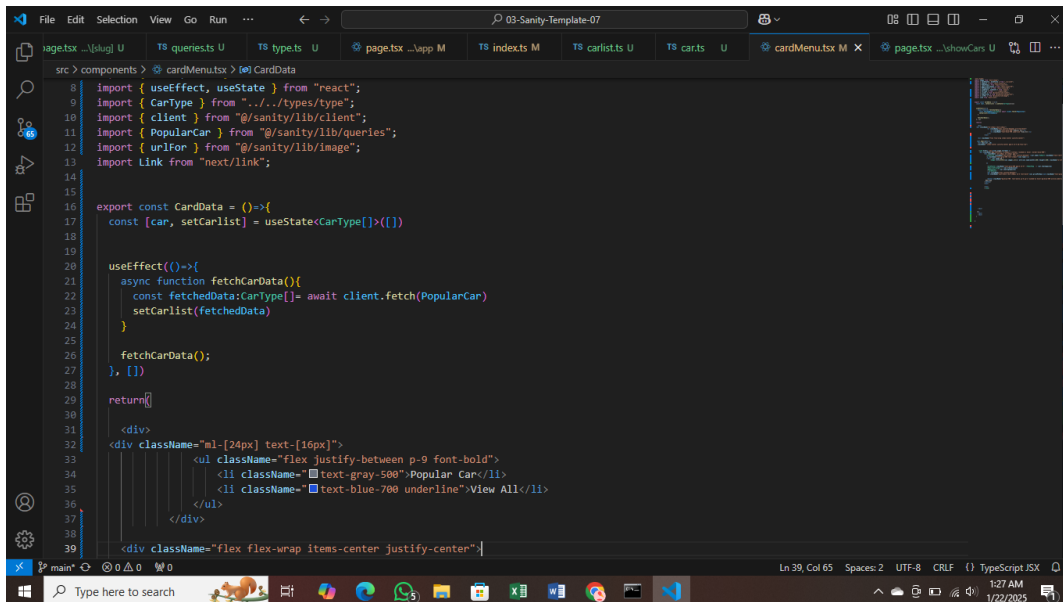


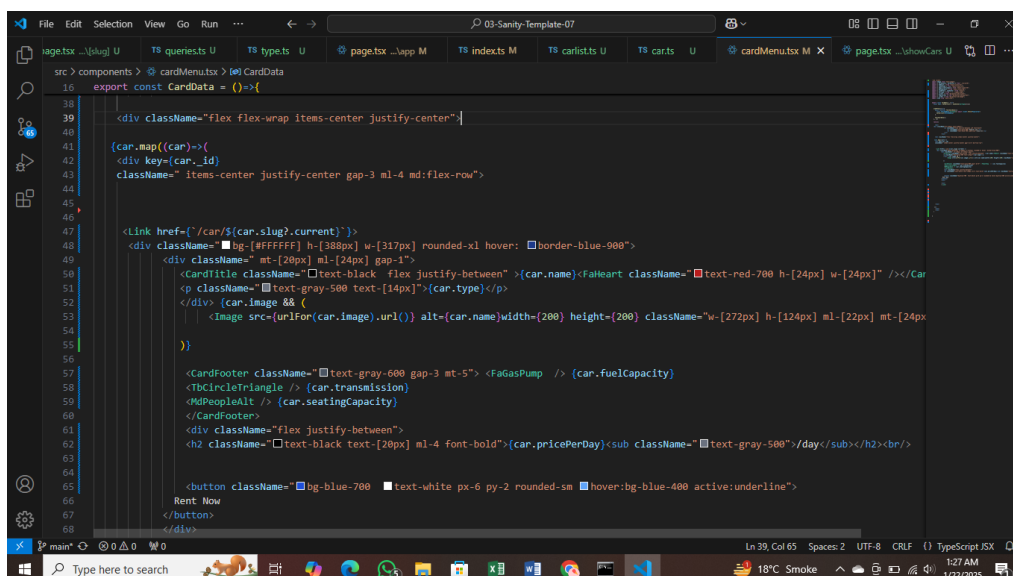
Hackathon # 03

Day: 04

API Integration and Migration In Frontend Data Fetching

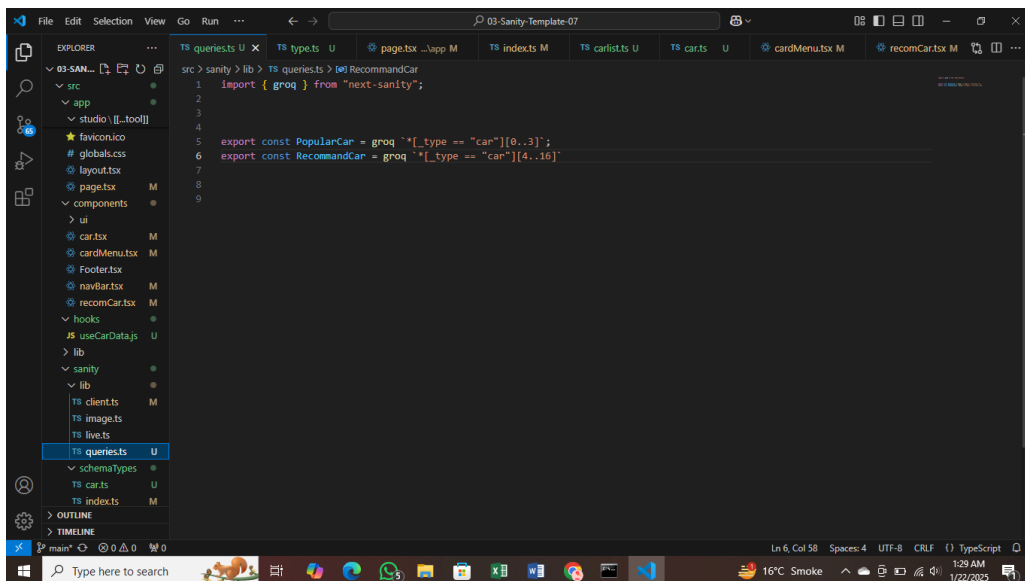


```
8 import { useEffect, useState } from "react";
9 import { CarType } from "../../types/type";
10 import { client } from "@sanity/lib/client";
11 import { PopularCar } from "@sanity/lib/queries";
12 import { urlFor } from "@sanity/lib/image";
13 import Link from "next/link";
14
15
16 export const CardData = () => {
17   const [car, setCarList] = useState<CarType[]>([])
18
19
20   useEffect(() => {
21     async function fetchCarData() {
22       const fetchedData: CarType[] = await client.fetch(PopularCar)
23       setCarList(fetchedData)
24     }
25
26     fetchCarData();
27   }, [])
28
29   return (
30     <div>
31       <div className="ml-[24px] text-[16px]">
32         <div className="flex justify-between p-9 font-bold">
33           <li className="text-gray-500">Popular Car</li>
34           <li className="text-blue-700 underline">View All</li>
35         </div>
36       </div>
37     </div>
38   )
39 }
```



```
16 export const CardData = () => {
38
39   <div className="flex flex-wrap items-center justify-center">
40
41     {car.map((car) => {
42       <div key={car._id}
43         className="items-center justify-center gap-3 ml-4 md:flex-row">
