I am Tabinda Aijza Ahmed Khan this is my day3 DOCUMENATION of Hackathon3

API Integration and Data Migration

1. Project Title:

Dynamic Product Display Using Sanity (banked) and Next.js(frontend)

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 components efficiently.

4. Technologies Used:

Frontend: Next.js (React Framework)

Backend: Sanity CMS

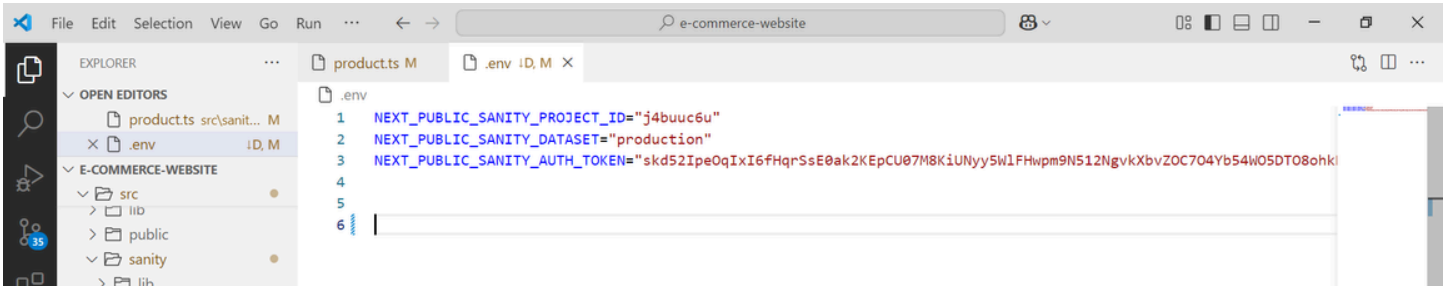
Styling: Tailwind CSS

Programming Language: TypeScript/JavaScript

5. Step-by-Step Implementation:

Set Up Sanity CMS:

Created a new dataset in Sanity.

