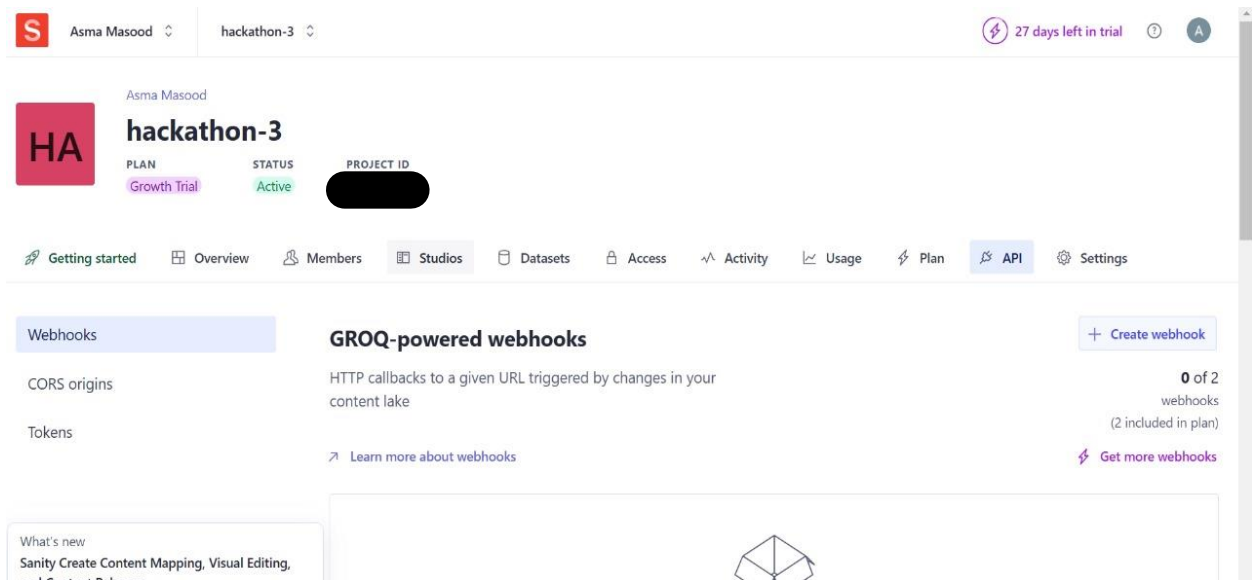


Day 3

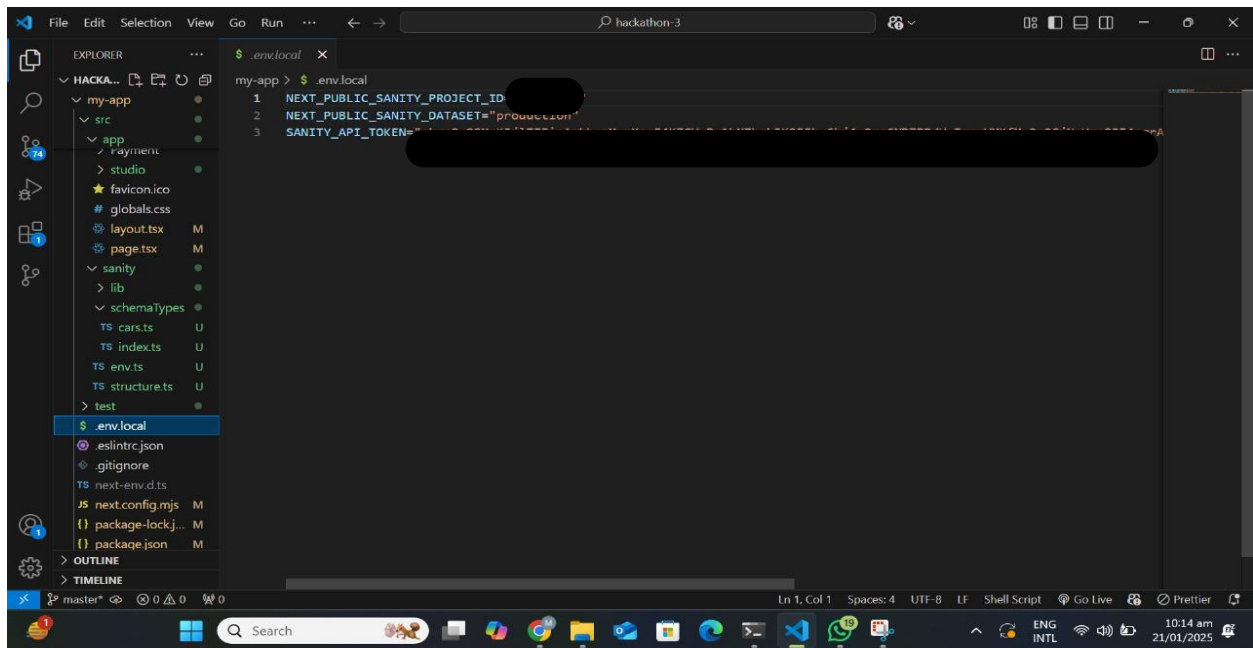
By following these steps, I can successfully integrate Sanity with an existing Next.js project, configure environment variables, create schemas, and populate data from external APIs. This setup provides a powerful and flexible foundation for managing your content efficiently.

