Day 3 - API Integration and Data Migration

1. Project Title: Next.js and Sanity for Dynamic Product Display

2. Goal:

This project's objective was to retrieve product data from Sanity CMS and present it dynamically, responsively, and with appropriate styling on a Next.js frontend.

3. Essential Elements:

Sanity Integration: Using GROQ queries, Sanity CMS and Next.js were successfully connected.
Dynamic Data Fetching: Product information such as name, price, description, and image were retrieved using Sanity's APIs.
Responsive Frontend Design: Tailwind CSS was used to create a responsive layout that ensures device compatibility.
Clean Code Structure: React components were effectively arranged and modular functions were used to get data.

4. Technology Employed:

Frontend: React Framework's Next.js
Sanity CMS is the backend.
Tailwind is the style CSS
Language of Programming: TypeScript/JavaScript

5. Implementation in Steps:

Configure Sanity CMS:
-In Sanity, a new dataset was created.
-A product schema with fields for name, price, description, image, and category has been added.
Configure Sanity Client:
-Installed the Sanity client in the Next.js project after configuring it.
-To connect to Sanity, create a reusable client instance.
GROQ Query Creation:
-To retrieve the necessary product fields, a GROQ query was written.
Data Fetched in Next.js:
-Sanity's APIs were called using a bespoke fetchProducts method.
-React's useState and useEffect hooks were used to manage the fetched data.
Frontend Rendering:
-Product data, such as name, price, description, and photos, are dynamically generated.
-To optimize image loading, Next.js's image was used.
Using Tailwind CSS for styling:
-A responsive grid layout was created.
-Hover effects were added to individual product cards to improve user interaction.

6. Problems and Fixes:

Problem: Using Next.js to handle dynamic photos from Sanity.
Solution: To generate image URLs from Sanity assets, a method was developed using next/image.
Problem: Handling Sanity's dynamic and real-time changes.
Solution: Using optimized GROQ queries, it was made sure that data rendering and fetching were done effectively.

7. Output:

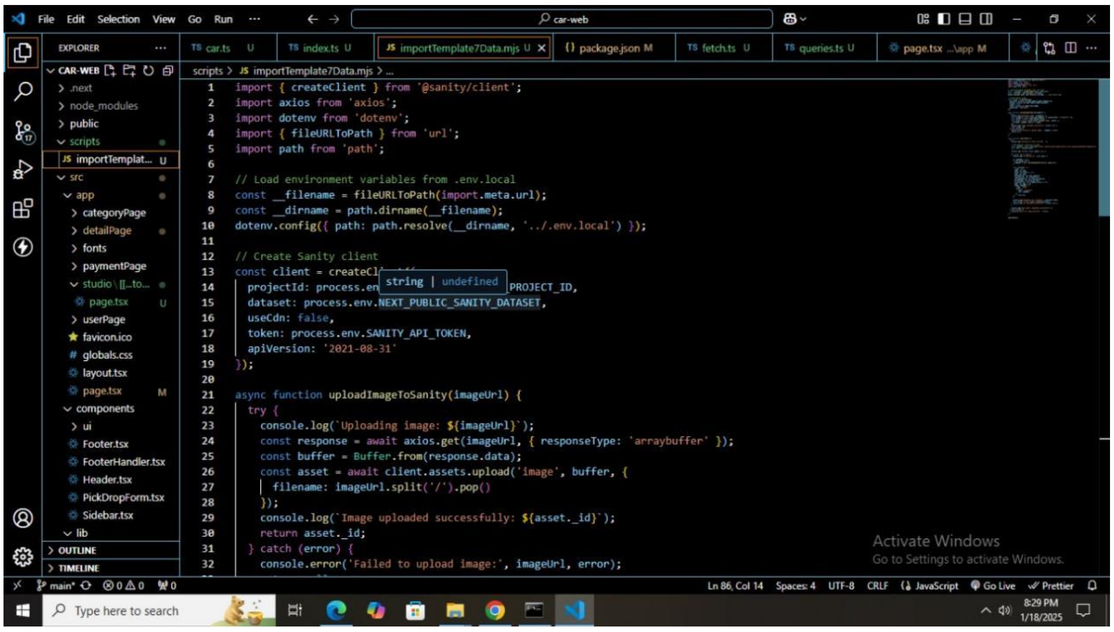
The result is a completely functional webpage with a responsive, clean design that dynamically shows product data from Sanity CMS.