44
45         <Link href={`/car/${car.slug?.current}`}>
46           <div className="bg-[#ffff] h-[180px] w-[317px] rounded-xl hover: border-blue-900">
47             <div className="mt-[20px] ml-[24px] gap-1">
48               <CardTitle className="text-black flex justify-between">{car.name}<FaHeart className="text-red-700 h-[24px] w-[24px]" /></CardTitle>
49               <p className="text-gray-500 text-[14px]">{car.type}</p>
50               <div>
51                 <Image src={urlFor(car.image.url())} alt={car.name} width={200} height={200} className="w-[272px] h-[124px] ml-[22px] mt-[24px]" />
52               </div>
53             <CardFooter className="text-gray-600 gap-3 mt-5">
54               <FaGasPump /> {car.fuelCapacity}
55               <TbCircleTriangle /> {car.transmission}
56               <MdPeopleAlt /> {car.seatingCapacity}
57             </CardFooter>
58             <div className="flex justify-between">
59               <h2 className="text-black text-[20px] ml-4 font-bold">{car.pricePerDay}<sub className="text-gray-500">/day</sub></h2><br/>
60               <button className="bg-blue-700 text-white px-6 py-2 rounded-sm hover:bg-blue-400 active:underline">
61                 Rent Now
62               </button>
63             </div>
64           </div>
65         </div>
66       </div>
67     })}
68   </div>
69 }
```