1. Setting up Environment variables: -

Take project id on sanity project and click on API then take token.



Then make .env.local file on your project and put in the values shown in screen shot.



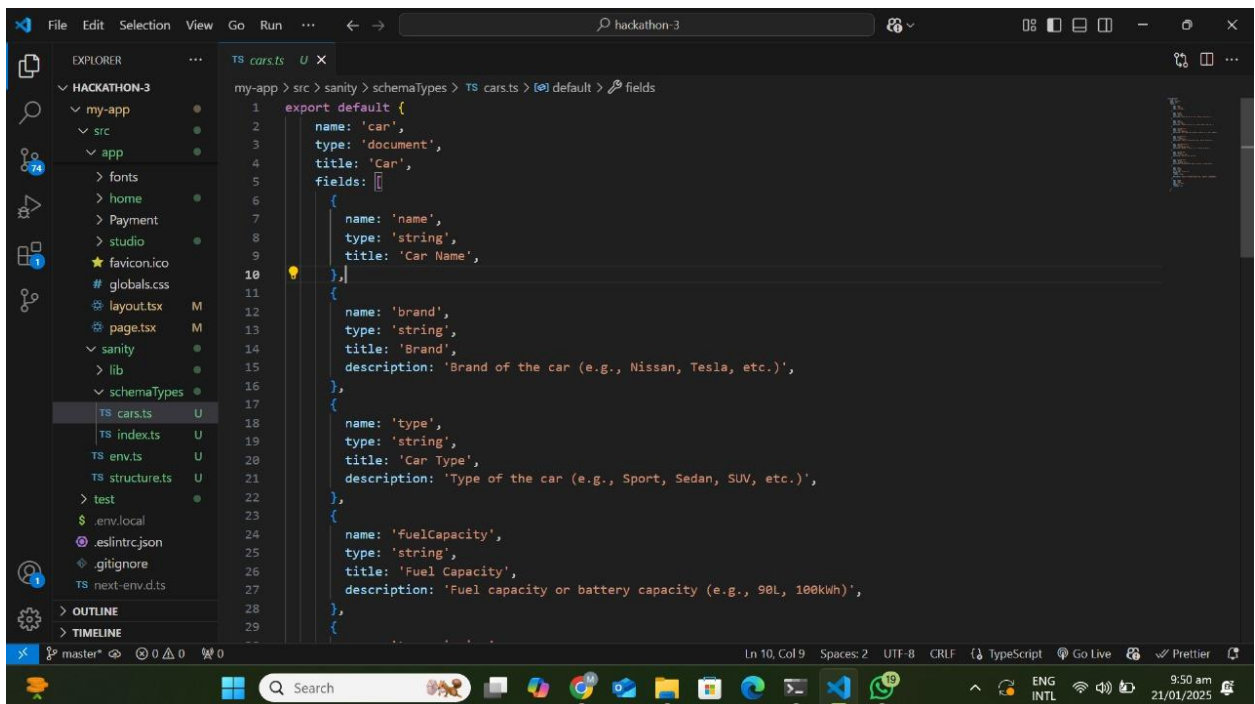
The screenshot shows a VS Code editor window with the file explorer on the left and the editor on the right. The file explorer shows a project structure with a 'my-app' folder containing 'src', 'app', 'payment', 'studio', 'sanity', 'lib', 'schemasTypes', 'TS cars.ts', 'TS index.ts', 'TS env.ts', 'TS structure.ts', 'test', and '.env.local'. The editor shows the content of the '.env.local' file:

```
my-app > $ .env.local
1 NEXT_PUBLIC_SANITY_PROJECT_ID=
2 NEXT_PUBLIC_SANITY_DATASET='production'
3 SANITY_API_TOKEN=
```

2. Creating schemas:-

Schemas define the structure of your content in Sanity.

I create schema in sanity/schemaType/cars.ts, shown in screen shot.

