Day 3 of Hackathone

API Integration and Data Migration

1. Project Title:

Dynamic Product Display Using Sanity and Next.js

2. Objective:

The goal of this project was to fetch product data from Sanity CMS and display it dynamically on a Next.js frontend with proper styling and responsiveness.

3. Key Features:

- Sanity Integration: Successfully connected Sanity CMS to Next.js using GROQ queries.
- Dynamic Data Fetching: Used Sanity's APIs to retrieve product details, including name, price, description, and image.
- Responsive Frontend Design: Built a responsive layout using Tailwind CSS, ensuring compatibility across devices.
- Clean Code Structure: Used modular functions for fetching data and organized react component efficiently.

4. Technologies Used:

- Frontend: Next.js (React Framework)
- Backend: Sanity CMS
- Styling: Tailwind CSS
- Programming Language: TypeScript/JavaScrip

5. Step-by-Step Implementation:

1. Set Up Sanity CMS:

- Created a new dataset in Sanity.
- Added a schema for products with fields like name, price, description, image, a category.

2. Configured Sanity Client:

- Installed the Sanity client in the Next.js project.
- Set up a reusable client instance to connect to Sanity.

3. Created GROQ Query:

- Wrote a GROQ query to fetch the required product field

4. Fetched Data in Next.js:

- Used a custom fetch Products function to call Sanity's APIs.
- Managed the fetched data using React's useState and useEffect hook

5. Frontend Rendering:

- Dynamically rendered product details, including name, price, description, and images.
- Used Image from Next.js for optimized image loading.

6. Styling with Tailwind CSS:

- Designed a responsive grid layout.
- Styled individual product cards with hover effects for better user interactio .

6. Challenges and Solutions:

- Challenge: Handling dynamic images from Sanity in Next.js. Solution: Used next/image and created a function to generate image URLs from Sanity assets.
- Challenge: Managing dynamic and real-time updates from Sanity. Solution: Ensured data fetching and rendering are handled efficiently with optimize GROQ queries.

7. Output:

- A fully functional webpage that dynamically displays product data from Sanity CMS in a clean and responsive design.

9. Learning Outcome:

Through this project, I gained hands-on experience with:

- Connecting a CMS to a modern frontend framework.
- Writing GROQ queries for fetching data efficient .
- Implementing responsive designs with Tailwind CSS.

