

Document Title: "Day 3 - API Integration Report – [MORENT]"

INSTALL NEXT.JS Framework

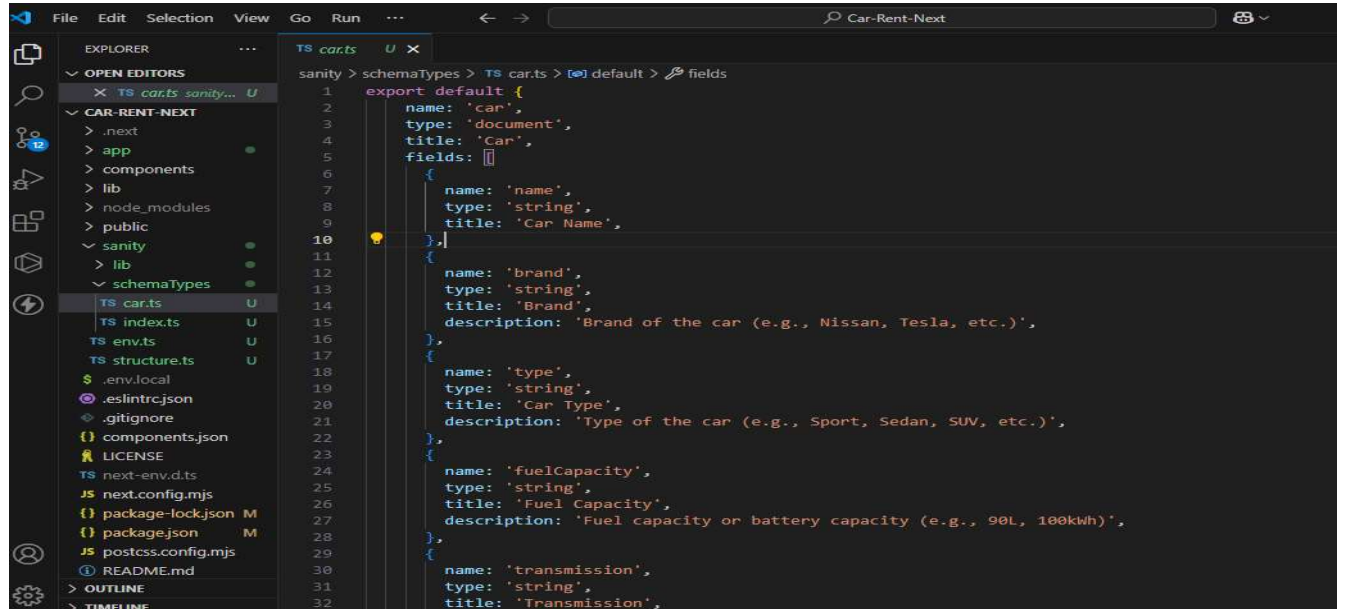
```
E:\Html Coding\Batch 1 Quarter 2\Hackathon Projects\3rd Hackathon\Functionaly Car-Rent\Car-Rent-Next>npm create sanity@latest -- --project 1ovg4bzy --dataset production
--template clean
You are logged in as umairnana090@gmail.com using Google
Fetching existing projects

? Would you like to add configuration files for a Sanity project in this Next.js folder? Yes
? Do you want to use TypeScript? Yes
? Would you like an embedded Sanity Studio? Yes
? What route do you want to use for the Studio? /studio
? Select project template to use Clean project with no predefined schema types
? Would you like to add the project ID and dataset to your .env.local file? Yes
Added http://localhost:3000 to CORS origins
Running 'npm install --legacy-peer-deps --save @sanity/vision@3 sanity@3 @sanity/image-url@1 styled-components@6'
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: 'rimraf@6.0.1',
npm WARN EBADENGINE   required: { node: '20 || >=22' },
npm WARN EBADENGINE   current: { node: 'v21.2.0', npm: '10.2.3' }
npm WARN EBADENGINE }
```

CREATE A PROJECT IN SANITY

The screenshot shows the Sanity dashboard for a project named "Car-Rent". The user is logged in as "Mr. USA". The project ID is "1ovg4bzy" and the dataset is "production". The plan is "Growth Trial" and the status is "Active". The dashboard includes a navigation bar with links to "Getting started", "Overview", "Members", "Studios", "Datasets", "Access", "Activity", "Usage", "Plan", "API", and "Settings". The "API" tab is selected, showing "Webhooks" and "GROQ-powered webhooks". There are 0 of 2 webhooks (2 included in plan). A "Create webhook" button is visible. A "What's new" section at the bottom left mentions "Sanity Create Content Mapping, Visual Editing, and Content Releases". A large box in the center says "There are no GROQ-powered webhooks in this project".