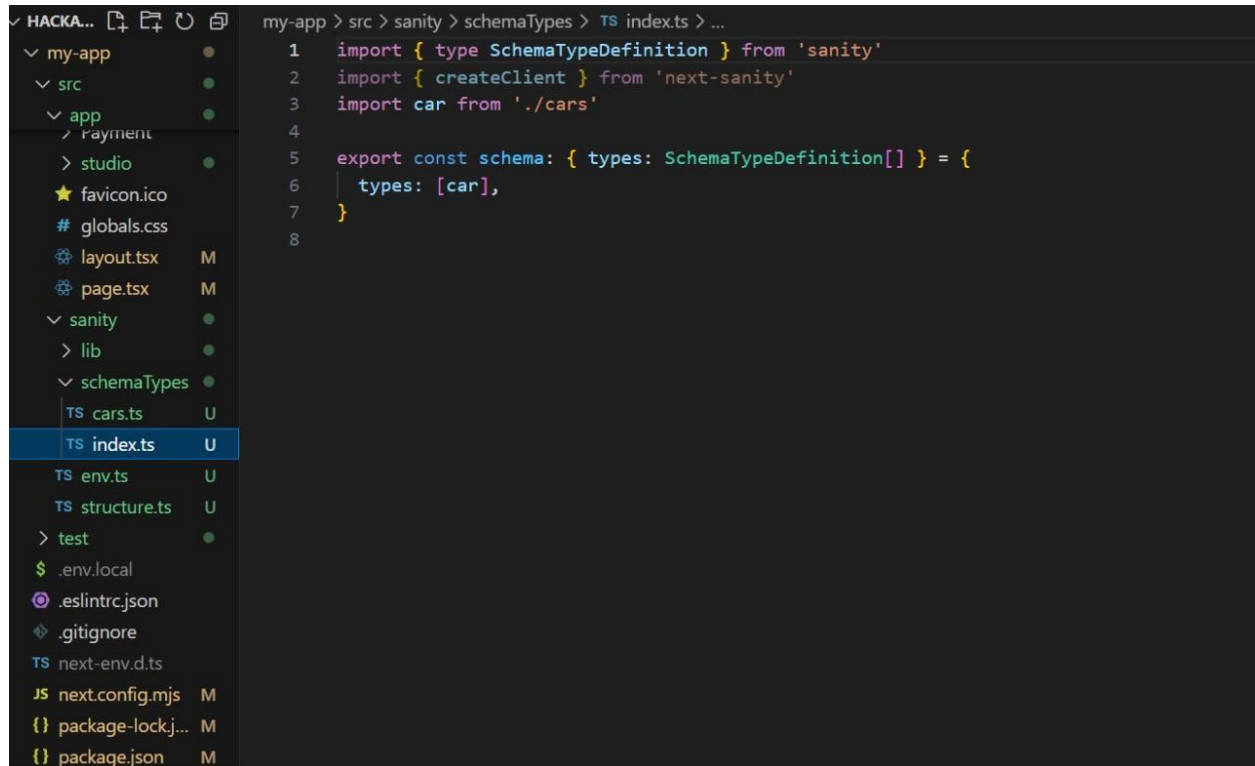


The screenshot shows a VS Code editor window with the file explorer on the left and the editor on the right. The file explorer shows a project structure with a 'HACKATHON-3' folder containing 'my-app', 'src', 'app', 'fonts', 'home', 'Payment', 'studio', 'favicon.ico', 'globals.css', 'layout.tsx', 'page.tsx', 'sanity', 'lib', 'schemasTypes', 'TS cars.ts', 'TS index.ts', 'TS env.ts', 'TS structure.ts', 'test', '.env.local', 'eslint.config', 'gitignore', 'next-env.d.ts', 'next.config.mjs', 'package-lock.json', and 'package.json'. The editor shows the content of the 'TS cars.ts' file:

```
my-app > src > sanity > schemaTypes > TS cars.ts > default > fields
1 export default {
2   name: 'car',
3   type: 'document',
4   title: 'Car',
5   fields: [
6     {
7       name: 'name',
8       type: 'string',
9       title: 'Car Name',
10    },
11    {
12      name: 'brand',
13      type: 'string',
14      title: 'Brand',
15      description: 'Brand of the car (e.g., Nissan, Tesla, etc.)',
16    },
17    {
18      name: 'type',
19      type: 'string',
20      title: 'Car Type',
21      description: 'Type of the car (e.g., Sport, Sedan, SUV, etc.)',
22    },
23    {
24      name: 'fuelCapacity',
25      type: 'string',
26      title: 'Fuel Capacity',
27      description: 'Fuel capacity or battery capacity (e.g., 90L, 100kWh)',
28    },
29  ],
30 }
```

3. Update index.ts file: -

I added sanity/schemaTypes/index.ts file include new product schema in types:[] and import the file shown in screen shot.

