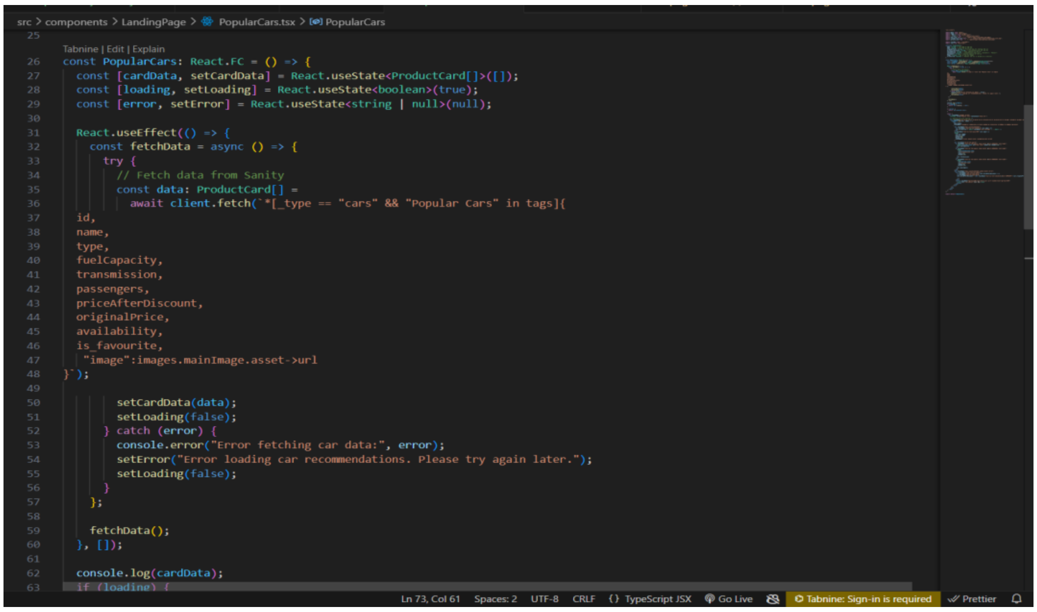
PARAMS
```
{

}
```

RESULT
```
[…] 8 items
▾ 0: {…} 11 properties
    priceAfterDiscount: null
    availability: true
    image: https://cdn.sanity.io/images/8jx3i1if/production/0079ebb2c4af2884278811eec4fe192d4343031d-224x100.png
    passengers: 6 People
    originalPrice: $74.00/day
    is_favourite: false
    id: 7
    name: All New Terios
    type: SUV
    fuelCapacity: 90L
    transmission: Manual
▸ 1: {…} 11 properties
▸ 2: {…} 11 properties
▸ 3: {…} 11 properties
▸ 4: {…} 11 properties
▸ 5: {…} 11 properties
▸ 6: {…} 11 properties
```

H. Fetching Data Locally with GROQ

- **Run the Query in Your Project:**

  Integrate this query into your local project (e.g., within a React or Next.js app) to display the fetched car data.


src > components > LandingPage > PopularCars.tsx > PopularCars
Tabnine | Edit | Explain
Ln 73, Col 61
Spaces: 2
UTF-8
CRLF
{} TypeScript JSX
Go Live
Tabnine: Sign-in is required
Prettier