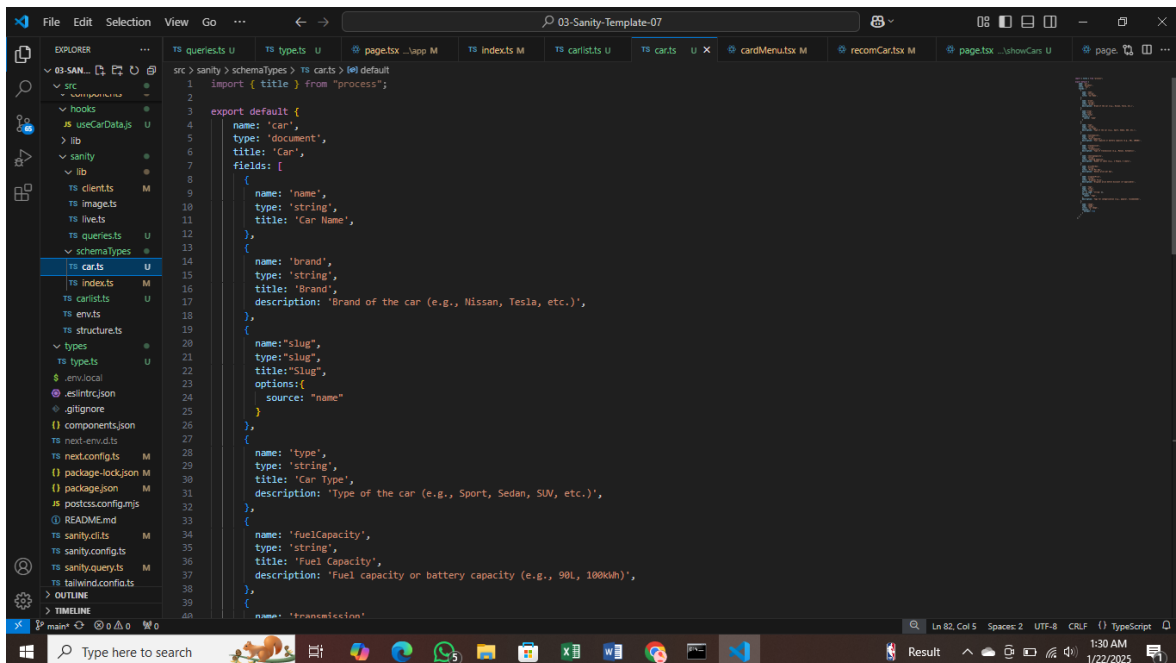
Queries



The screenshot shows a VS Code editor window with the file explorer on the left. The active file is `src/sanity/lib/queries.ts`. The code defines two GraphQL queries: `PopularCar` and `RecommandCar`. The `RecommandCar` query is selected in the editor.

```
1 import { groq } from "next-sanity";
2
3
4
5 export const PopularCar = groq `*[_type == "car"][0..3]`;
6 export const RecommandCar = groq `*[_type == "car"][4..16]`;
```