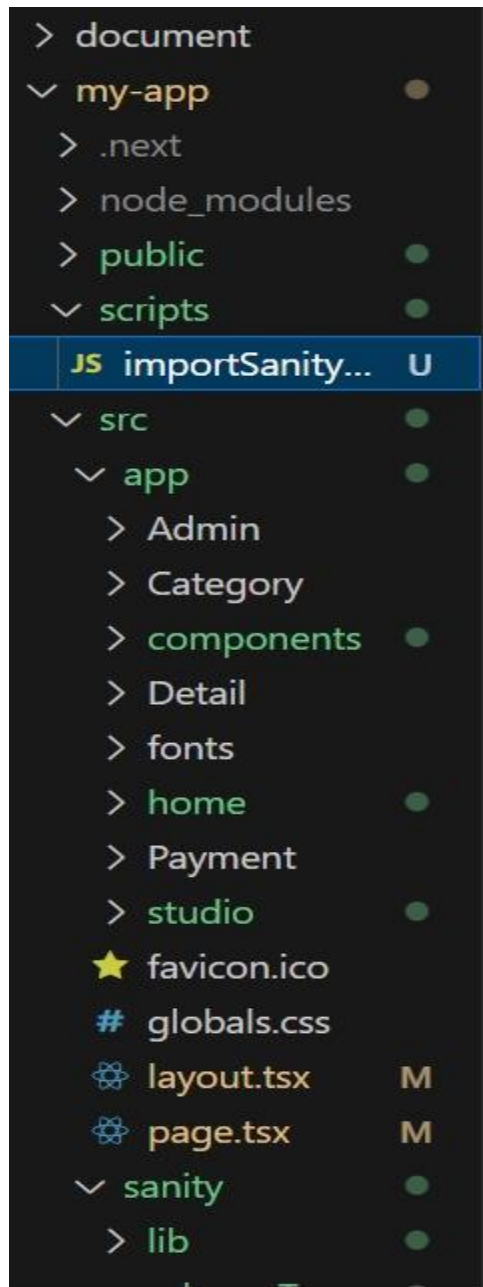


The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders like 'my-app', 'src', 'app', 'sanity', and 'schemaTypes'. The 'index.ts' file in the 'schemaTypes' folder is selected. The code editor shows the following TypeScript code:

```
my-app > src > sanity > schemaTypes > TS index.ts > ...  
1  import { type SchemaTypeDefinition } from 'sanity'  
2  import { createClient } from 'next-sanity'  
3  import car from './cars'  
4  
5  export const schema: { types: SchemaTypeDefinition[] } = {  
6    types: [car],  
7  }  
8
```

4. Setting for Data Import: -

I created folder, scripts folder and created new file "importSanityData.mjs" in this folder to import data from an external Api in to Sanity shows in screen shot.



```
JS importSanityData.mjs U X
my-app > scripts > JS importSanityData.mjs > ...
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, {
27        filename: imageUrl.split('/').pop()
28      });
29      console.log(`Image uploaded successfully: ${asset._id}`);
30    } catch (error) {
31      console.error('Error uploading image:', error);
32    }
33  }
```

5. Run the command: -

Install the necessary packages. Run the following command in your terminal:

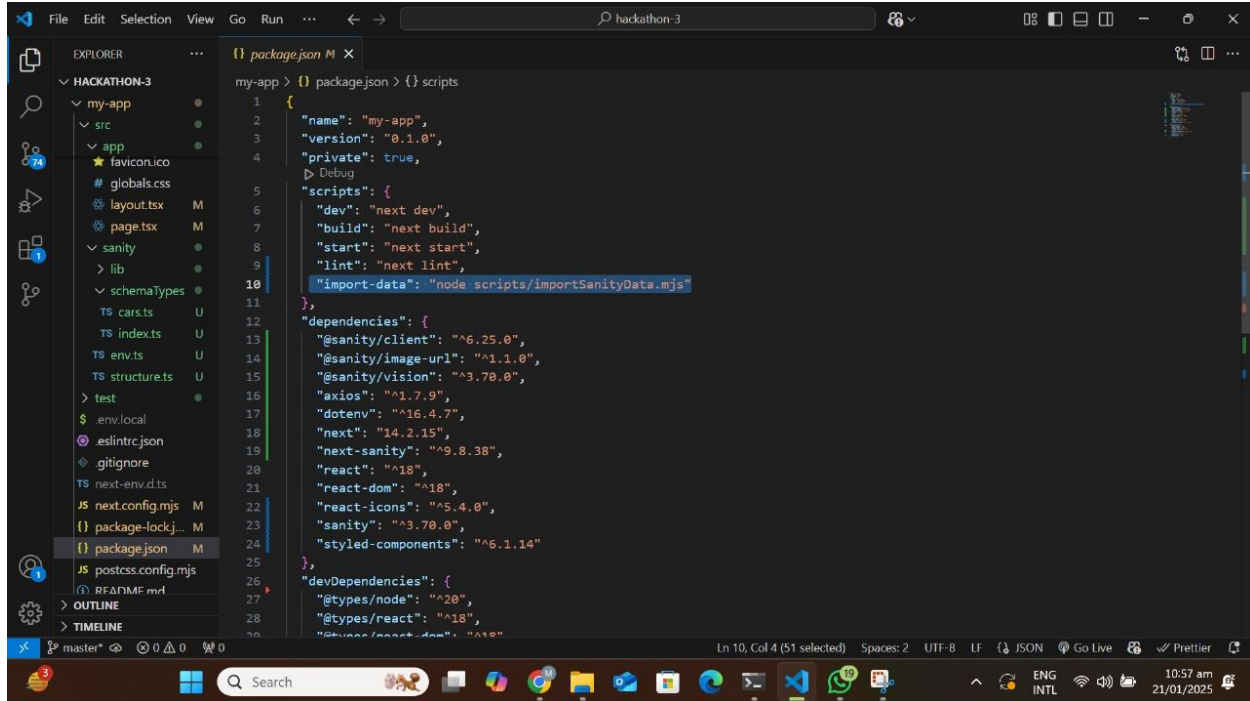
npm install @sanity/client axios dotenv.

6. Changes in packagejson file:-

To run the import script, we need to add scripts to our package.json file. Open your package.json and add the following to the "scripts" section:

"import-data": "node scripts/importSanityData.mjs"

Show in screen shot.