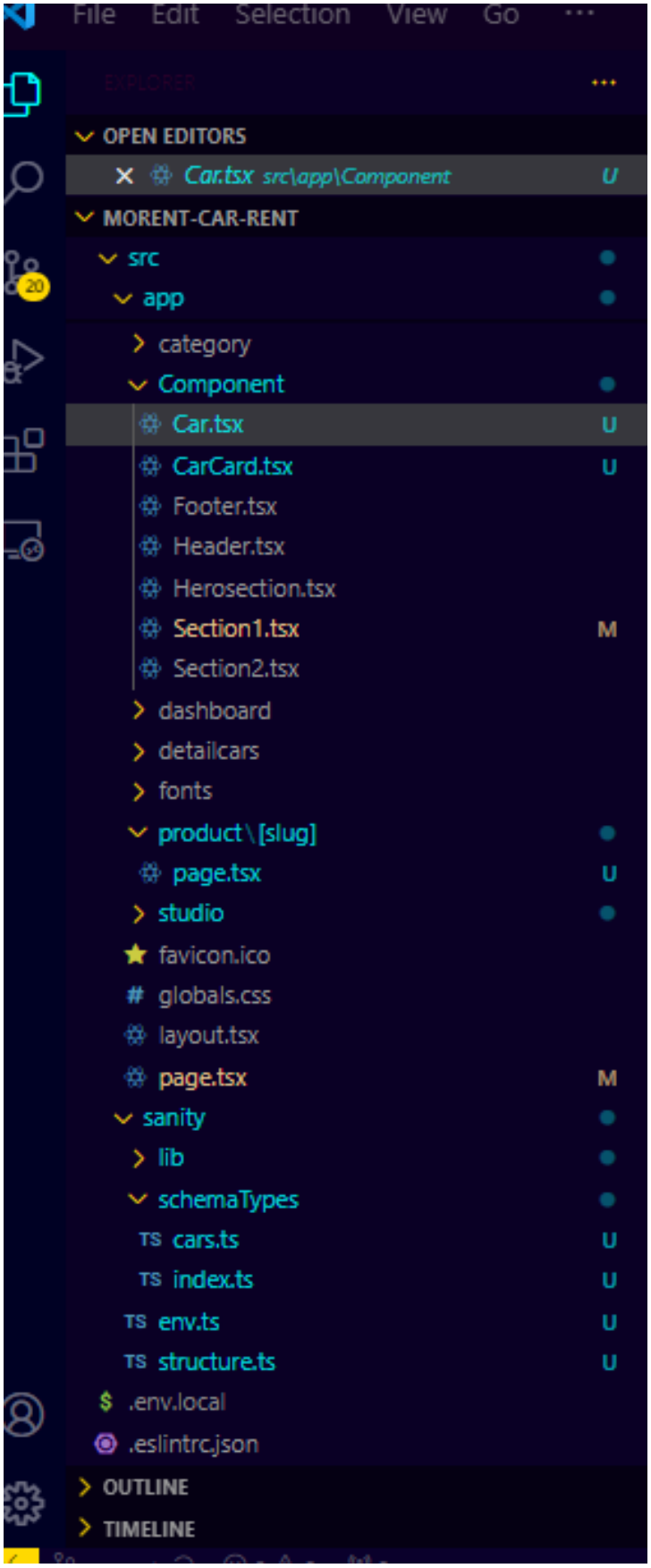
10. Conclusion:

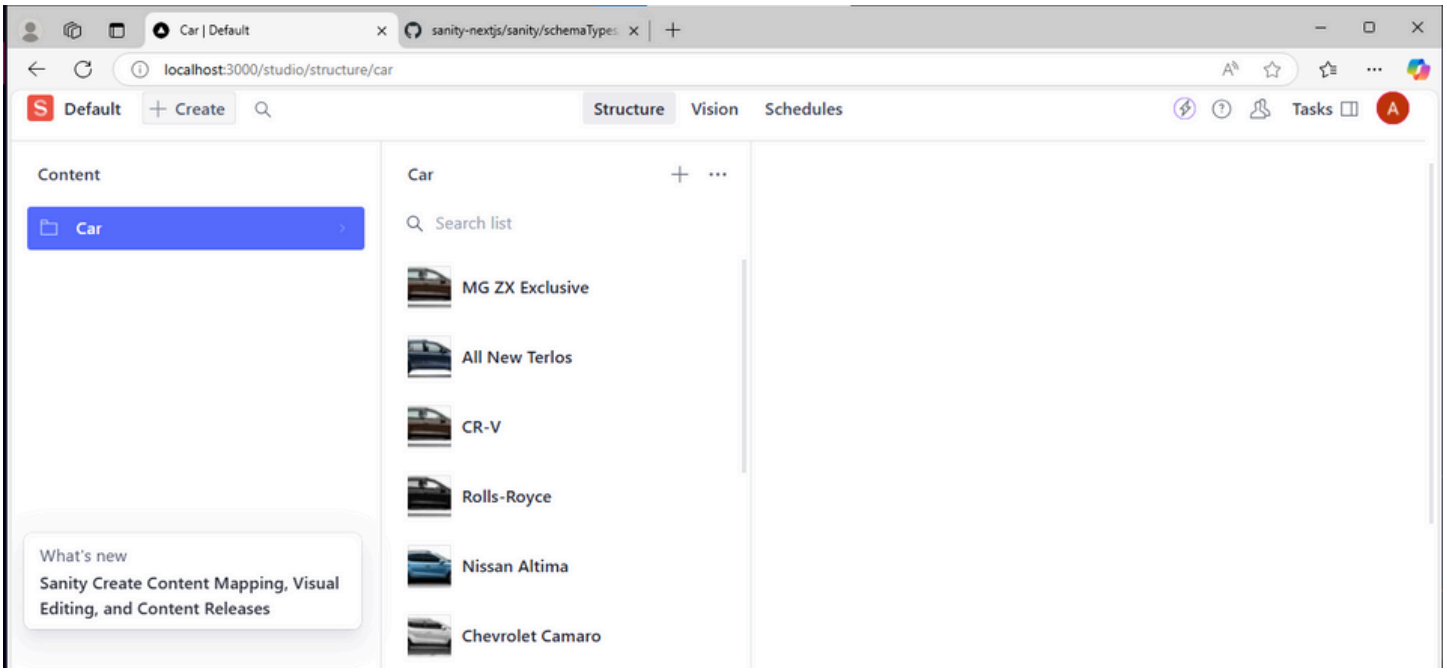
This project demonstrates the ability to integrate a CMS backend (Sanity) with a modern frontend framework (Next.js) for dynamic content rendering, providing a robust and scalable solution for real-world applications.

