

# "Marketplace Technical Foundation - [Car-Rental Website]"

Day 3: API Data Migration to Sanity & Integration with Next.js:

## Integrating API with Sanity in a Next.js Project

### Step-by-Step Guide:

#### 1. Creating a Sanity Project:

- Visit [sanity.io](https://sanity.io) and create a new project.
- Note down the `projectId`, `dataset`, and generate an API token for accessing the project.

#### 2. Installing Sanity in the Hackathon Project:

- Add Sanity to your existing project using the terminal:  
`npm install @sanity/client`

#### 3. Setting Up Environment Variables:

- Create a `.env.local` file in the root of your project.
- Add the following variables:  
`SANITY_PROJECT_ID=your_project_id`
- `SANITY_DATASET=your_dataset`
  - `SANITY_API_TOKEN=your_api_token`

#### 4. Adding the Provided API to the Project:

- In the root directory of your project, create a `scripts` folder.
- Inside the `scripts` folder, create a file named `importTemplateData7.mjs`.
- Copy and paste the provided API logic into this file.

The screenshot shows the VS Code editor with the file `importTemplate7Data.mjs` open. The Explorer sidebar on the left shows the project structure with folders like `.next`, `node_modules`, `public`, and `scripts`. The `scripts` folder is expanded, showing `importTemplate7Data.mjs`. The main editor displays the following JavaScript code:

```

37 async function importData() {
38   try {
39     console.log('Fetching car data from API...');
40
41     // API endpoint containing car data
42     const response = await axios.get('https://sanity-nextjs-application.vercel.app/api/hackathon/template7');
43     const cars = response.data;
44
45     console.log(`Fetched ${cars.length} cars`);
46
47     for (const car of cars) {
48       console.log(`Processing car: ${car.name}`);
49
50       let imageRef = null;
51       if (car.image_url) {
52         imageRef = await uploadImageToSanity(car.image_url);
53       }
54
55       const sanityCar = {
56         _type: 'car',
57         name: car.name,
58         brand: car.brand || null,
59         type: car.type,
60         fuelCapacity: car.fuel_capacity,
61         transmission: car.transmission,
62         seatingCapacity: car.seating_capacity,
63         pricePerDay: car.price_per_day,
64         originalPrice: car.original_price || null,
65         tags: car.tags || [],
66         image: imageRef ? {
67           _type: 'image',
68           asset: {

```

The status bar at the bottom indicates the file is in the `main` branch, with 0 errors and 0 warnings. The bottom right corner shows the status bar with `Ln 1, Col 1`, `Spaces: 2`, `UTF-8`, `CRLF`, `JavaScript`, `Go Live`, and `Prettier`.

## 5. Updating `package.json`:

- Open `package.json` and add the following script:  
"import-data": "node scripts/importTemplateData7.mjs"