The screenshot shows a Visual Studio Code editor window with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with a 'my-app' directory containing 'src', 'app', 'lib', 'schemaTypes', 'test', and 'env.local'. The code editor displays the 'package.json' file with the following content:

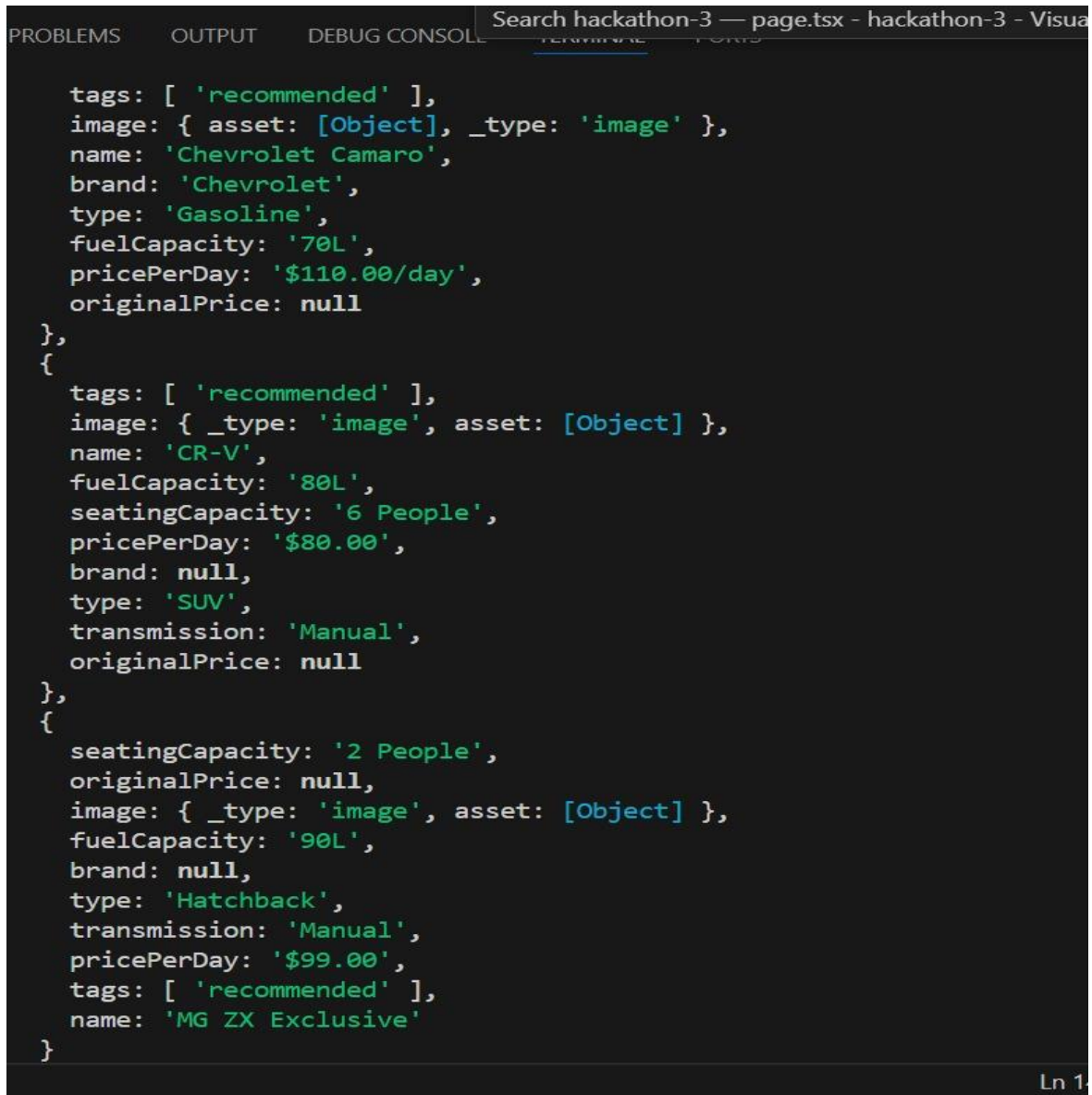
```
{
  "name": "my-app",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint",
    "import-data": "node scripts/importSanityData.mjs"
  },
  "dependencies": {
    "@sanity/client": "^6.25.0",
    "@sanity/image-url": "^1.1.0",
    "@sanity/vision": "^3.70.0",
    "axios": "^1.7.9",
    "dotenv": "^16.4.7",
    "next": "14.2.15",
    "next-sanity": "^9.8.38",
    "react": "^18",
    "react-dom": "^18",
    "react-icons": "^5.4.0",
    "sanity": "^3.70.0",
    "styled-components": "^6.1.14"
  },
  "devDependencies": {
    "@types/node": "^20",
    "@types/react": "^18",
    "@types/react-dom": "^18"
  }
}
```

The 'import-data' script is highlighted in blue. The status bar at the bottom indicates 'Ln 10, Col 4 (51 selected)' and 'Spaces: 2 UTF-8 LF JSON Go Live Prettier'.