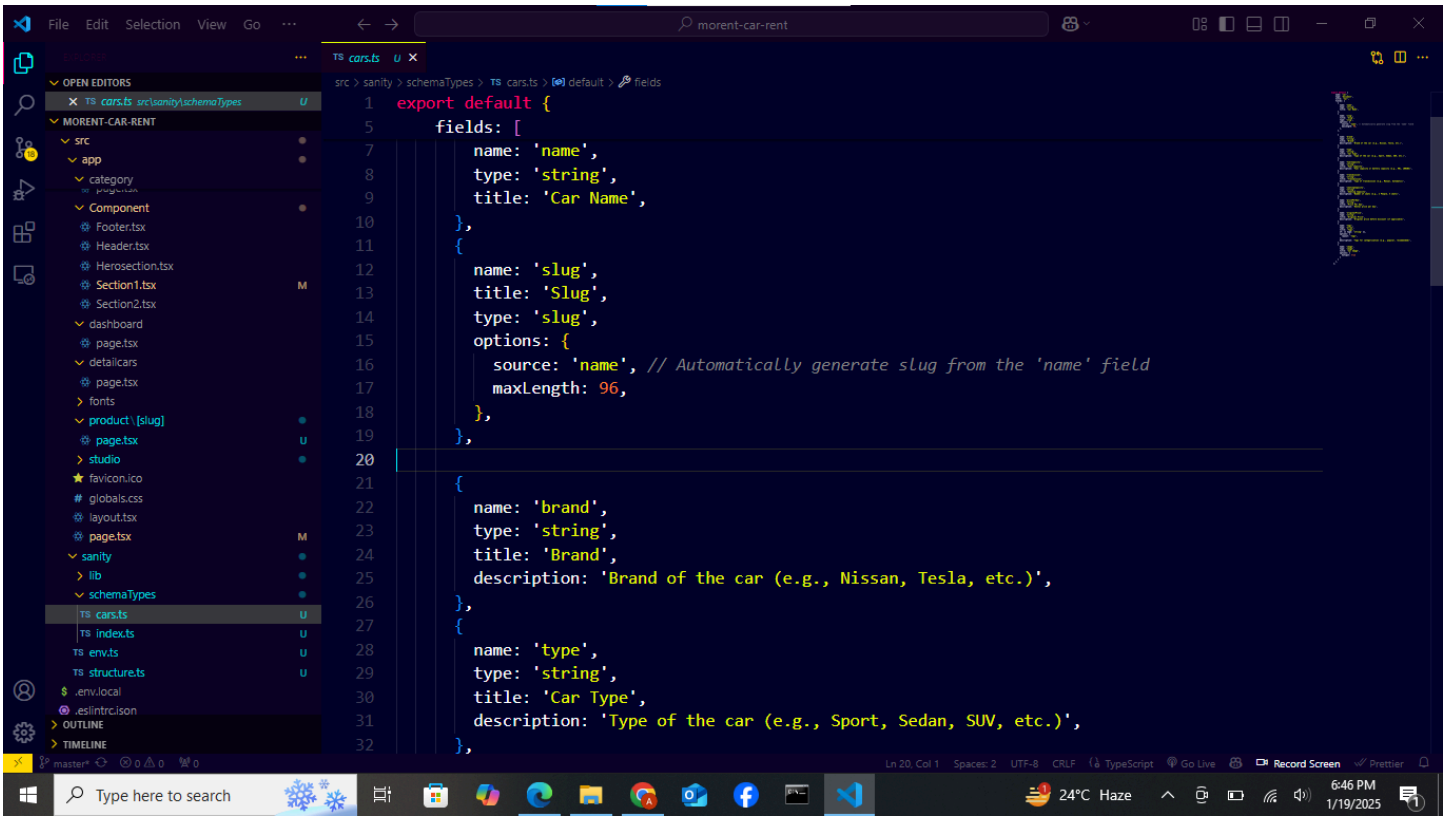
Hackathon Day 3 Task Submission

