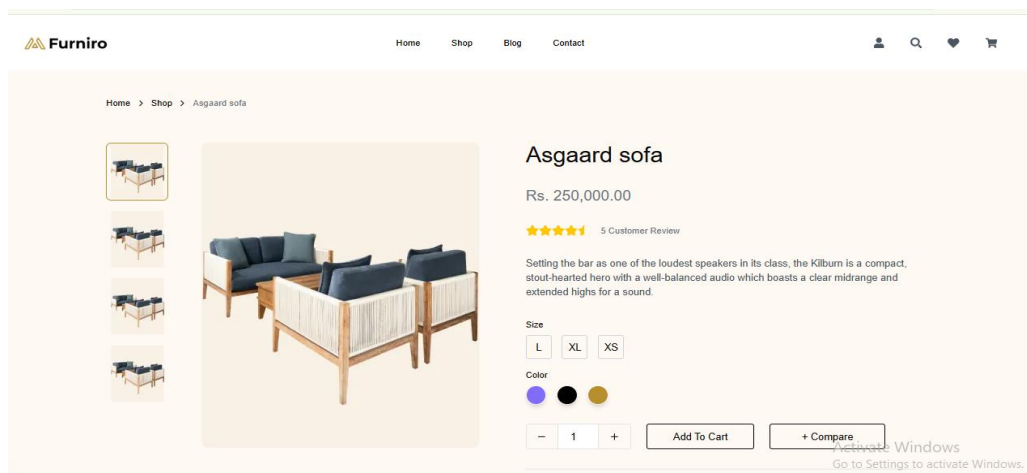


Day 4 - Dynamic Frontend Components General E-commerce Website

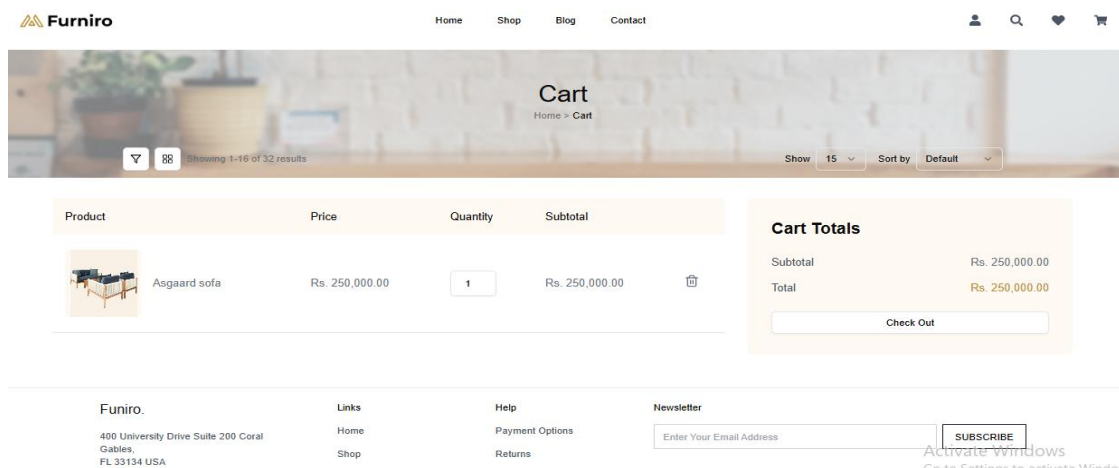
FURNIRO

1) OPERATIONAL DELIVERY

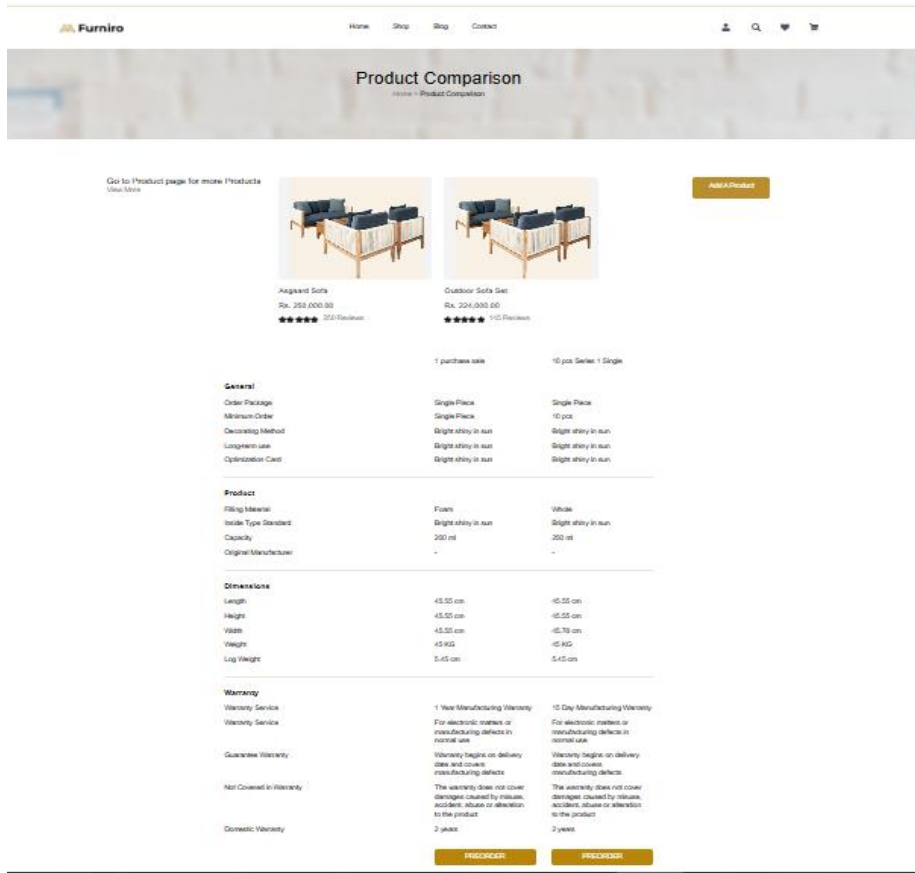
❖ Functional Product page



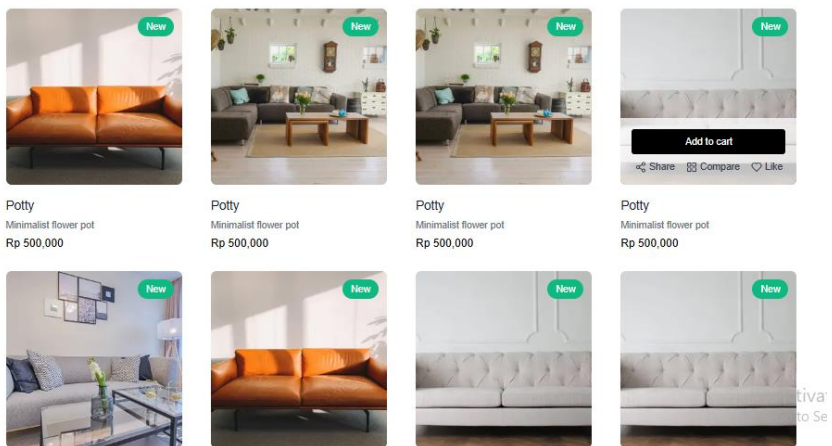
❖ Cart Page

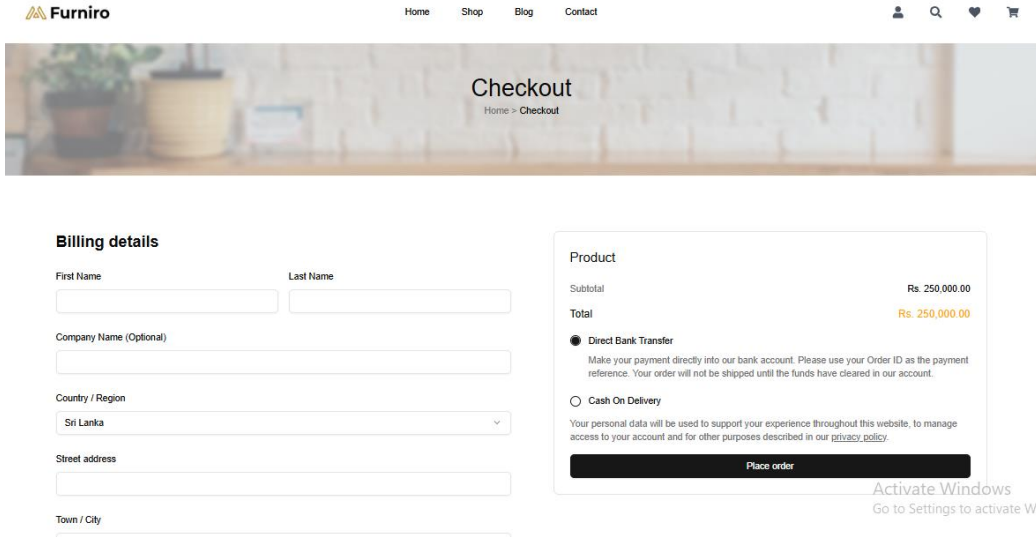


Product Comparison

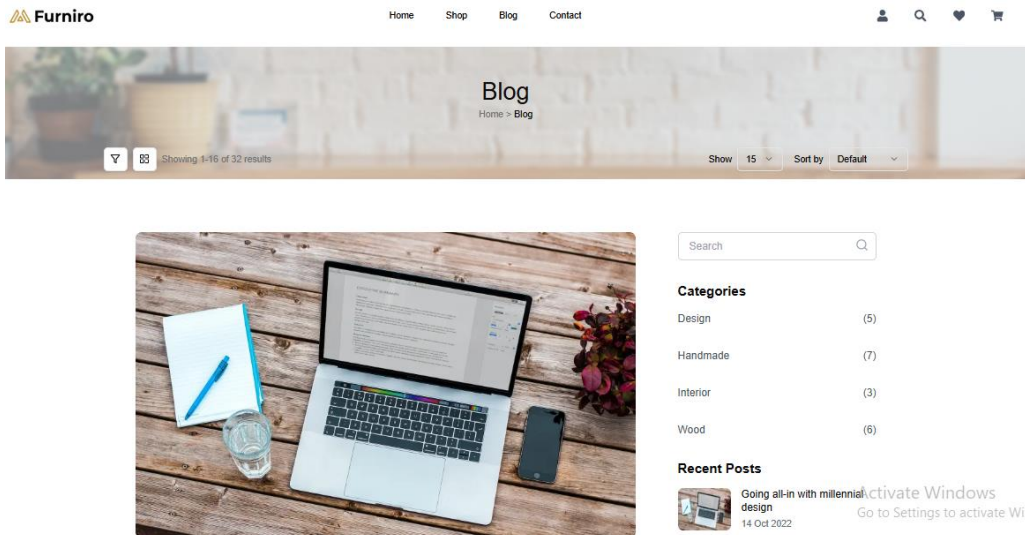


Price Filter And pagination



❖ Checkout Page with Functionality

❖ Functional Blog Page



2) Project Overview

❖ Key Components

<u>COMPONENT</u>	<u>DESCRIPTION</u>
AddToCartButton.tsx	Handles adding products to the cart using Cart.
ComparisonContext.tsx	Stores and manages product comparison data.
footer.tsx	Displays the website footer with links and relevant information