7. Run command: -

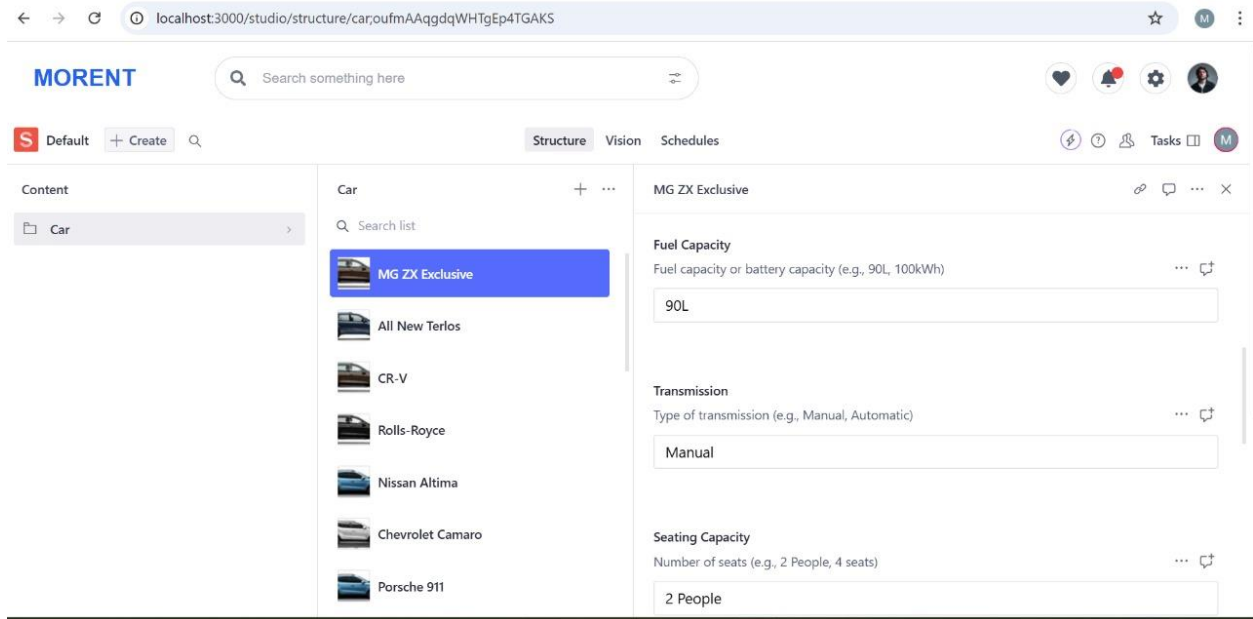
Now you can run the import script using:

```
npm run import-data
```



```
Search hackathon-3 — page.tsx - hackathon-3 - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE
tags: [ 'recommended' ],
image: { asset: [Object], _type: 'image' },
name: 'Chevrolet Camaro',
brand: 'Chevrolet',
type: 'Gasoline',
fuelCapacity: '70L',
pricePerDay: '$110.00/day',
originalPrice: null
},
{
  tags: [ 'recommended' ],
  image: { _type: 'image', asset: [Object] },
  name: 'CR-V',
  fuelCapacity: '80L',
  seatingCapacity: '6 People',
  pricePerDay: '$80.00',
  brand: null,
  type: 'SUV',
  transmission: 'Manual',
  originalPrice: null
},
{
  seatingCapacity: '2 People',
  originalPrice: null,
  image: { _type: 'image', asset: [Object] },
  fuelCapacity: '90L',
  brand: null,
  type: 'Hatchback',
  transmission: 'Manual',
  pricePerDay: '$99.00',
  tags: [ 'recommended' ],
  name: 'MG ZX Exclusive'
}
Ln 1
```

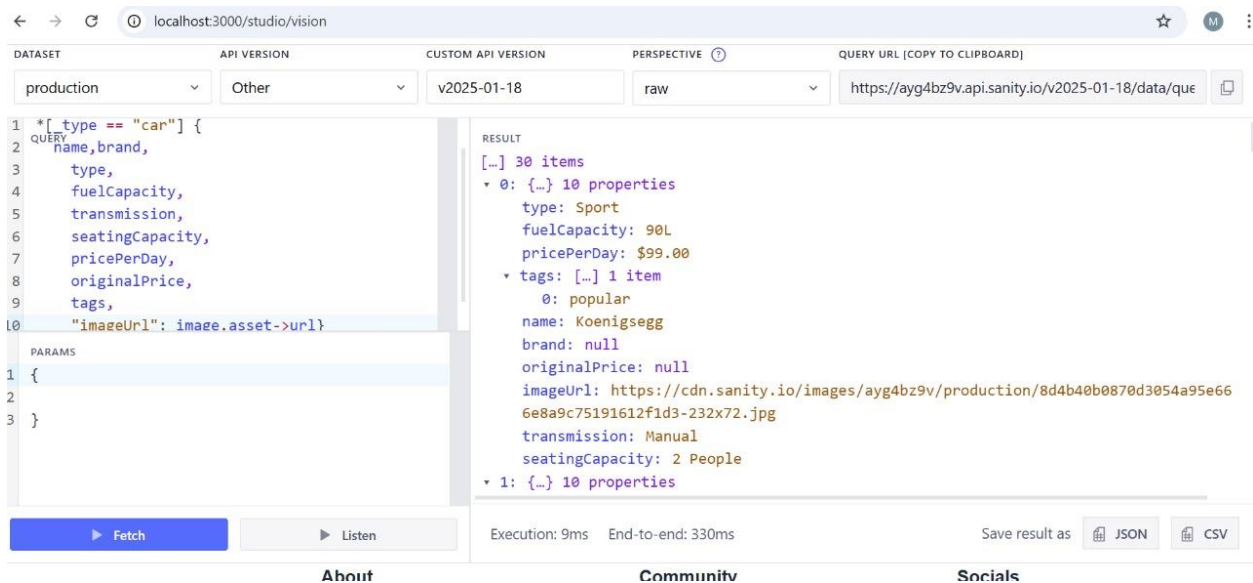
This script will fetch cars from the sir Ashribs API, in to Sanity store, and then create new cars documents in your Sanity dataset.