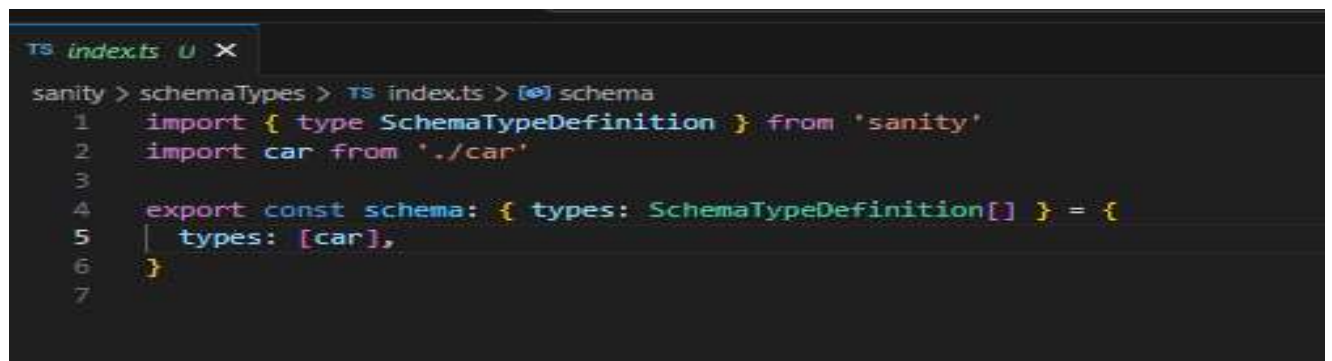
Make Schema of Cars



The screenshot shows the VS Code editor with the file explorer on the left and the editor window on the right. The file explorer shows the project structure with the following files and folders: .next, app, components, lib, node_modules, public, sanity, lib, schemaTypes, TS car.ts, TS index.ts, TS env.ts, TS structure.ts, .env.local, .eslintrc.json, .gitignore, components.json, LICENSE, next-env.d.ts, next.config.mjs, package-lock.json, package.json, postcss.config.mjs, and README.md. The editor window shows the content of TS car.ts, which defines a schema for cars. The schema is a Sanity schema type definition with the following fields: name (string), type (string), title (string), description (string), fuelCapacity (string), and transmission (string). The schema is exported as a default export.

```
sanity > schemaTypes > TS car.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      name: 'transmission',
31      type: 'string',
32      title: 'Transmission',
33    }
34  ]
35 }
```

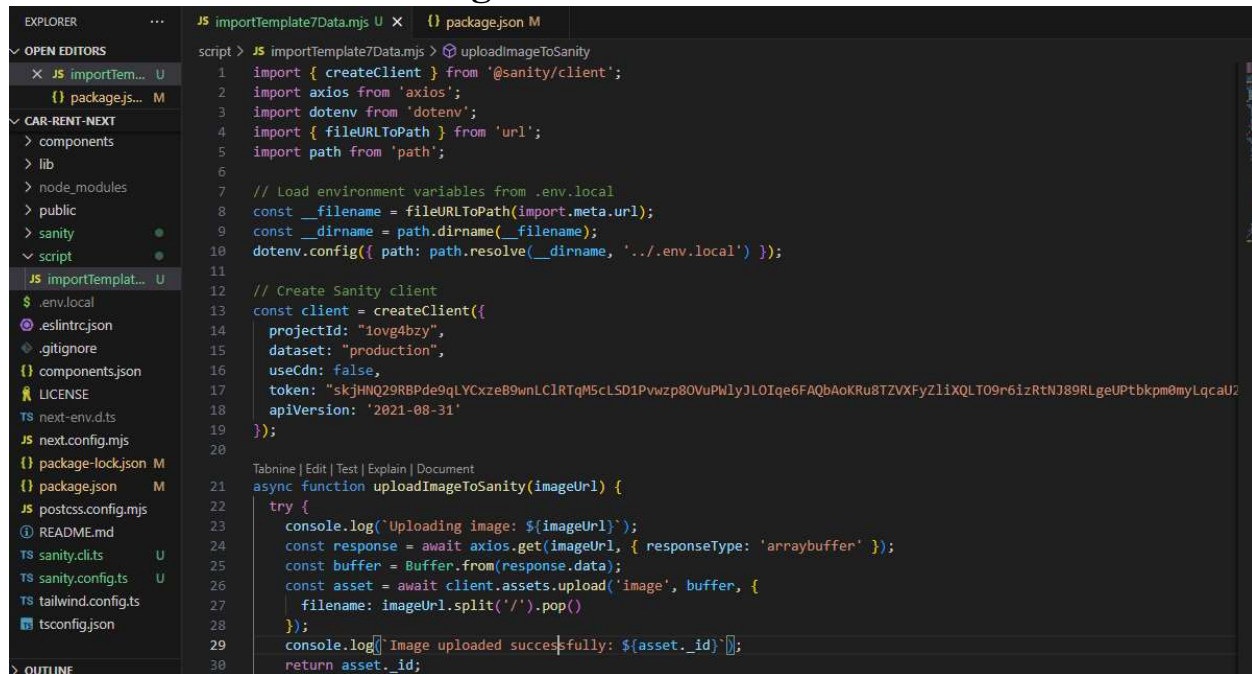
Set Schema File in index.ts to show in STUDIO