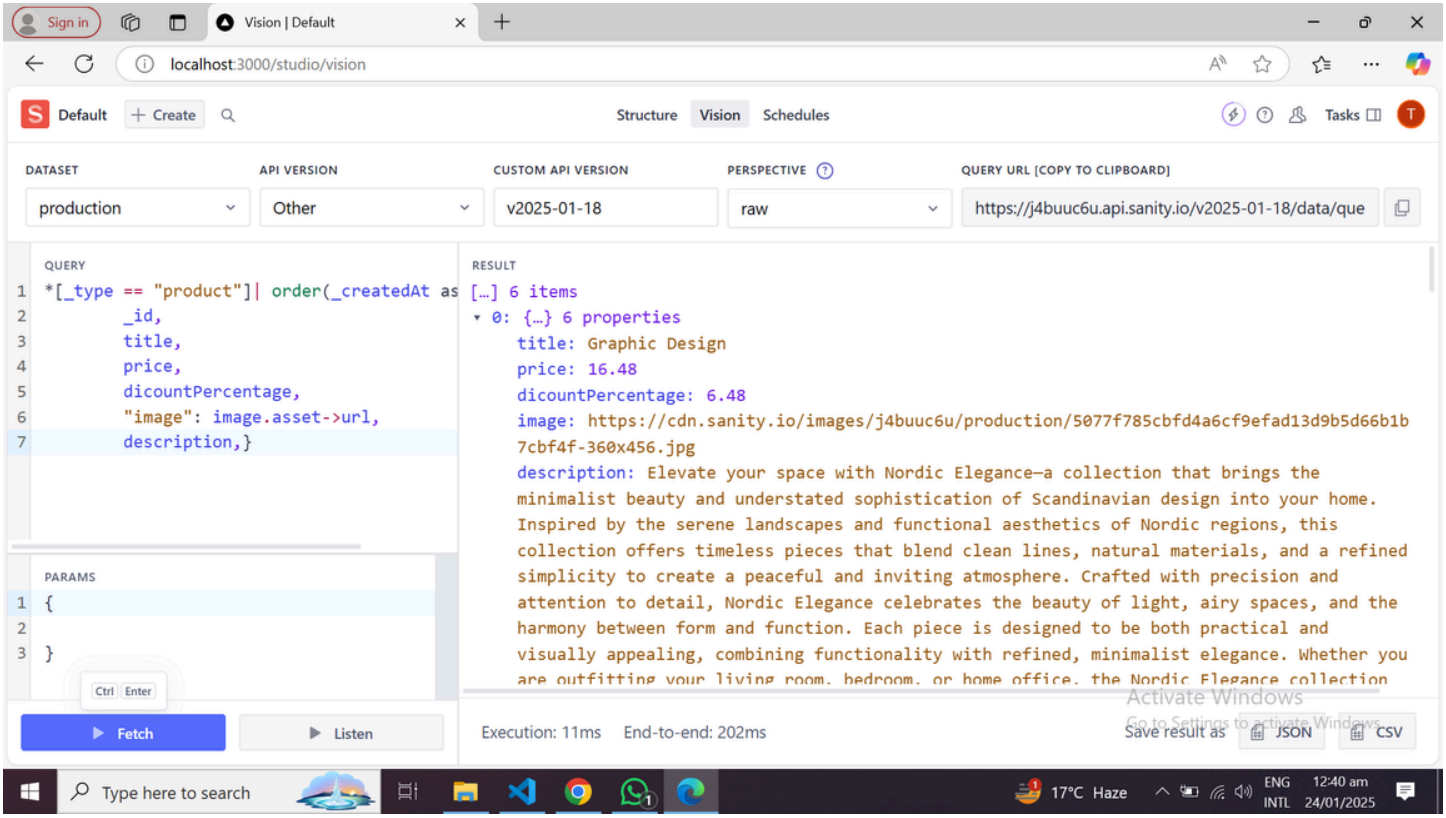


Added a schema for products with fields like name, price, description, image, and category.

Configured Sanity Client:

Set up a reusable client instance to connect to Sanity.

Created GROQ Query:

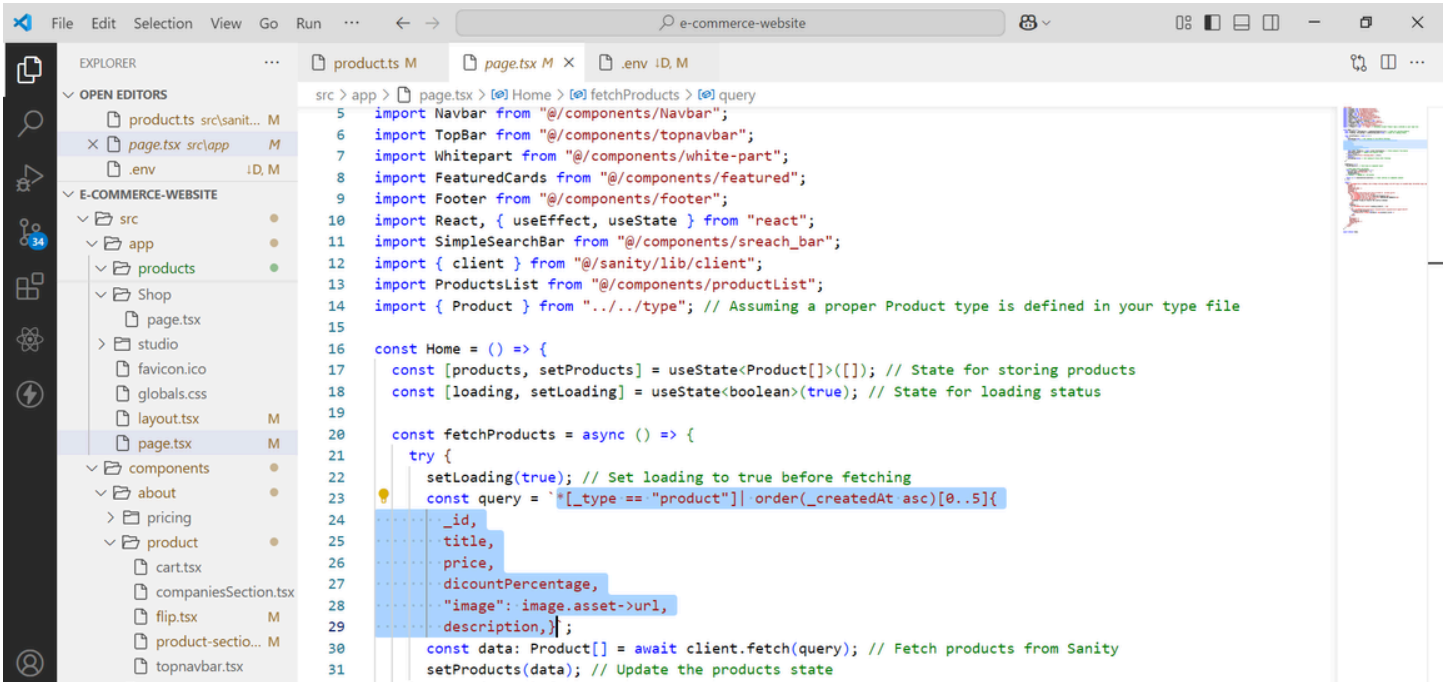


Wrote a GROQ query to fetch the required product field.Fetched Data in Next.js:

Used a custom fetch Products function to call Sanity’s APIs.

