Day 4 - Building Dynamic Frontend Components for Your Marketplace

Technical Report on Building Dynamic Frontend Components:

This report outlines the process of building dynamic frontend components using **Next.js, Sanity CMS, and API data**, covering the steps taken, challenges encountered, and best practices followed.

Steps Taken to Build and Integrate Components:

1. **Project Setup:** Initialized a Next.js project and configured it for dynamic rendering.
2. **Sanity CMS Integration:** Connected Sanity as the headless CMS to manage structured content.
3. **API Fetching:** Used Next.js **getStaticProps** and **getServerSideProps** to fetch and pre-render data efficiently.
4. **Component Development:** Built reusable, dynamic React components that consume API data and CMS content.
5. **State Management:** Implemented **React Context API** or other state management tools to handle dynamic interactions.
6. **Optimization:** Applied caching, lazy loading, and SSR/ISR strategies to improve performance.

Challenges Faced and Solutions Implemented:

- **Data Fetching Complexity:**
    - Solution: Used **Next.js API routes** to standardize data fetching and reduce redundant requests.
- **CMS Data Synchronization:**
    - Solution: Implemented **webhooks in Sanity** to trigger revalidation and keep the frontend updated.
- **Performance Bottlenecks:**

- Solution: Optimized images using **Next.js Image component** and minimized API calls using caching strategies.

Best Practices Followed:

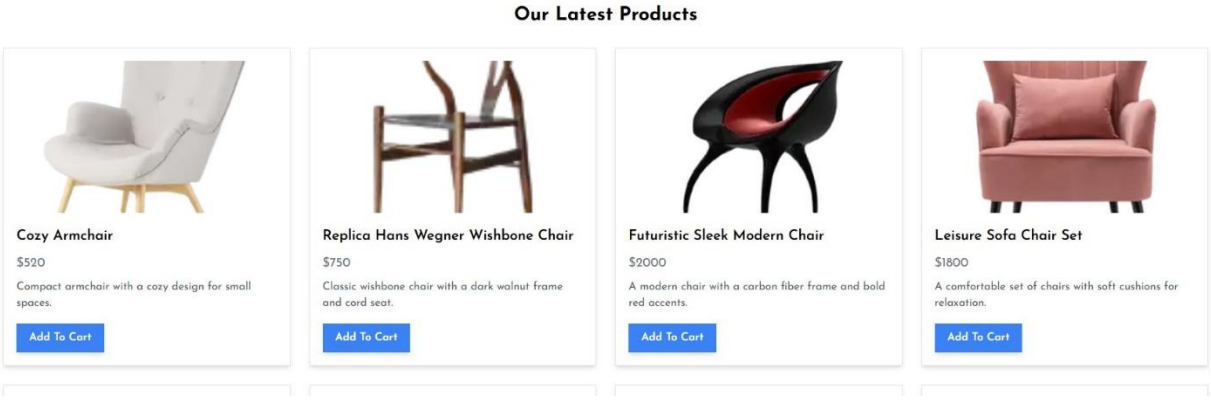
- **Modular Components:** Developed reusable UI components to maintain scalability.
- **Optimized Data Fetching:** Used ISR and SSR strategically to balance performance and freshness.
- **SEO & Accessibility:** Ensured pages are SEO-friendly using Next.js **Head** and followed accessibility guidelines.
- **Code Maintainability:** Followed **TypeScript** and clean code principles to enhance readability and debugging.

Tools & Technologies Used:

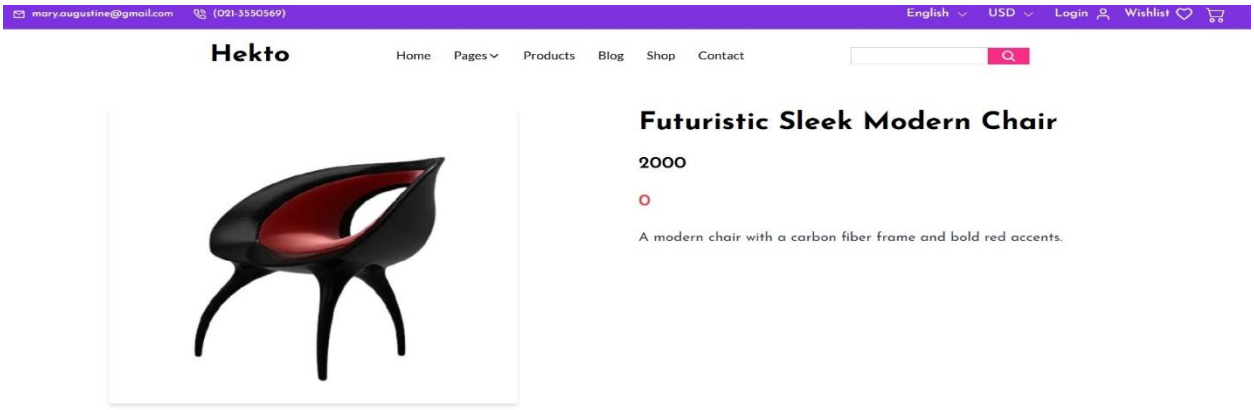
Frontend: Next.js / Tailwind CSS
Backend: Sanity CMS
Statement Management: API

