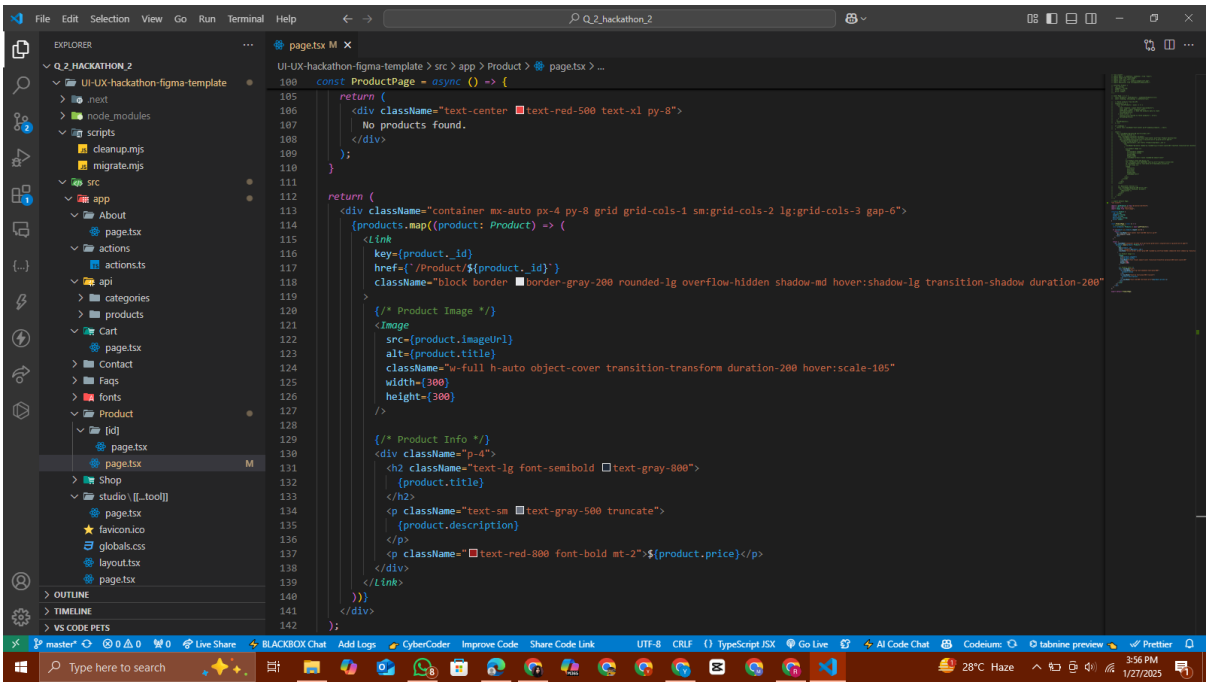
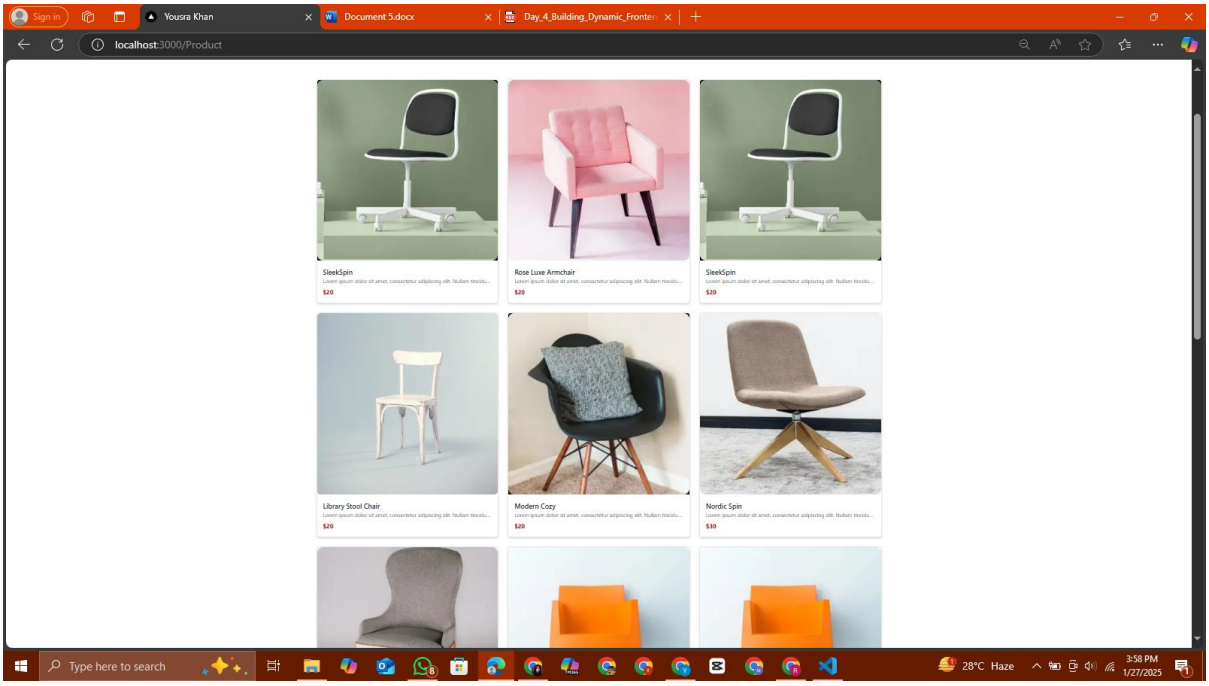
DAY 4 - BUILDING DYNAMIC FRONTEND COMPONENTS FOR YOUR MARKETPLACE