Managed the fetched data using React's useState and useEffect hooks.

Frontend Rendering:



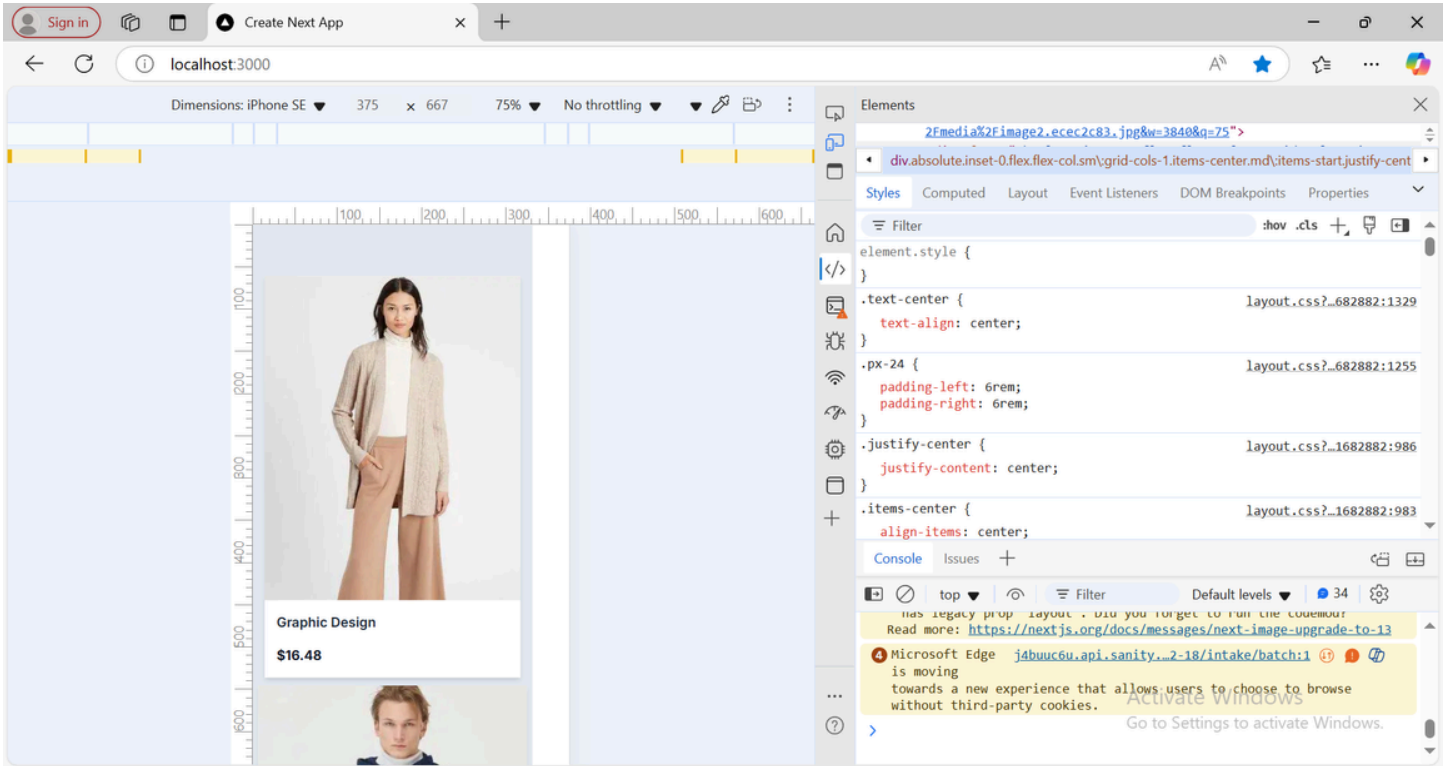
Dynamically rendered product details, including name, price, description, and images.

Used Image from Next.js for optimized image loading

Styling with Tailwind CSS:

Designed a responsive grid layout.

Styled individual product cards with hover effects for better user interaction.



6. Challenges and Solutions:

Dynamic Images from Sanity: Used next/image to generate URLs for Sanity assets.

Dynamic and Real-time Updates: Optimized GROQ queries for efficient data fetching and rendering.