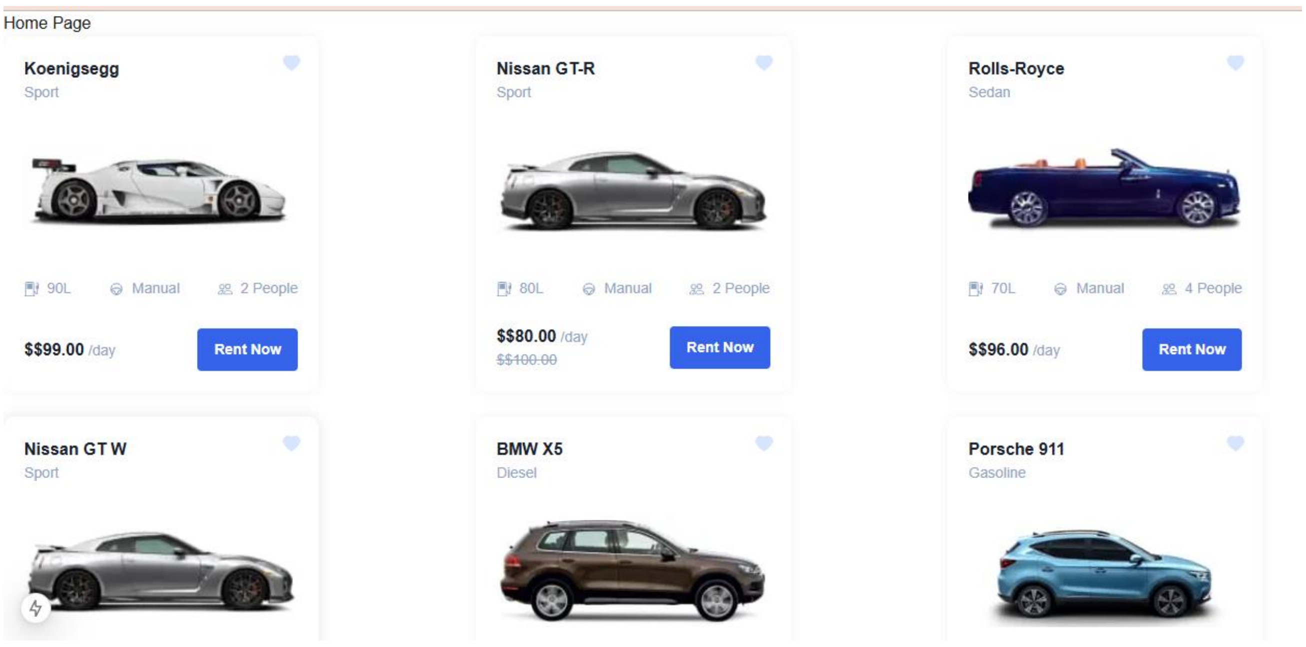
```tsx

const PopularCars: React.FC = () => {
  const [cardData, setCardData] = React.useState<ProductCard[]>([]);
  const [loading, setLoading] = React.useState<boolean>(true);
  const [error, setError] = React.useState<string | null>(null);

  React.useEffect(() => {
    const fetchData = async () => {
      try {
        // Fetch data from Sanity
        const data: ProductCard[] =
          await client.fetch(`*[_type == "cars" && "Popular Cars" in tags]{
  id,
  name,
  type,
  fuelCapacity,
  transmission,
  passengers,
  priceAfterDiscount,
  originalPrice,
  availability,
  is_favourite,
   "image":images.mainImage.asset->url
}`);

        setCardData(data);
        setLoading(false);
      } catch (error) {
        console.error("Error fetching car data:", error);
        setError("Error loading car recommendations. Please try again later.");
        setLoading(false);
      }
    };

    fetchData();
  }, []);

  console.log(cardData);
  if (loading) {
```

- **Test on Localhost:**

  Start your development server and verify that the data is being fetched and displayed correctly.

Conclusion

- By following these steps, you have successfully:
- Created an API for your rental car data.
- Set up a clean Sanity CMS project.
- Defined and imported a custom schema for cars.
- Imported API data into Sanity.
- Fetched and displayed the data locally using GROQ queries.
- This workflow ensures a seamless connection between your mock API and Sanity, allowing you to manage and display your data effectively.



Home Page

Koenigsegg
Sport
90L Manual 2 People
$$99.00 /day
Rent Now

Nissan GT-R
Sport
80L Manual 2 People
$$80.00 /day
~~$$100.00~~
Rent Now

Rolls-Royce
Sedan
70L Manual 4 People
$$96.00 /day
Rent Now

Nissan GT W
Sport

BMW X5
Diesel

Porsche 911
Gasoline