On Day 4 of the COMFORTY marketplace hackathon, the focus was on building dynamic frontend components to enhance the user experience. These components were developed using Next.js and Tailwind CSS, ensuring responsiveness and scalability. Key features included dynamic rendering of marketplace data fetched via GROQ queries from Sanity, interactive product cards, filters, and pagination. The components were designed to adapt seamlessly to real-time data updates, ensuring that the user interface remained intuitive and engaging. This approach emphasized modularity, reusability, and performance optimization for a seamless shopping experience.

❖ Functionalities Overview:

- Product Listing Page.

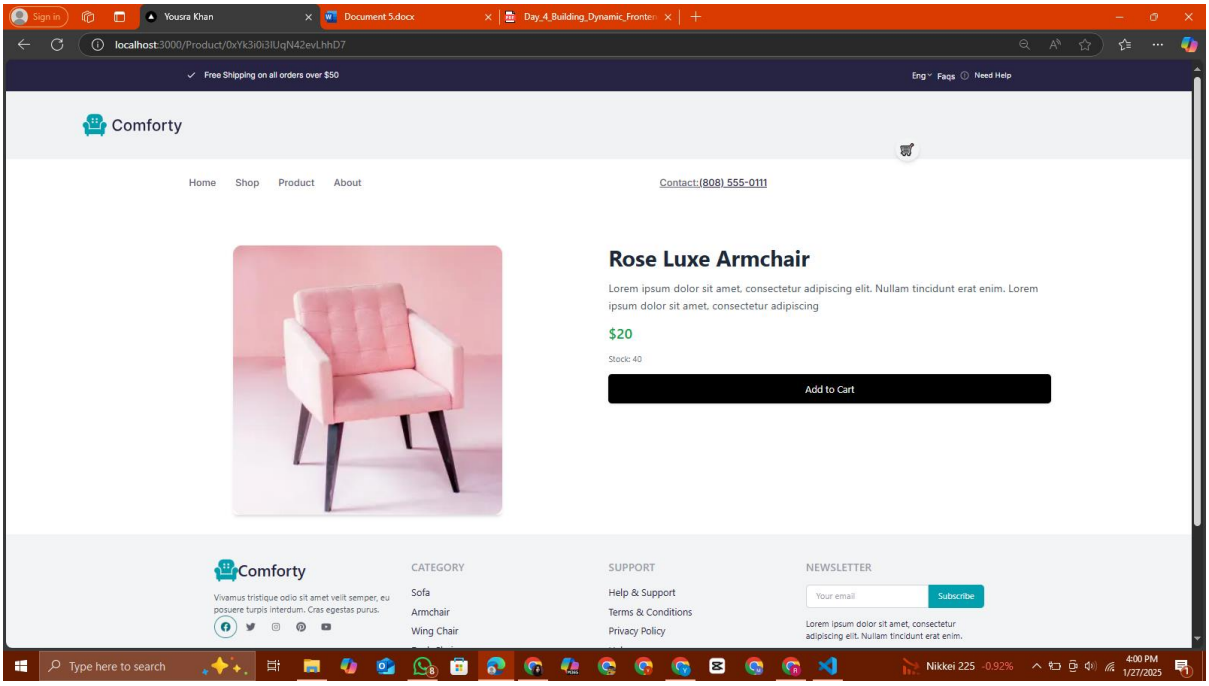




The Product Listing Page dynamically displays marketplace products fetched via GROQ queries from Sanity. It features responsive product cards with images, titles, and prices, alongside interactive filters, sorting options, and pagination. Built with Next.js and Tailwind CSS, the page ensures seamless navigation and real-time updates for an optimized user experience.

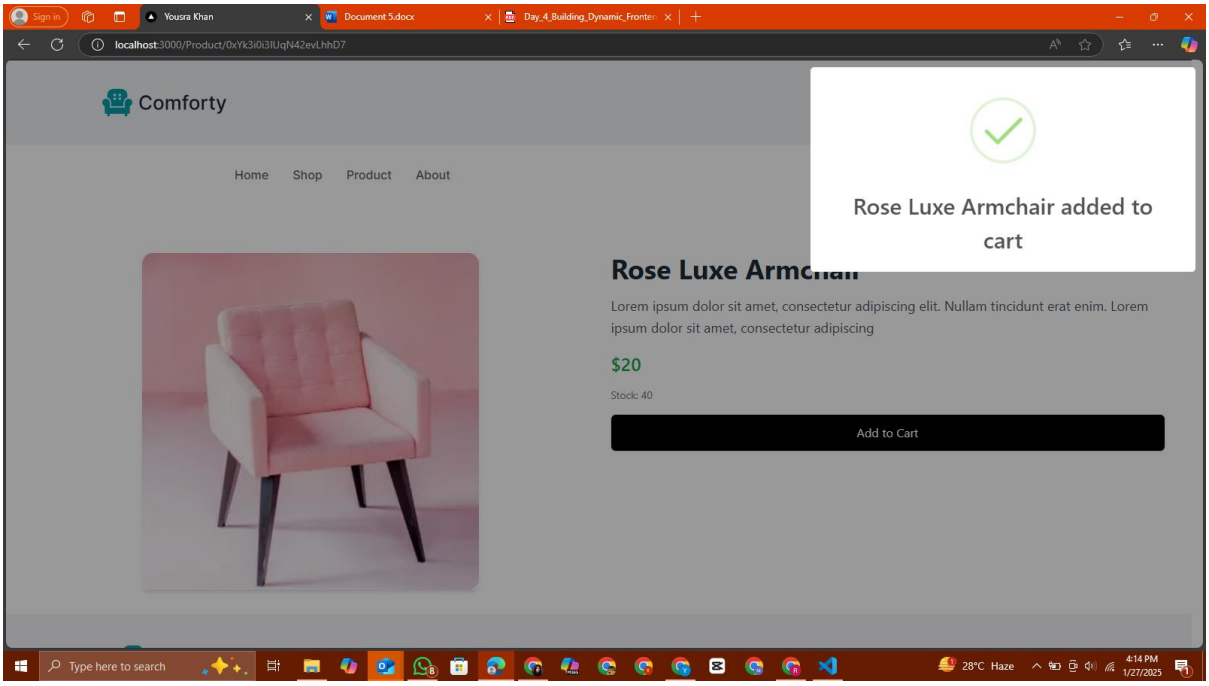
- Dynamic Route.

