Day 4 Documentation

Project Title: Building Dynamic Frontend Components for Marketplace

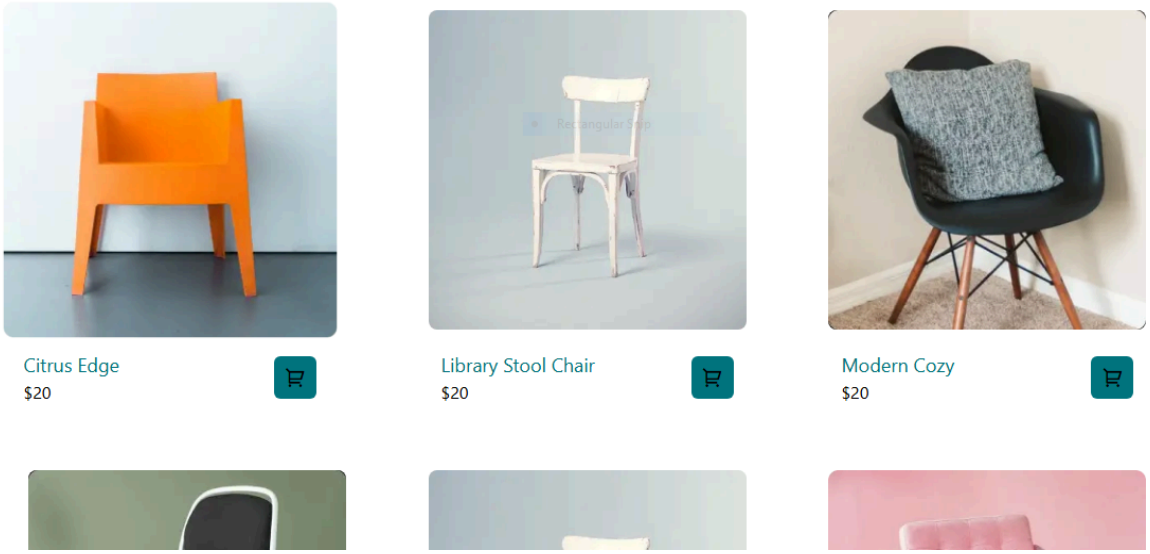
Objective

To dynamically render product listings and product detail pages by integrating data from Sanity CMS, implementing dynamic routing, adding cart and wishlist functionalities, and securing the checkout process with Clerk authentication.

Tasks Completed

1. Fetching Data from Sanity CMS for Product Listing

- Replaced static product data with dynamic data fetched from Sanity CMS.
- Used `client.fetch` method to retrieve products including title, price, description, and image.
- **Result:** All products are dynamically rendered on the product listing page.



2. Created a Reusable ProductCard Component

- Built a `ProductCard` component to display individual product data.
- Used props to pass product details (image, title, price).
- **Result:** Component is reused in the product listing page, dynamically rendering products.

3. Implemented Dynamic Routing for Product Detail Pages

- Created a dynamic route (`app/Products/[id]/page.tsx`) to handle individual product pages.
- Used `params.id` to fetch data for a specific product.
- **Result:** Clicking a product card navigates to a detail page with complete product information.

4. Solved Image Rendering Issues

- Fixed errors related to empty `src` attributes in the `Image` component.
- Used Sanity's image URL builder to convert image objects to URLs.
- **Result:** Images now correctly display.

5. Added Cart Functionality using Context API

- Implemented a `CartContext` to manage cart state across the application.
- Users can add products to the cart from both the product listing and detail pages.
- **Result:** Items are stored in the cart with their quantity and price.

6. Added Wishlist Functionality using Context API

- Implemented a `WishlistContext` to allow users to save favorite products.
- Users can add/remove products from their wishlist.
- **Result:** A dedicated wishlist section where users can manage their saved items.

7. Created Checkout Page with Sanity Integration

- Built a checkout page where users enter personal details and confirm their order.
- Submitted order data, including user information and cart items, to Sanity CMS.
- **Result:** Orders are successfully stored in Sanity, allowing for further order management.

8. Added Order Schema in Sanity CMS

- Created a schema to store order details, including user data and purchased items.
- Ensured structured storage of order history.
- **Result:** Orders are properly recorded for tracking.

9. Integrated Clerk Authentication

- Implemented Clerk for user authentication.
- Users must be logged in to access the checkout page.
- **Result:** Ensured secure transactions by restricting checkout access to logged-in users.

Challenges Faced

- **404 Page Not Found Error:** Fixed by ensuring the correct folder structure and URL matching.
- **Empty Image Source Error:** Resolved using conditional rendering and Sanity's image builder.

Technologies Used

- **Next.js** for frontend framework and dynamic routing.

- **Sanity CMS** for content management and data fetching.
- **Tailwind CSS** for responsive design and styling.
- **Context API** for state management (cart & wishlist).
- **Clerk** for authentication.

Conclusion

This task helped in understanding advanced state management, dynamic data fetching, and authentication integration. Implementing cart and wishlist functionalities using Context API improved state handling, while Clerk authentication added a security layer to the checkout process.