hero.tsx	The main banner or hero section for the homepage.
navbar.tsx	The navigation bar containing links and the cart icon.
ProductCard.tsx	Displays individual product details in a card format.
ProductClient.tsx	Manages product-related API calls or client-side logic
ProductList.tsx	Dynamically fetches and displays a list of products
ShopClient.tsx	Handles the shop page functionality and client-side logic

Scripts and Logic for API Integration & Dynamic Routing

- ❖ API Integration
- ❖ API Endpoint Used

Fetching Data:

- ❖ Products are fetched using fetch or axios in ProductList.

Steps Taken to Build and Integrate Components

1. Project Setup:

- o Initialized a Next.js project and configured the required dependencies.

2. Component Structure:

- o Built reusable components like ProductCard.tsx,Navbar.tsx, and Footer.tsx.

3. API Integration

- o Fetched product data from the API endpoint and dynamically displayed it.

4. UI Enhancements:

- o Designed a responsive and visually appealing UI using Tailwind CSS.

5. Testing & Optimization:

- o Debugged API calls, optimized rendering, and added error handling for better user experience.

❖ **Best Practices Followed**

Component Reusability

- Created modular, reusable components (e.g., Hero.tsx, Navbar.tsx, Footer.tsx) for maintainability

. Optimized API Calls

- Implemented efficient fetching with try-catch error handling for smooth data retrieval.

Code Maintainability

- Followed a clean project structure (components/ui/, lib/, etc.) for organized code.

Performance Optimization

- Used lazy loading and memorization to improve page load times.

Prepared By:

Anosha Hassan