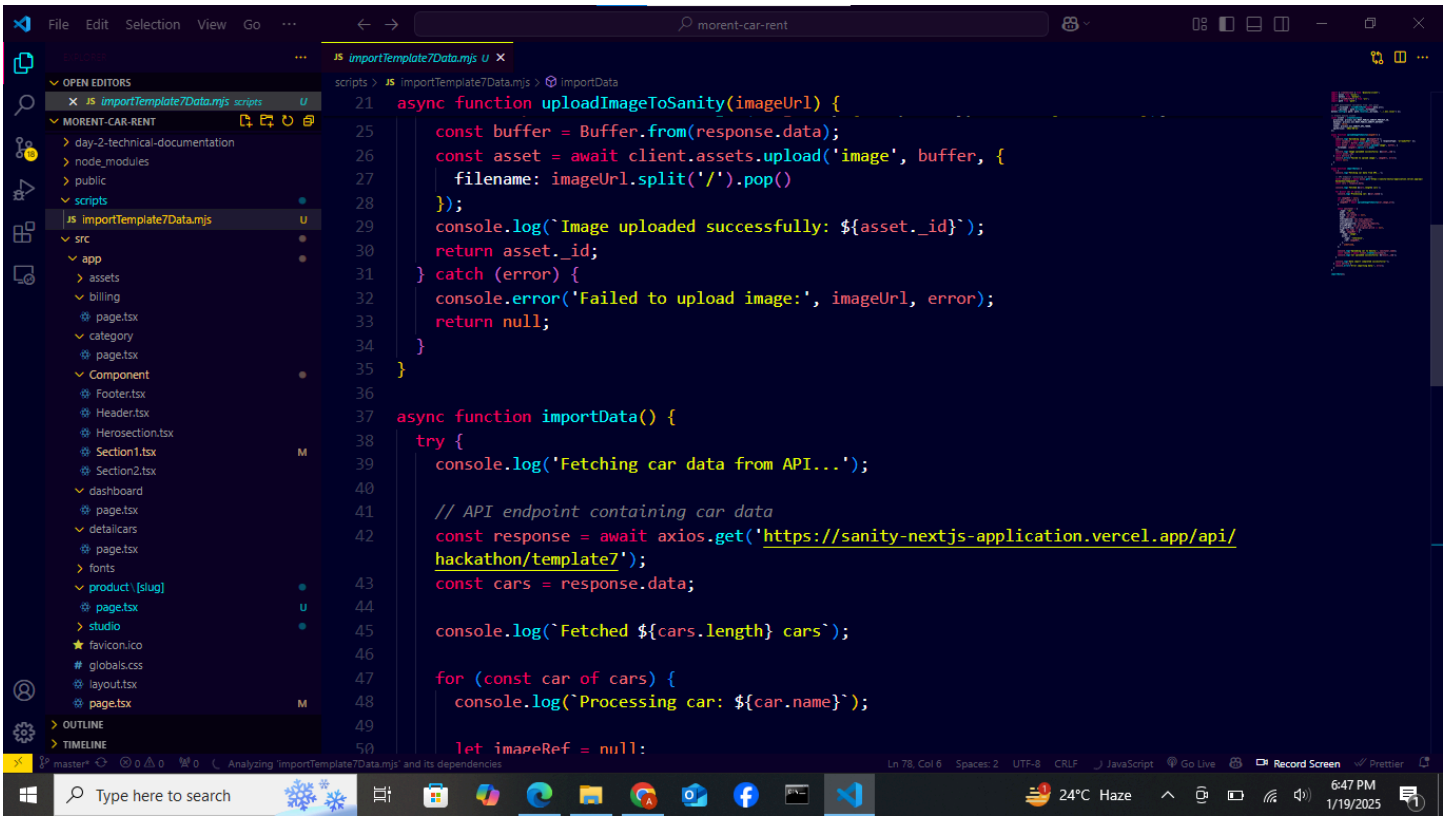
Name: Aliza Faizan