7. Output:

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

8. Learning Outcome:

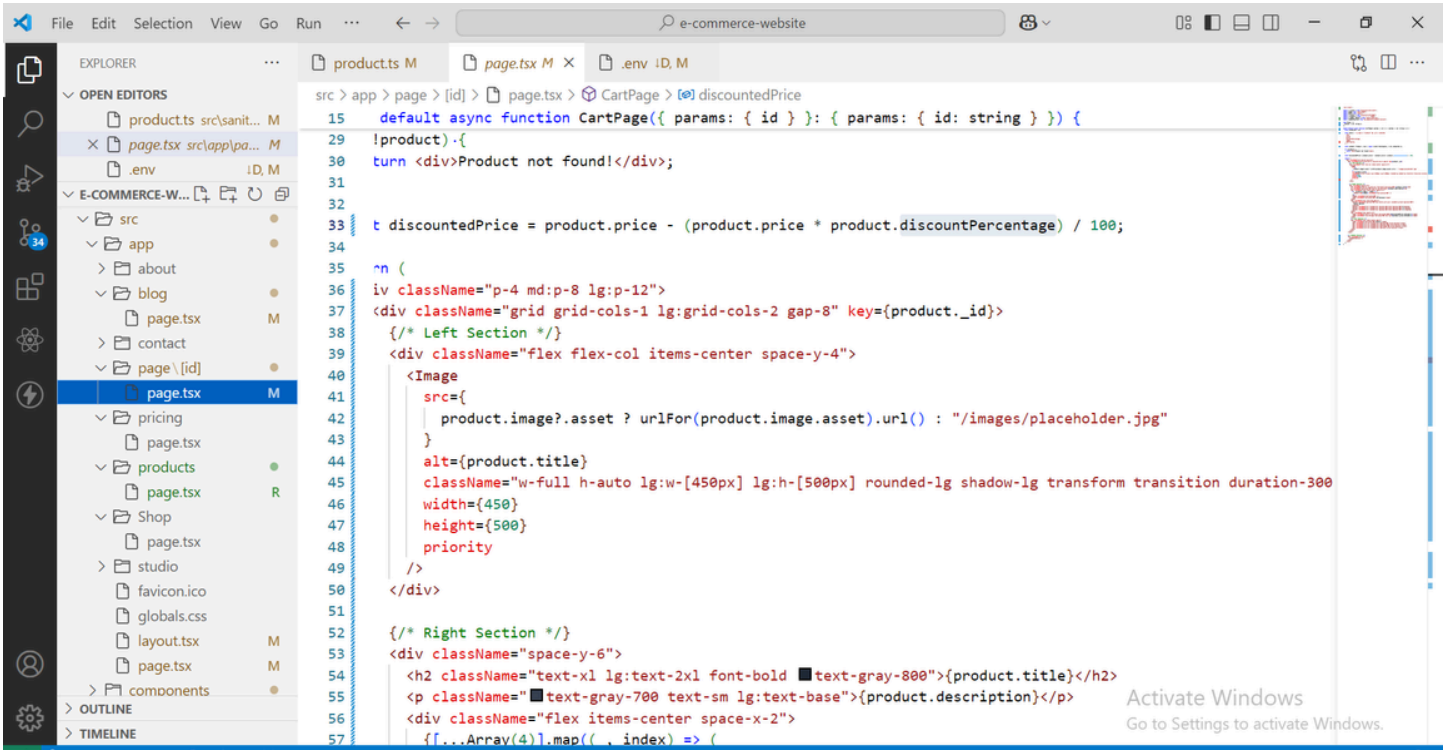
Gained hands-on experience with connecting a CMS to a modern frontend framework.

Writing GROQ queries for efficient data fetching.

Implementing responsive designs with Tailwind CSS.

9. 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.


File Edit Selection View Go Run
e-commerce-website
EXPLORER
product.ts M
page.tsx M
.env ↓D, M
OPEN EDITORS
product.ts src\sanit... M
page.tsx src\app\pa... M
.env ↓D, M
E-COMMERCE-W...
src
app
about
blog
page.tsx M
contact
page\[id]
page.tsx M
pricing
page.tsx
products
page.tsx R
Shop
page.tsx
studio
favicon.ico
globals.css
layout.tsx M
page.tsx M
components
OUTLINE
TIMELINE
src > app > page > [id] > page.tsx > CartPage > discountedPrice
default async function CartPage({ params: { id } }: { params: { id: string } }) {
!product)·{
turn Product not found! ;
t discountedPrice = product.price - (product.price * product.discountPercentage) / 100;
n (
iv className="p-4 md:p-8 lg:p-12">

{/* Left Section */}

<Image
src={
product.image?.asset ? urlFor(product.image.asset).url() : "/images/placeholder.jpg"
}
alt={product.title}
className="w-full h-auto lg:w-[450px] lg:h-[500px] rounded-lg shadow-lg transform transition duration-300
width={450}
height={500}
priority
/>

{/* Right Section */}

<h2 className="text-xl lg:text-2xl font-bold text-gray-800">{product.title}</h2>
{product.description}

{[...Array(4)].map((_, index) => (
Activate Windows
Go to Settings to activate Windows.