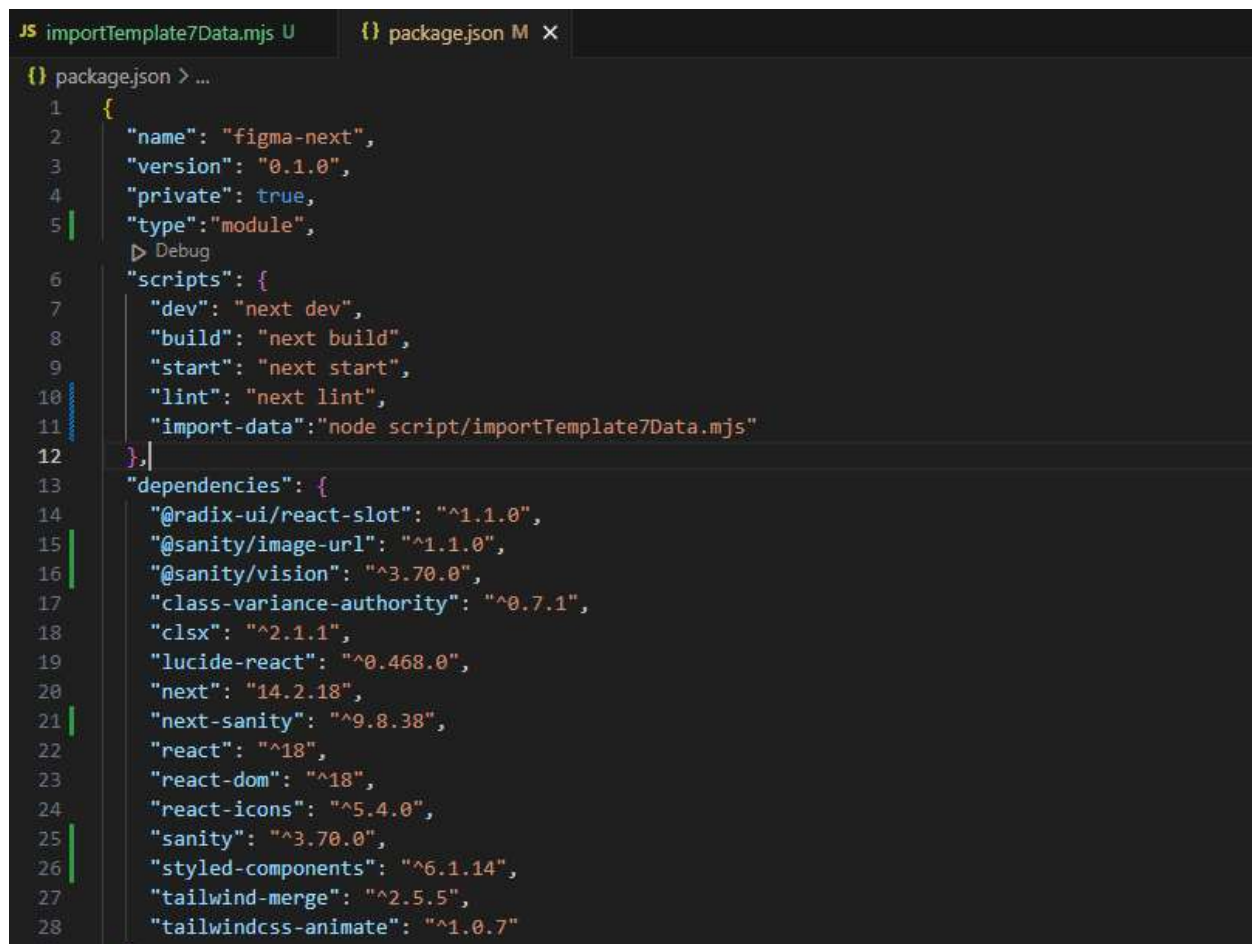
The screenshot shows the VS Code editor with the file explorer on the left and the editor window on the right. The file explorer shows the project structure with the following files and folders: .next, app, components, lib, node_modules, public, sanity, lib, schemaTypes, TS car.ts, TS index.ts, TS env.ts, TS structure.ts, .env.local, .eslintrc.json, .gitignore, components.json, LICENSE, next-env.d.ts, next.config.mjs, package-lock.json, package.json, postcss.config.mjs, and README.md. The editor window shows the content of TS index.ts, which sets up the schema for the studio. The schema is a Sanity schema type definition with the following fields: name (string), type (string), title (string), description (string), fuelCapacity (string), and transmission (string). The schema is exported as a default export.