8. Fetch data: -

I click into vision, vision page is display then write groq query to fetch data in project.

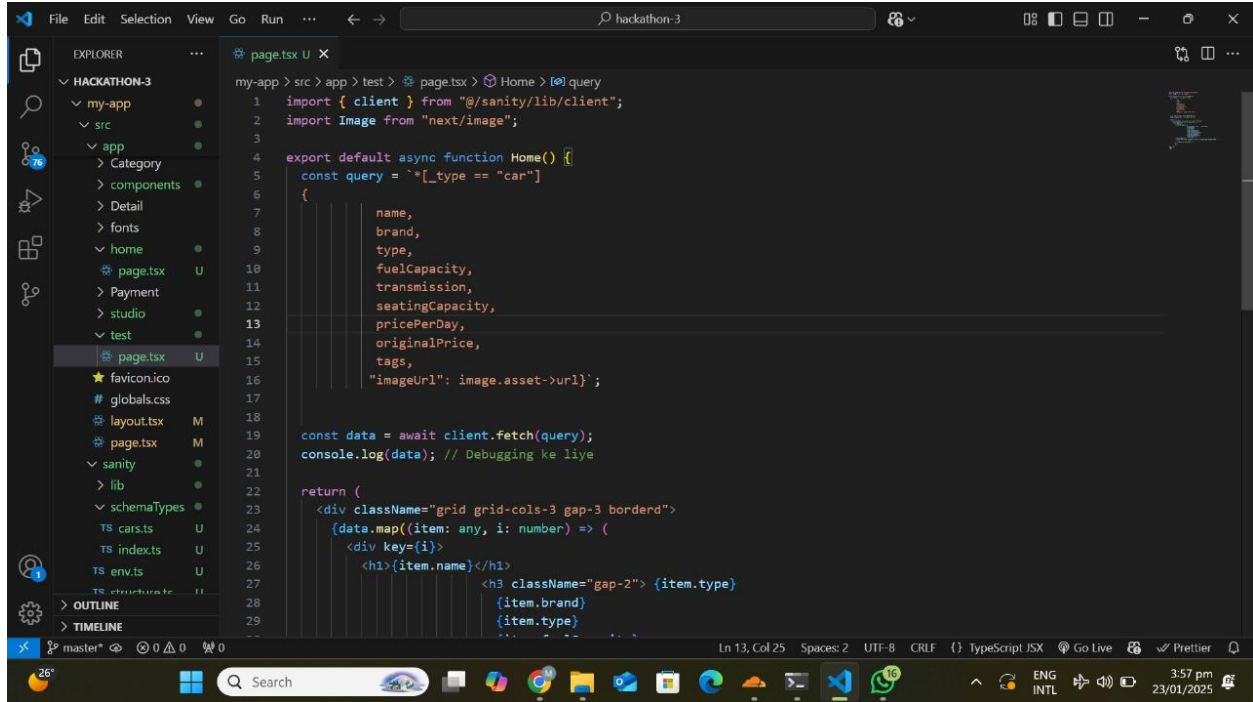
Shown in screen shot.