Screenshots:

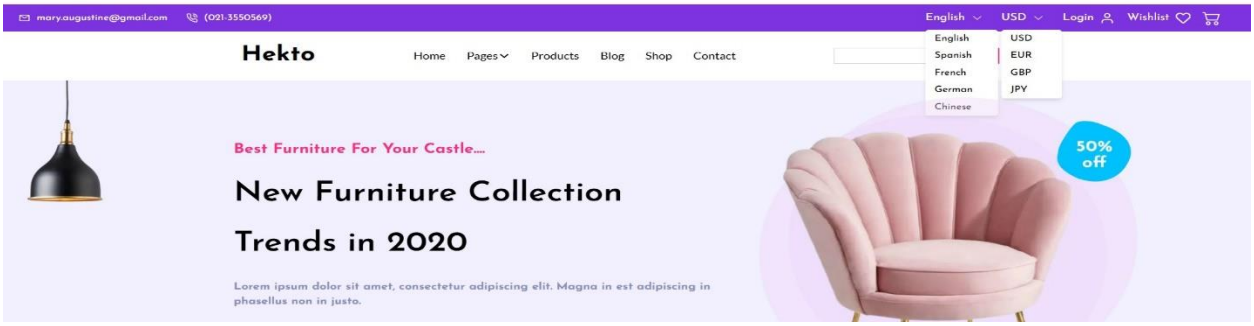
1. Add-to-Cart & Notifications: Created cart functionality on fetched data with toast notifications to enhance the shopping experience.



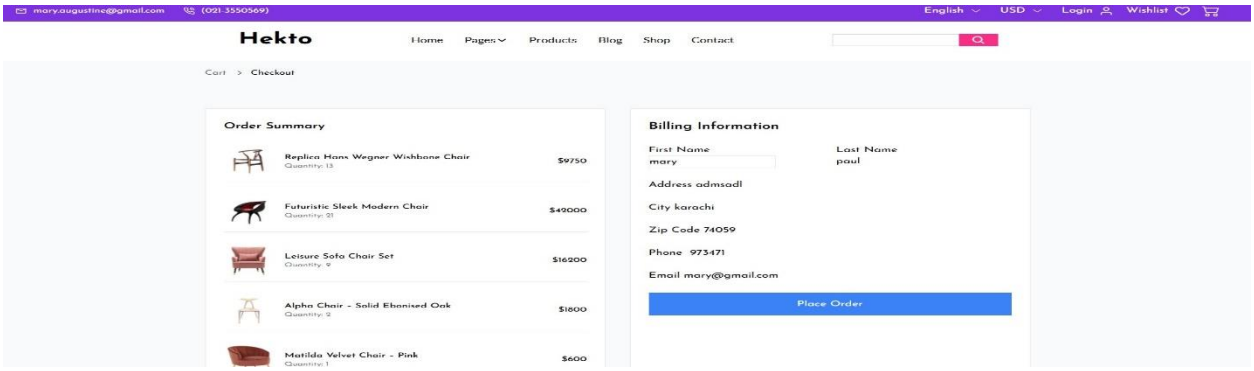
2.Dynamic Product Details: Implemented dynamic routing on fetched data for detailed product pages.



3.Search Bar: Added different languages option to filter desire products.



4.Real-Time Stock Management: (Checkout) Integrated Sanity CMS to handle real-time stock updates, ensuring inventory changes dynamically when products are added to the cart.



Conclusion:

By integrating **Next.js, Sanity CMS, and API data**, we successfully built a dynamic, scalable, and high-performance frontend. The project demonstrated the importance of **modular architecture, optimized data fetching, and performance-driven development**. Addressing challenges through strategic solutions ensured seamless content synchronization and a smooth user experience. Following best practices in component design and API management helped create a maintainable and future-proof system.

Day-4 Checklist

Self-Validation Check List				
Frontend Component Development	Styling and Responsiveness	Code Quality	Documentation & Submission	Final Review
✓	✓	✓	✓	✓