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

Migration of Cars Data

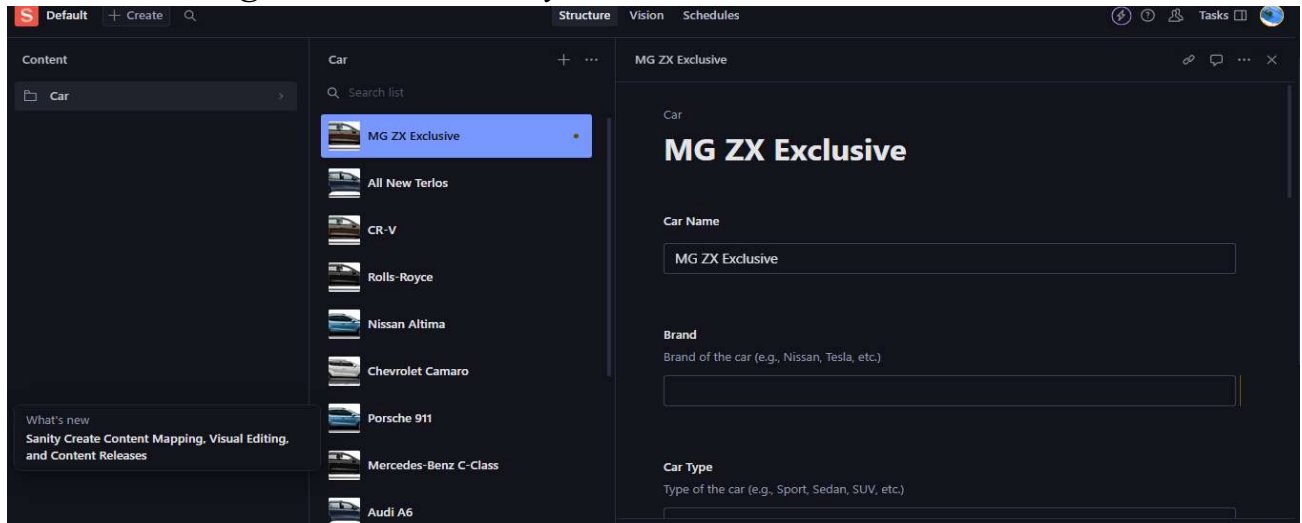


```
script > JS importTemplate7Data.mjs > uploadImageToSanity
1 import { createClient } from '@sanity/client';
2 import axios from 'axios';
3 import dotenv from 'dotenv';
4 import { fileURLToPath } from 'url';
5 import path from 'path';
6
7 // Load environment variables from .env.local
8 const __filename = 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: "1ovg4bzy",
15   dataset: "production",
16   useCdn: false,
17   token: "skjHnQ29RBPde9qLYCxeB9wnLCIRtqM5cLSD1Pvwzp80VuPWlyJLOIqe6FAQbAoKRU8TZVXFyZliXQLT09r6izRtNJ89RLgeUPtbkpm0myLqcaU2",
18   apiVersion: '2021-08-31'
19 });
20
21 Tabnine | Edit | Test | Explain | Document
22 async function uploadImageToSanity(imageUrl) {
23   try {
24     console.log('Uploading image: ${imageUrl}');
25     const response = await axios.get(imageUrl, { responseType: 'arraybuffer' });
26     const buffer = Buffer.from(response.data);
27     const asset = await client.assets.upload('image', buffer, {
28       filename: imageUrl.split('/').pop()
29     });
30     console.log('Image uploaded successfully: ${asset.id}');
31     return asset._id;
32   } catch (error) {
33     console.error('Error uploading image: ', error);
34   }
35 }
```



```
JS importTemplate7Data.mjs U {} package.json M X
{} package.json > ...
1 {
2   "name": "figma-next",
3   "version": "0.1.0",
4   "private": true,
5   "type": "module",
6   "scripts": {
7     "dev": "next dev",
8     "build": "next build",
9     "start": "next start",
10    "lint": "next lint",
11    "import-data": "node script/importTemplate7Data.mjs"
12  },
13  "dependencies": {
14    "@radix-ui/react-slot": "^1.1.0",
15    "@sanity/image-url": "^1.1.0",
16    "@sanity/vision": "^3.70.0",
17    "class-variance-authority": "^0.7.1",
18    "clsx": "^2.1.1",
19    "lucide-react": "^0.468.0",
20    "next": "14.2.18",
21    "next-sanity": "^9.8.38",
22    "react": "^18",
23    "react-dom": "^18",
24    "react-icons": "^5.4.0",
25    "sanity": "^3.70.0",
26    "styled-components": "^6.1.14",
27    "tailwind-merge": "^2.5.5",
28    "tailwindcss-animate": "^1.0.7"
29  }
}
```