8. Learning Outcome:

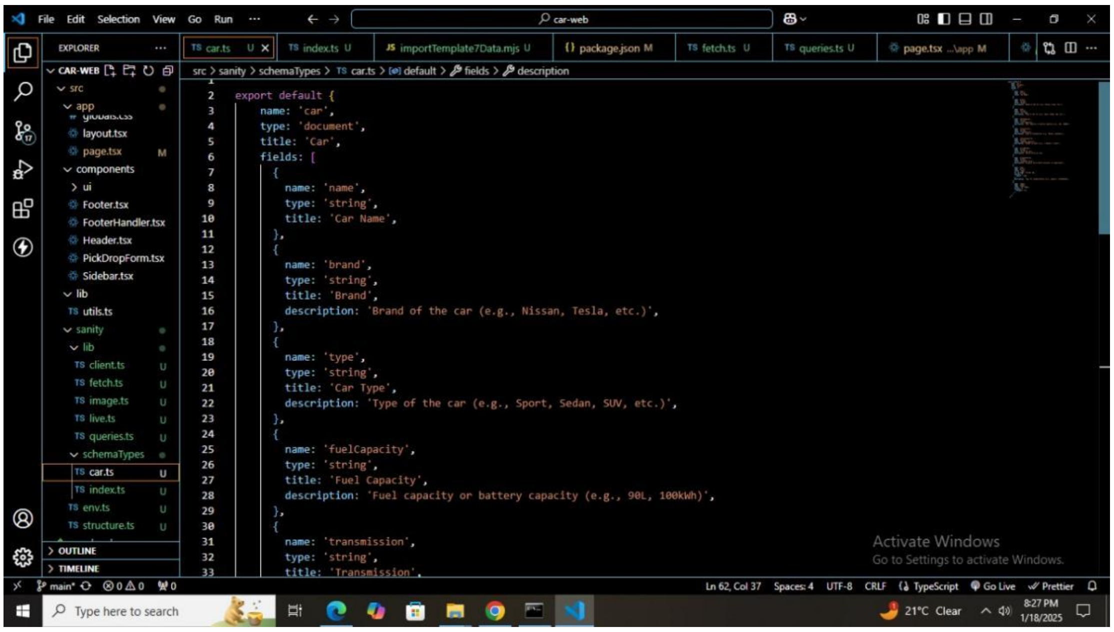
I was able to get practical experience with: ● Linking a CMS to a contemporary frontend framework.
- Creating GROQ queries to effectively retrieve data.
- Using Tailwind CSS to implement responsive designs.

9. Conclusion:

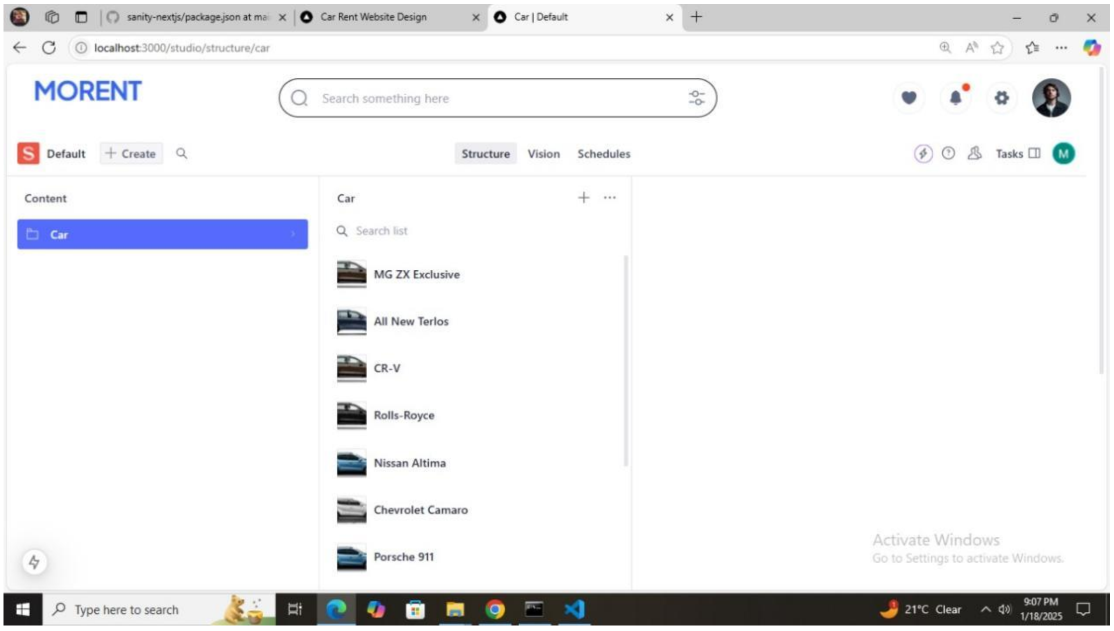
This project shows how to combine a cutting-edge frontend framework (Next.js) with a CMS backend (Sanity) for dynamic content rendering, offering a reliable and expandable solution for practical applications.