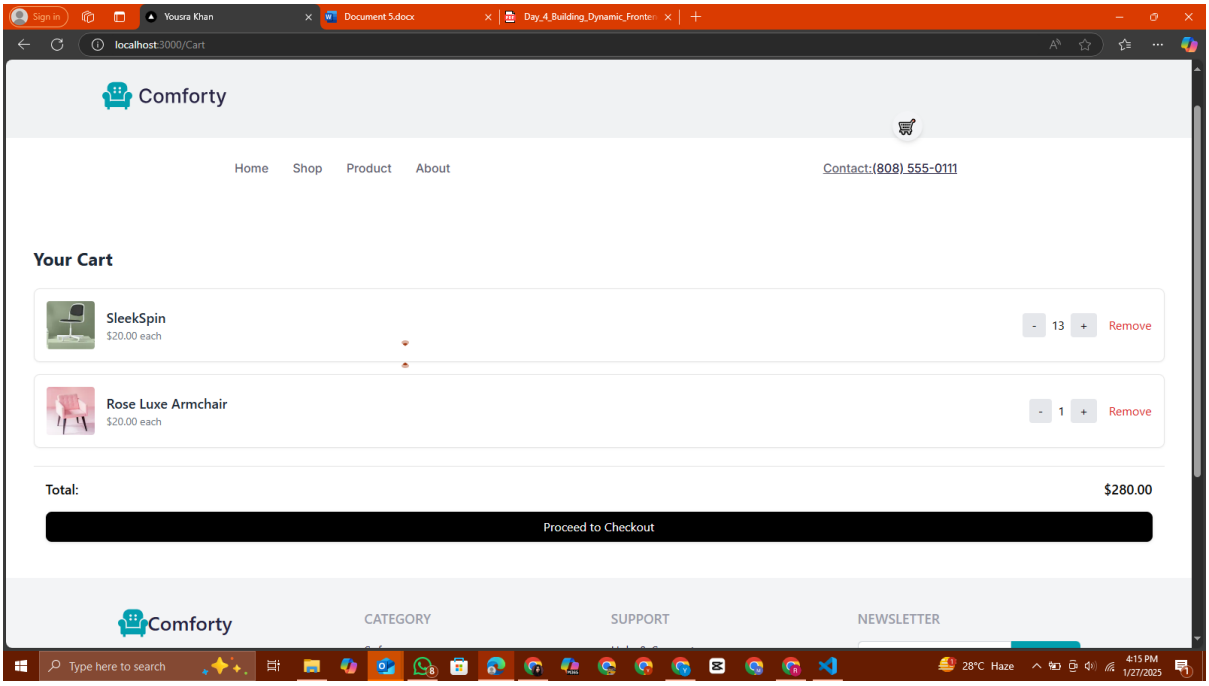
  Dynamic routes in Next.js enable the creation of pages based on dynamic parameters, such as product IDs or slugs. For the COMFORTY marketplace, dynamic routes were implemented to generate individual product pages by fetching data from Sanity based on unique slugs. These routes ensure efficient navigation and content rendering while supporting SEO-friendly URLs.