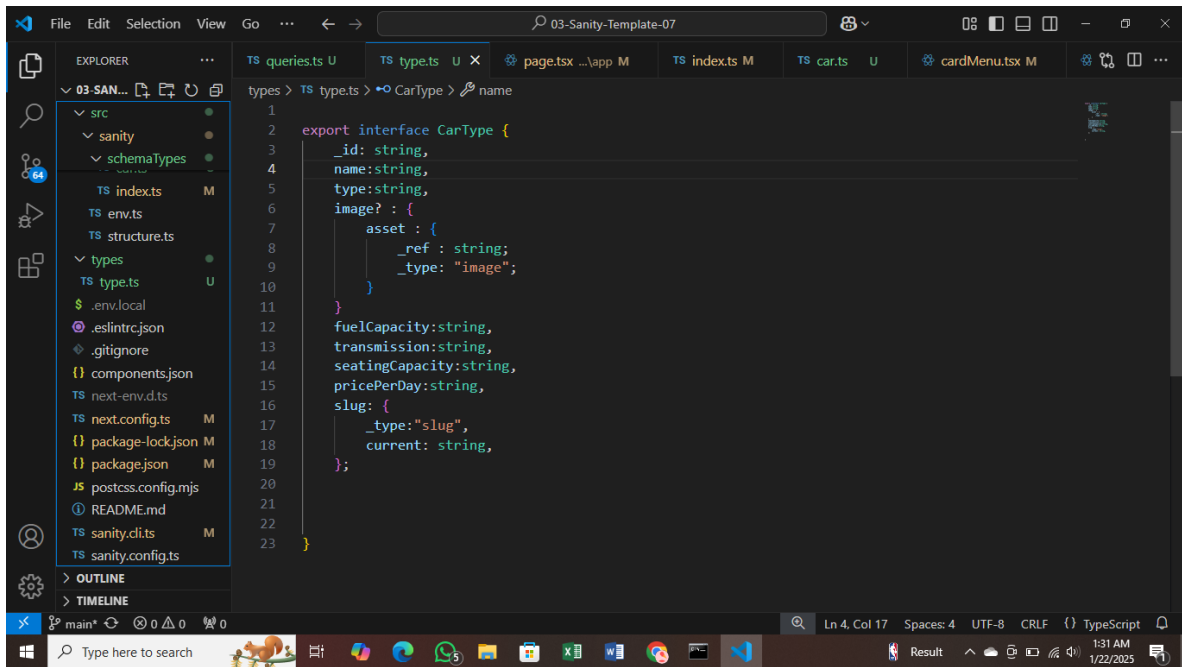
Schema



The screenshot shows a VS Code editor window with the file explorer on the left. The active file is `src/sanity/schemas/types.ts`. The code defines a GraphQL schema for a car. The `car` type is selected in the editor.

```
1 import { title } from "process";
2
3 export default {
4   name: 'car',
5   type: 'document',
6   title: 'Car',
7   fields: [
8     {
9       name: 'name',
10      type: 'string',
11      title: 'Car Name',
12    },
13     {
14       name: 'brand',
15       type: 'string',
16       title: 'Brand',
17       description: 'Brand of the car (e.g., Nissan, Tesla, etc.)',
18     },
19     {
20       name: 'slug',
21       type: 'slug',
22       title: 'Slug',
23       options: {
24         source: "name"
25       },
26     },
27     {
28       name: 'type',
29       type: 'string',
30       title: 'Car Type',
31       description: 'Type of the car (e.g., Sport, Sedan, SUV, etc.)',
32     },
33     {
34       name: 'fuelCapacity',
35       type: 'string',
36       title: 'Fuel Capacity',
37       description: 'Fuel capacity or battery capacity (e.g., 90L, 100kWh)',
38     },
39   ],
40   // ...transmission
41 }
```