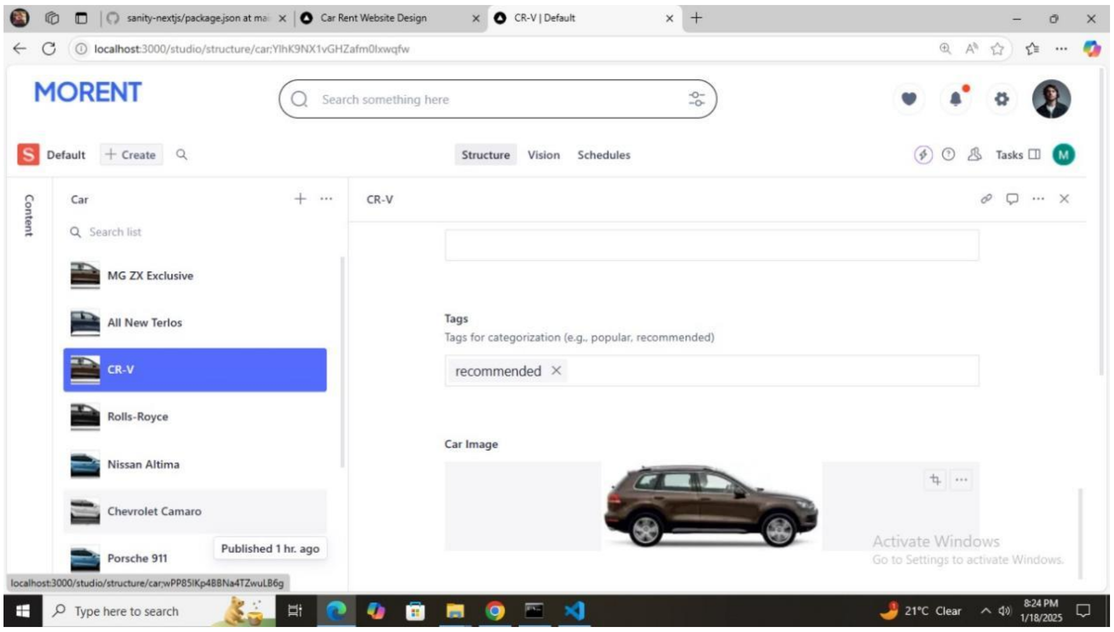
```
import { createClient } from '@sanity/client';
import axios from 'axios';
import dotenv from 'dotenv';
import { fileURLToPath } from 'url';
import path from 'path';

// Load environment variables from .env.local
const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);
dotenv.config({ path: path.resolve(__dirname, '../.env.local') });

// Create Sanity client
const client = createCl
  projectId: process.en string | undefined _PROJECT_ID,
  dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
  useCdn: false,
  token: process.env.SANITY_API_TOKEN,
  apiVersion: '2021-08-31'
});

async function uploadImageToSanity(imageUrl) {
  try {
    console.log(`Uploading image: ${imageUrl}`);
    const response = await axios.get(imageUrl, { responseType: 'arraybuffer' });
    const buffer = Buffer.from(response.data);
    const asset = await client.assets.upload('image', buffer, {
      filename: imageUrl.split('/').pop()
    });
    console.log(`Image uploaded successfully: ${asset._id}`);
    return asset._id;
  } catch (error) {
    console.error('Failed to upload image:', imageUrl, error);
```



```
export default {
    name: 'car',
    type: 'document',
    title: 'Car',
    fields: [
      {
        name: 'name',
        type: 'string',
        title: 'Car Name',
      },
      {
        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.)',
      },
      {
        name: 'fuelCapacity',
        type: 'string',
        title: 'Fuel Capacity',
        description: 'Fuel capacity or battery capacity (e.g., 90L, 100kWh)',
      },
      {
        name: 'transmission',
        type: 'string',
        title: 'Transmission',
```



MG ZX Exclusive
All New Terlos
CR-V
Rolls-Royce
Nissan Altima
Chevrolet Camaro
Porsche 911



sanity-nextjs/package.json at ma
Car Rent Website Design
CR-V | Default
localhost:3000/studio/structure/car;YlhK9NX1vGHZafm0lxwqfw
MORENT
Search something here
Default
Create
Structure
Vision
Schedules
Tasks
Content
Car
Search list
MG ZX Exclusive
All New Terlos
CR-V
Rolls-Royce
Nissan Altima
Chevrolet Camaro
Porsche 911
Published 1 hr. ago
localhost:3000/studio/structure/car;wPP85IKp4BBNa4TZwuL86g
CR-V
Tags
Tags for categorization (e.g., popular, recommended)
recommended
Car Image
Activate Windows
Go to Settings to activate Windows.
Type here to search
21°C Clear
8:24 PM
1/18/2025