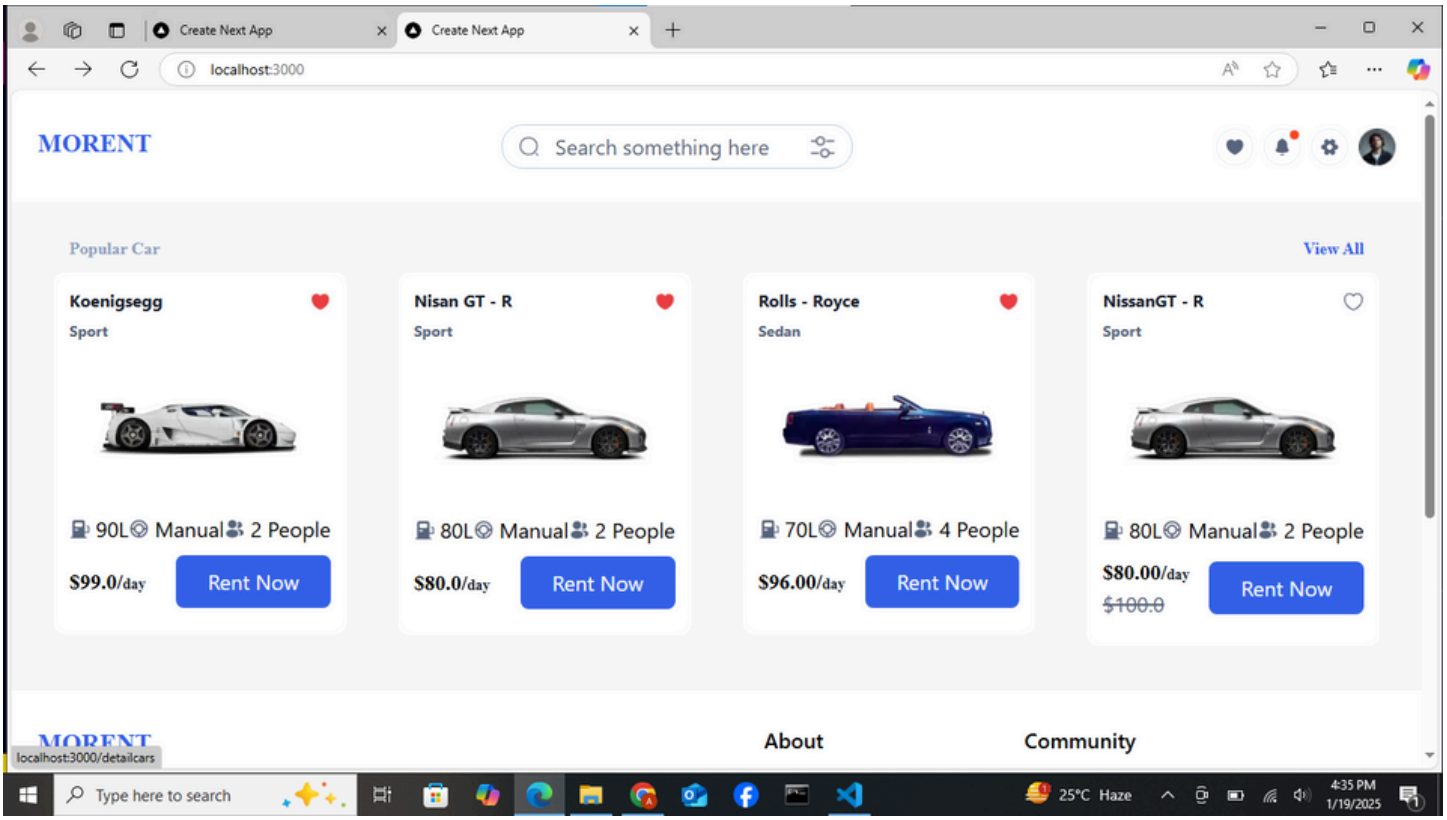
Project Name: Sanity: A Step-by-Step Guide