The screenshot shows the VS Code editor with the `package.json` file open. The Explorer sidebar on the left shows the project structure with folders like `.next`, `node_modules`, `public`, and `scripts`. The `scripts` folder is expanded, showing `importTemplate7Data.mjs`. The main editor displays the following JSON content:

```

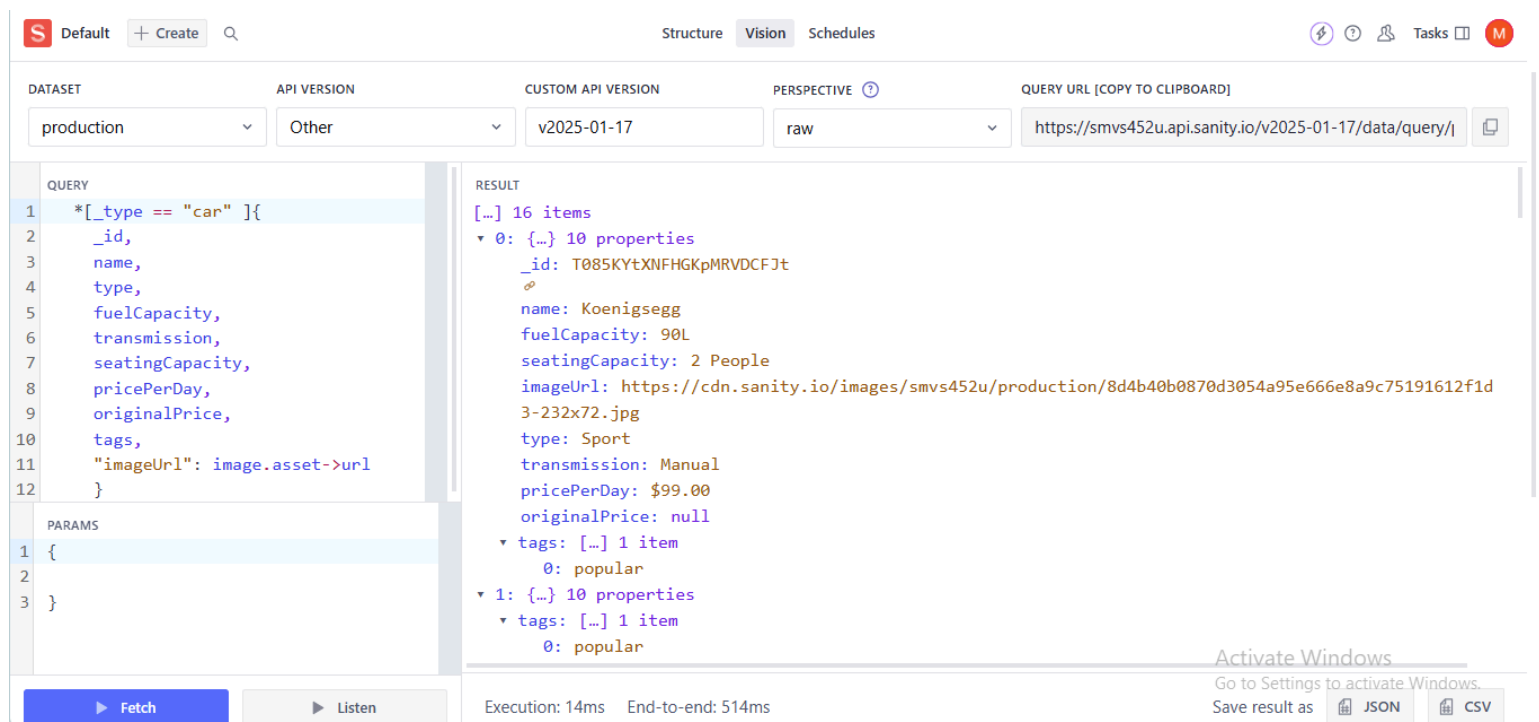
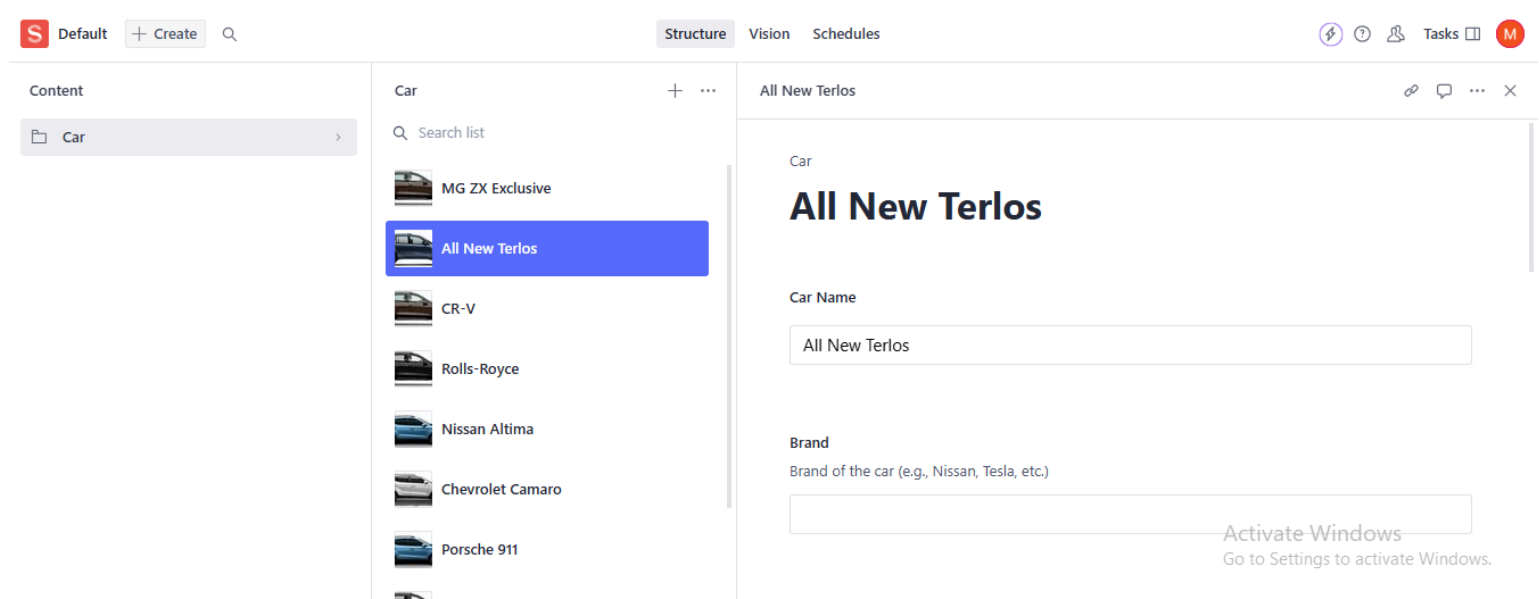
1 {
2   "name": "hackathon",
3   "version": "0.1.0",
4   "private": true,
5   "scripts": {
6     "dev": "next dev",
7     "build": "next build",
8     "start": "next start",
9     "lint": "next lint",
10    "import-data": "node scripts/importTemplate7Data.mjs"
11  },