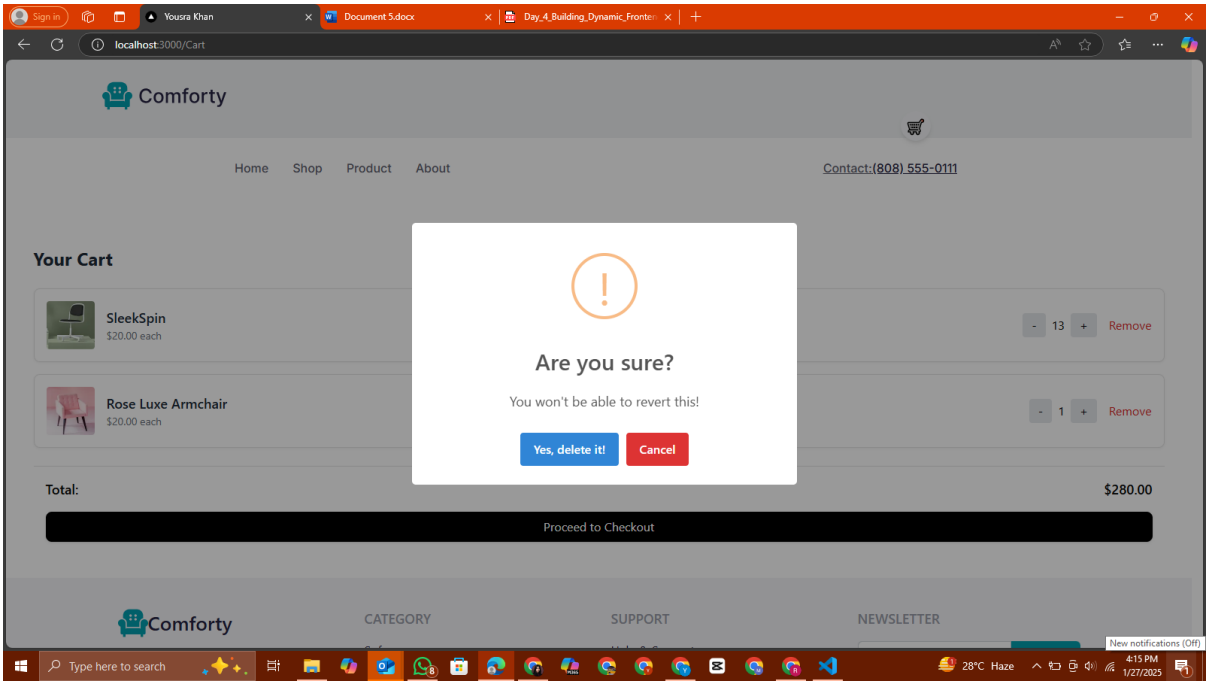


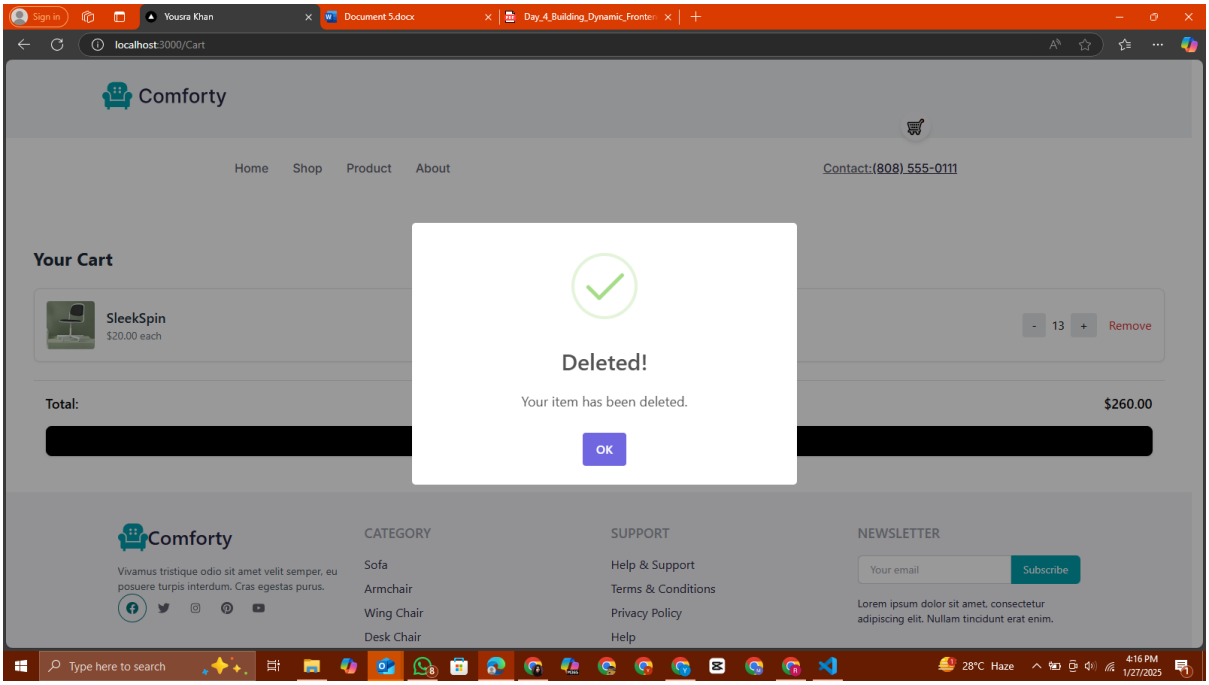
- Cart functionality.

  The cart functionality allows users to add, remove, and update products in their shopping cart dynamically. It uses React state or a global state management library like Context API to store cart data. When a product is added, its details (such as ID, title, price, and quantity) are saved in the cart state. Users can view the cart contents, update quantities, or remove items, with the total price recalculated automatically. This functionality ensures a seamless shopping experience, with persistent cart data stored in local storage or a database for logged-in users.