File Edit Selection View Go
EXPLORER
OPEN EDITORS
Car.tsx src\app\Component U
MORENT-CAR-RENT
src
app
category
Component
Car.tsx U
CarCard.tsx U
Footer.tsx
Header.tsx
Herosection.tsx
Section1.tsx M
Section2.tsx
dashboard
detailcars
fonts
product\[slug]
page.tsx U
studio
favicon.ico
globals.css
layout.tsx
page.tsx M
sanity
lib
schemaTypes
cars.ts U
index.ts U
env.ts U
structure.ts U
.env.local
.eslintrc.json
OUTLINE
TIMELINE


Car | Default
sanity-nextjs/sanity/schemaTypes
localhost:3000/studio/structure/car
Default
Create
Structure
Vision
Schedules
Tasks
Content
Car
Car
Search list
MG ZX Exclusive
All New Terlos
CR-V
Rolls-Royce
Nissan Altima
Chevrolet Camaro
What's new
Sanity Create Content Mapping, Visual Editing, and Content Releases


File Edit Selection View Go
morent-car-rent
cars.ts
src > sanity > schemaTypes > cars.ts > default > fields
OPEN EDITORS
MORENT-CAR-RENT
src
app
Component
Footer.tsx
Header.tsx
Herosection.tsx
Section1.tsx
Section2.tsx
dashboard
page.tsx
detailcars
page.tsx
fonts
product\[slug]
page.tsx
studio
favicon.ico
globals.css
layout.tsx
page.tsx
sanity
lib
schemaTypes
cars.ts
index.ts
env.ts
structure.ts
.env.local
.eslintrc.json
OUTLINE
TIMELINE
export default {
fields: [
{
name: 'name',
type: 'string',
title: 'Car Name',
},
{
name: 'slug',
title: 'Slug',
type: 'slug',
options: {
source: 'name', // Automatically generate slug from the 'name' field
maxLength: 96,
},
},
{
name: 'brand',
type: 'string',
title: 'Brand',
description: 'Brand of the car (e.g., Nissan, Tesla, etc.)',
},
{
name: 'type',
type: 'string',
title: 'Car Type',
description: 'Type of the car (e.g., Sport, Sedan, SUV, etc.)',
},
Ln 20, Col 1
Spaces: 2
UTF-8
CRLF
TypeScript
Go Live
Record Screen
Prettier
Type here to search
24°C Haze
6:46 PM
1/19/2025





Today All Over Work

Thank You!