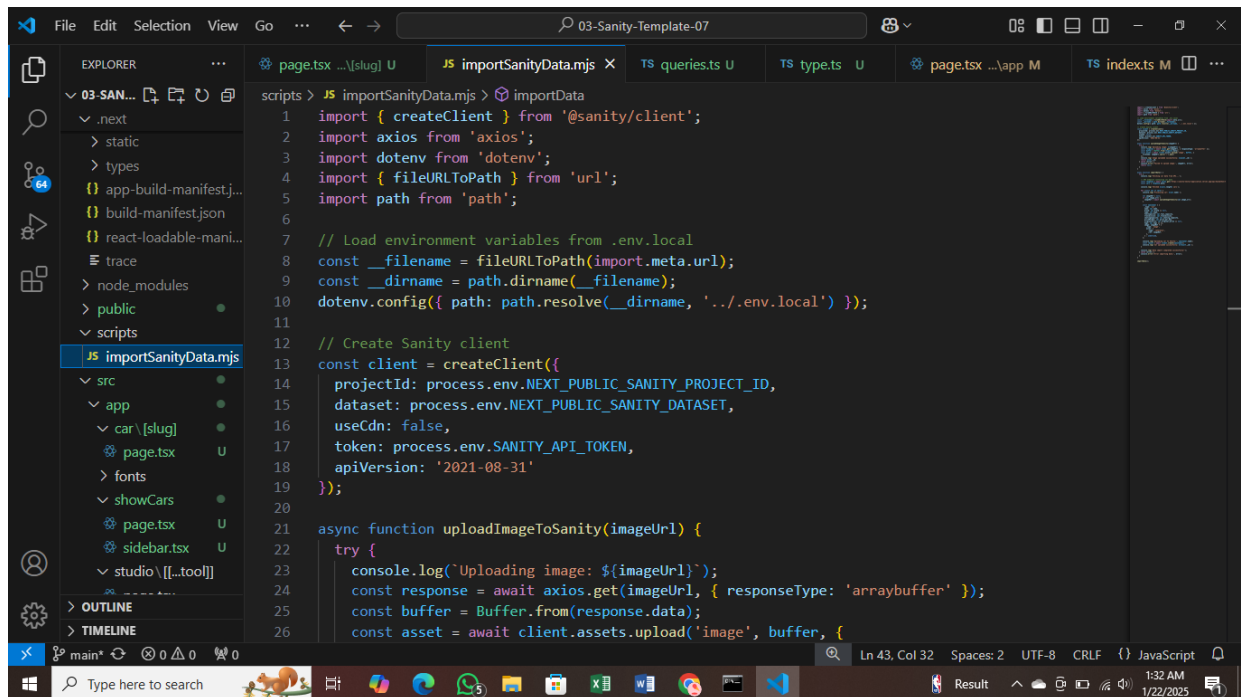
Interface



This screenshot shows the VS Code editor with a TypeScript file named `types.ts` open. The Explorer sidebar on the left shows the project structure, including a `types` folder. The main editor area displays the following TypeScript code:

```
1 export interface CarType {
2   _id: string,
3   name: string,
4   type: string,
5   image?: {
6     asset: {
7       _ref: string;
8       _type: "image";
9     }
10  }
11 }
12 fuelCapacity: string,
13 transmission: string,
14 seatingCapacity: string,
15 pricePerDay: string,
16 slug: {
17   _type: "slug",
18   current: string,
19 };
20 }
21
22
23 }
```

The status bar at the bottom indicates the file is at Line 4, Column 17, using UTF-8 encoding and CRLF line endings. The language mode is set to TypeScript.



This screenshot shows the VS Code editor with a JavaScript file named `importSanityData.mjs` open. The Explorer sidebar on the left shows the project structure, including a `scripts` folder. The main editor area displays the following JavaScript code:

```
1 import { createClient } from '@sanity/client';
2 import axios from 'axios';
3 import dotenv from 'dotenv';
4 import { fileURLToPath } from 'url';
5 import path from 'path';
6
7 // Load environment variables from .env.local
8 const __filename = fileURLToPath(import.meta.url);
9 const __dirname = path.dirname(__filename);
10 dotenv.config({ path: path.resolve(__dirname, '../.env.local') });
11
12 // Create Sanity client
13 const client = createClient({
14   projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
15   dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
16   useCdn: false,
17   token: process.env.SANITY_API_TOKEN,
18   apiVersion: '2021-08-31'
19 });
20
21 async function uploadImageToSanity(imageUrl) {
22   try {
23     console.log(`Uploading image: ${imageUrl}`);
24     const response = await axios.get(imageUrl, { responseType: 'arraybuffer' });
25     const buffer = Buffer.from(response.data);
26     const asset = await client.assets.upload('image', buffer, {
```

The status bar at the bottom indicates the file is at Line 43, Column 32, using UTF-8 encoding and CRLF line endings. The language mode is set to JavaScript.

Frontend

MORENT

Q Search something here



The Best Platform for Car Rental

Ease of doing a car rental safely and
reliably. Of course at a low price.

Rental Car



Easy way to rent a car at a low price

Providing cheap car rental services and
safe and comfortable facilities.

Rental Car



Popular Car

[View All](#)

Nissan GT-R
Sport



Nissan GT-R
Sport



Rolls-Royce
SUV



CR-V
SUV



80L Manual 2 People

80L Manual 2 People

70L Manual 6 People

80L Manual 6 People

\$80.00/day

Rent Now

\$80.00/day

Rent Now

\$72.00/day

Rent Now

\$80.00/day

Rent Now

Recommand Car

MG ZX Exclusive
Hatchback



Rolls-Royce
Sedan



Tesla Model 3
Electric



Ford Mustang
Gasoline



BMW X5
Diesel



4



 50L  Manual  5 seats

[Rent Now](#)

Discord
Instagram
Twitter
Facebook

Electric



100kWh Manual 5 seats

[Rent Now](#)

Hybrid