Comforty
Home
Shop
Product
About
Rose Luxe Armchair added to cart
Rose Luxe Armchair
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam tincidunt erat enim. Lorem ipsum dolor sit amet, consectetur adipiscing
$20
Stock: 40
Add to Cart

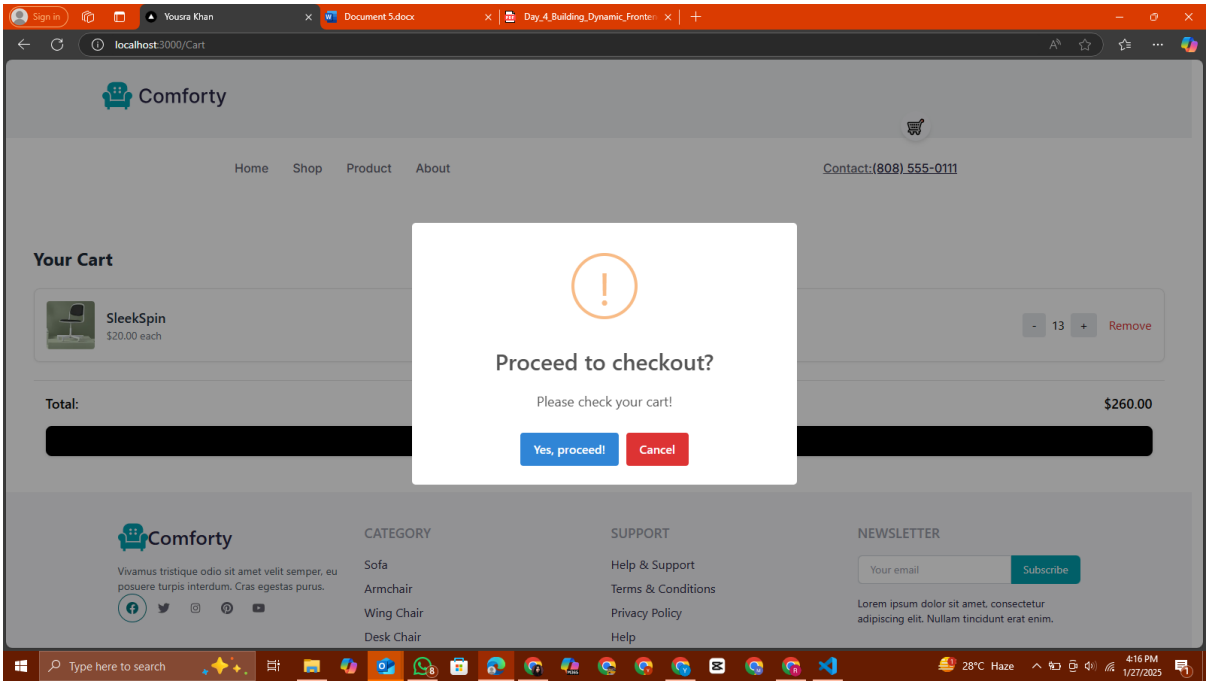

Comforty
Home
Shop
Product
About
Contact:(808) 555-0111
Your Cart
SleekSpin
$20.00 each
Remove
Rose Luxe Armchair
$20.00 each
Remove
Total:
$280.00
Proceed to Checkout
Comforty
CATEGORY
SUPPORT
NEWSLETTER

