API Integration Report - [FURNITURE Market]

In my project, I fetched data from an API connected to Sanity using Next.js. I used the getStaticProps method to retrieve the product data at build time. After fetching the data, I passed it to the UI and displayed the product information on individual product pages. This ensured that the content was dynamic and updated in real time without reloading the page.

"Benefits of Fetching Data from Sanity API for Enhanced Flexibility and Performance"

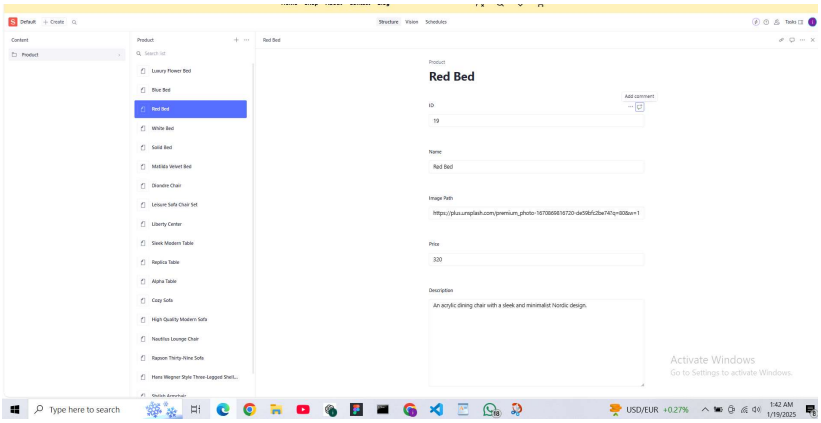
Fetching data from Sanity via an API has several advantages. Sanity is a headless CMS that allows you to easily manage your content. When you fetch data from Sanity using an API, you can update the content without changing the code. Fetching data from Sanity improves your website's scalability and flexibility, and allows you to update content quickly. This method also enhances the performance of your site, especially if you have large data. Thanks to structured data, displaying it on the front-end becomes easier, and with real-time syncing, you always show fresh and updated content.

Setting Up Sanity: First, I created a project on Sanity and set up the schema for the data I wanted to display. The schema included fields like title, description,name,price and image for the content that would be fetched.

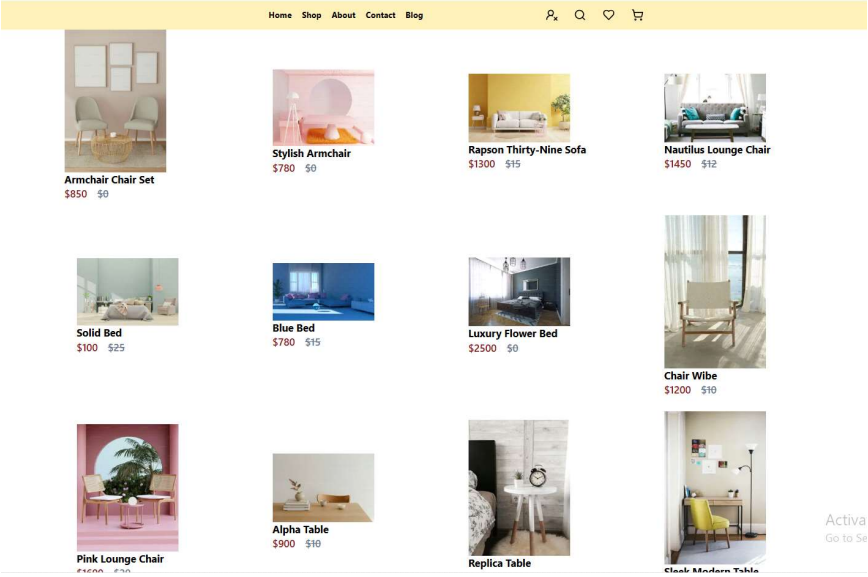
Fetching Data from API: I used Next.js API routes to create an endpoint that fetches data from an external API. This was done using fetch() or axios to get the data asynchronously. The response was then formatted and stored in a state variable in my component.

Access to the API: https://template-0-beta.vercel.app/api/product

Connecting with Sanity: To connect to Sanity, I used the @sanity/client package. I set up the client in my Next.js project by adding the project ID and dataset name. This allowed me to retrieve and manage content directly from Sanity.



Displaying Data: After retrieving the data from both the API and Sanity, I displayed it on the frontend. For the API data, I mapped over the fetched array and rendered each item in a list. Similarly, I displayed Sanity data by querying the Sanity database and rendering it in the UI.



In my project, I implemented dynamic product pages where each product's details are displayed on a unique page. This approach enhances the user experience by presenting relevant information for each product individually.