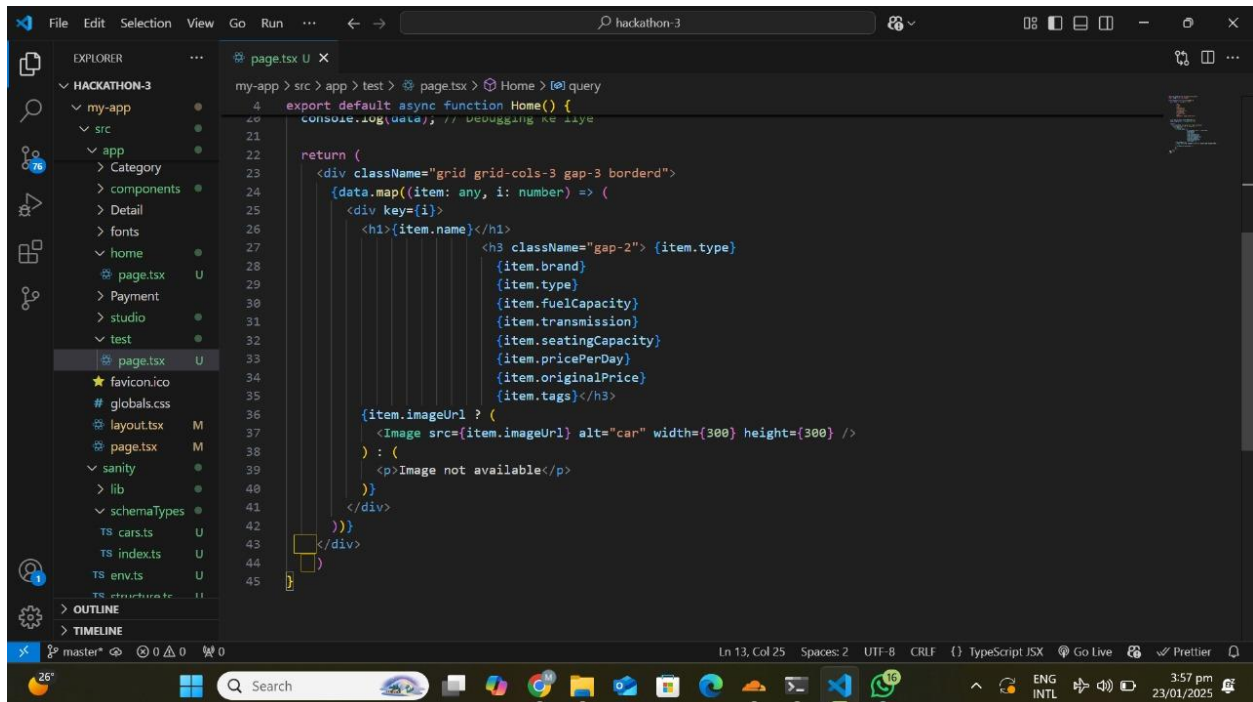
9. Test Folder: -

Created test folder to display the images.

And content in vs code shown in screen shot.

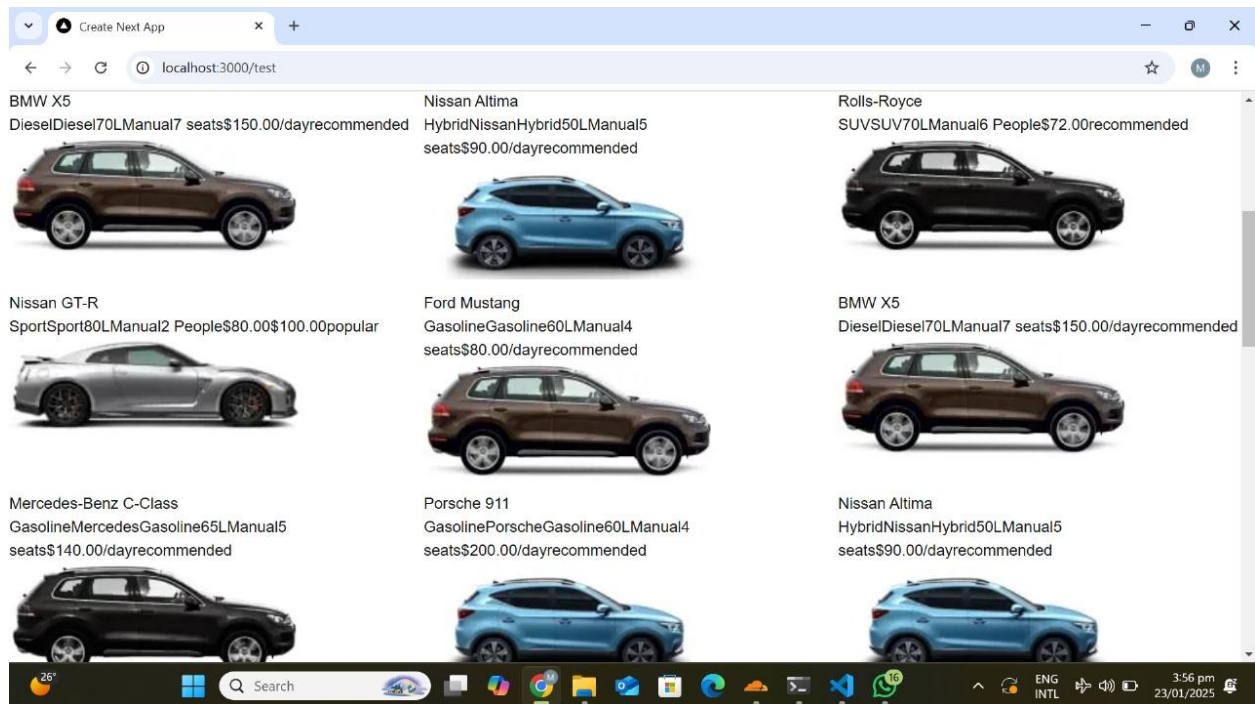


```
1 import { client } from "@sanity/lib/client";
2 import Image from "next/image";
3
4 export default async function Home() {
5   const query = `*[_type == "car"]`
6
7   {
8     name,
9     brand,
10    type,
11    fuelCapacity,
12    transmission,
13    seatingCapacity,
14    pricePerDay,
15    originalPrice,
16    tags,
17    "imageUrl": image.asset->url`;
18
19   const data = await client.fetch(query);
20   console.log(data); // Debugging ke liye
21
22   return (
23     <div className="grid grid-cols-3 gap-3 border">
24       {data.map((item: any, i: number) => (
25         <div key={i}>
26           <h1>{item.name}</h1>
27           <h3 className="gap-2"> {item.type}
```



```
28     {item.brand}
29     {item.type}
30     {item.fuelCapacity}
31     {item.transmission}
32     {item.seatingCapacity}
33     {item.pricePerDay}
34     {item.originalPrice}
35     {item.tags}</h3>
36     {item.imageUrl ? (
37       <Image src={item.imageUrl} alt="car" width={300} height={300} />
38     ) : (
39       <p>Image not available</p>
40     )}
41   </div>
42 )}
43 </div>
44 )
45 }
```

Then display our data: -



Then we put data into our web site.