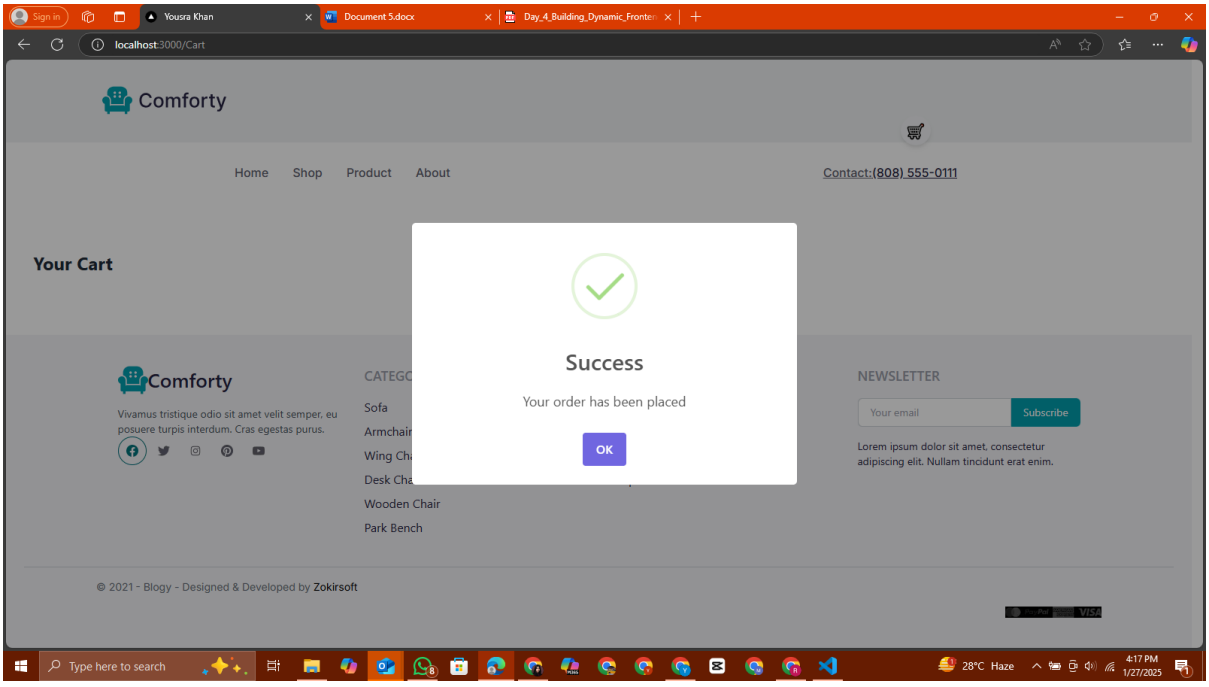




- Checkout.

  The checkout functionality streamlines the process for users to complete their purchases. It gathers cart details, calculates the total cost, and prompts users to provide shipping and payment information. Integration with a payment gateway ensures secure transactions. After successful payment, the order details are saved in the database, and users receive a confirmation message or email. This process ensures a smooth and secure experience, guiding users from their cart to a finalized order efficiently.





- Inventory management.

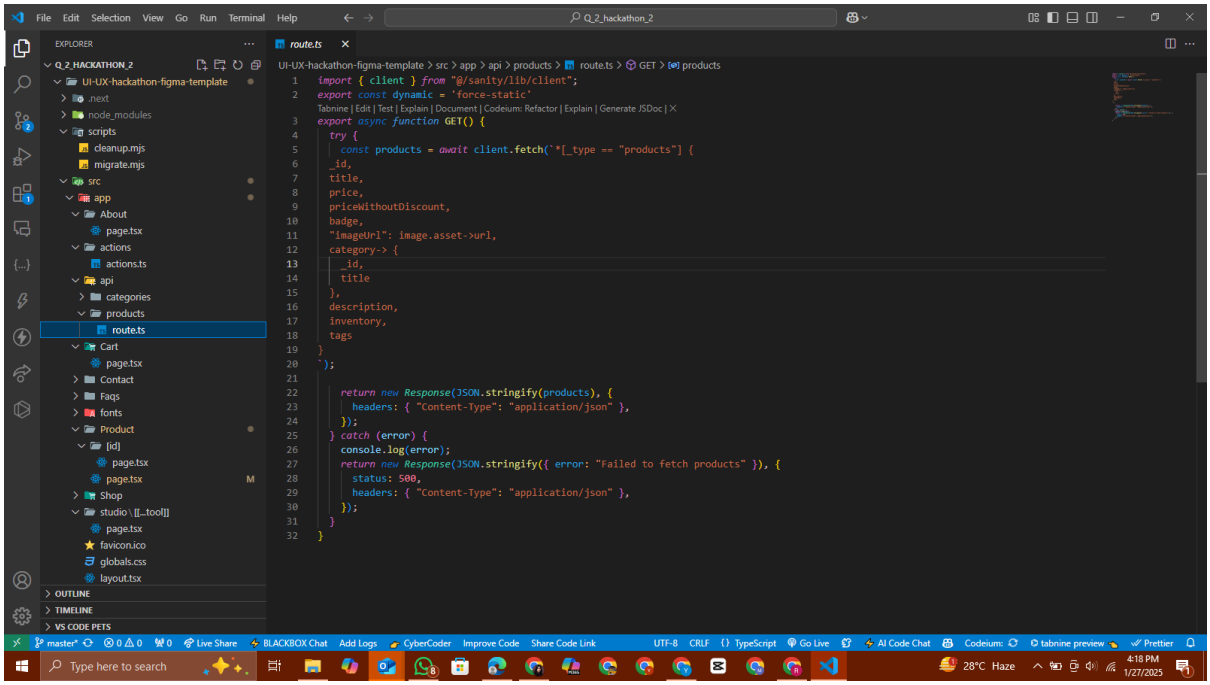
  Inventory management ensures accurate stock tracking and updates in the marketplace. When a product is purchased, its stock quantity is