```

The status bar at the bottom indicates the file is in the `main` branch, with 0 errors and 0 warnings. The bottom right corner shows the status bar with `Ln 1, Col 1`, `Spaces: 2`, `UTF-8`, `CRLF`, `JavaScript`, `Go Live`, and `Prettier`.

## 6. Running the Import Script:

- Use the terminal to run the script:  
npm run import-data
- This command imports the data into Sanity.



## 7. Creating the Car Schema in Sanity:

- Navigate to the **sanity** folder in your project.
- Inside the **schemas** folder, create a new file named **car.ts**.

The screenshot shows the Visual Studio Code interface with a project named 'hackathon-ui-ux'. The Explorer sidebar on the left shows the file structure, with the 'sanity' folder expanded to show 'schemaTypes'. Inside 'schemaTypes', the file 'car.ts' is selected. The main editor displays the content of 'car.ts', which defines a schema for a car. The breadcrumb at the top of the editor reads: 'src > sanity > schemaTypes > TS cars.ts > [1] carSchema > fields'.

```

1  const carSchema = {
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       name: 'transmission',
31       type: 'string',
32       title: 'Transmission',

```

## 8. Registering the Schema:

- Import the schema into the **index.ts** file in the **schemas** folder:

```
TS index.ts  X
src > sanity > schemaTypes > TS index.ts > ...
1  import { type SchemaTypeDefinition } from 'sanity';
2  import cars from './cars';
3
4  export const schema: { types: SchemaTypeDefinition[] } = {
5    types: [cars],
6  }
7
```

### Final Notes:

- Ensure you run the Sanity Studio locally to verify the schema and data.
- Test the integration by fetching the car data in your Next.js application using `@sanity/client`.

```
TS client.ts  X
src > sanity > lib > TS client.ts > ...
1  import { createClient } from 'next-sanity'
2
3  import { apiVersion, dataset, projectId } from '../env'
4
5  export const client = createClient({
6    projectId,
7    dataset,
8    apiVersion,
9    useCdn: true, // Set to false if statically generating pages, using ISR or tag-based revalidation
10 })
11
```