Show This Migration Data in my STUDIO :

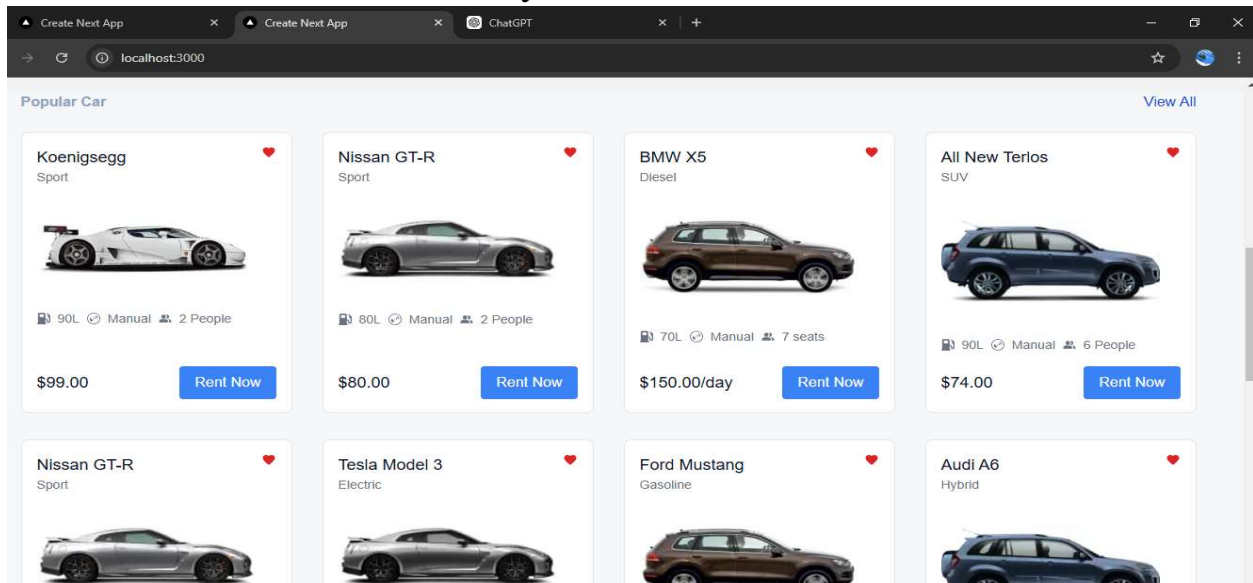


Fetch Data From GROQ Language

```
const fetching = `*[_type == "car"]{
  _id,
  name,
  type,
  pricePerDay,
  originalPrice,
  fuelCapacity,
  seatingCapacity,
  transmission,
  brand,
  tags,
  "imageUrl": image.asset->url,
  _createdAt,
  _updatedAt
}[0..11]`

const data = await client.fetch(fetching)
console.log(data);
```

The Cars Data are Successfully Show:



The Successfully Car Data is Migrated and Show in My Website.