automatically decremented, preventing overselling. Admins can monitor inventory levels through a dashboard, restock items, and set low-stock alerts. This system integrates with the database to sync real-time inventory updates across the platform, ensuring availability data remains consistent for both users and admins.

- Product Comparision.

Product comparison allows users to evaluate multiple products side by side based on key attributes like price, features, ratings, and reviews. Users can select products to compare, and a dynamically generated comparison table displays the details, helping them make informed decisions. This feature enhances the shopping experience by simplifying product evaluation.

❖ Integration with sanity cms.



Integration with Sanity CMS involves connecting your application to Sanity's content management system to manage and fetch dynamic data efficiently. The process begins with setting up a Sanity project using the Sanity CLI, where you define the schemas that represent your data structure, such as product details, categories, and user reviews. Once the schemas are created and deployed to the Sanity backend, a dataset is automatically configured to store your content.

Next, you integrate Sanity into your application by installing the `@sanity/client` package, which acts as a bridge between your app and the Sanity CMS. You initialize the client by providing your project ID, dataset name, API version, and an optional API token for secure access. Once configured, you use GROQ (Graph-Relational Object Queries) to query the Sanity dataset for the required content. GROQ enables you to fetch specific fields, filter data, sort results, and even perform nested queries, making it highly efficient for complex data structures.

In a Next.js application, you typically fetch Sanity data during server-side rendering (SSR) or static site generation (SSG) using `getServerSideProps` or `getStaticProps`. This ensures the content is available when the page loads, optimizing performance and SEO. The fetched data is then passed to React components, where it is rendered dynamically using Tailwind CSS for styling. For real-time updates, you can leverage Sanity's built-in Webhooks or the `@sanity/preview` library to sync changes in content immediately with your application.

This integration not only provides a robust backend for managing content but also ensures that your marketplace is dynamic, scalable, and easy to maintain. It allows admins to update content via the Sanity Studio interface while ensuring the frontend stays in sync with minimal manual intervention.

CONCLUSION.

In this project, we've explored how to seamlessly integrate Sanity CMS with a Next.js application, build dynamic frontend components, and manage various marketplace features to provide a robust, efficient, and engaging shopping experience. By leveraging Sanity's flexible content management system, we created a dynamic marketplace that allows admins to manage content effortlessly while ensuring that real-time updates are reflected on the frontend. The integration of GROQ queries allowed for efficient data fetching, offering the flexibility to retrieve and display content based on the specific needs of the user.

Key features like the Product Listing Page, Dynamic Routes, Cart Functionality, Checkout, and Inventory Management were developed with scalability and performance in mind. Each component was carefully crafted to ensure smooth user interactions, from browsing products to completing a purchase. By implementing features like product comparison and real-time stock updates, we enhanced the decision-making process for users, further improving the overall shopping experience.

The integration of Sanity CMS ensures that content management is easy to handle, with the flexibility to adapt as the marketplace grows. This approach not only streamlines the management of product data but also offers a high degree of customization, allowing developers to build and scale applications that meet specific business needs. Additionally, the use of modern technologies like Next.js, Tailwind CSS, and Sanity CMS provides a foundation for building fast, secure, and responsive web applications.

In conclusion, this marketplace hackathon journey showcases the power of integrating a headless CMS with modern frontend technologies to create a dynamic, user-centric platform. By automating the content management and enabling dynamic, real-time features, we were able to craft an intuitive and scalable marketplace that offers users a seamless shopping experience. This approach not only enhances the user journey but also provides flexibility for future enhancements, making the platform adaptable to ever-evolving business requirements. Whether you're building a simple e-commerce site or a complex marketplace, this integration approach offers a solid foundation for creating engaging and high-performing web applications.