Day 4

Dynamic Frontend Components– [Furniro] By M.Anees

Furniro is a top-class furniture website dedicated to providing ease and convenience for both shopkeepers and customers.
For local sellers, Furniro offers a platform to reach a broader audience by leveraging the power of websites and the internet. This helps them showcase their furniture collections online, increasing visibility and sales opportunities.

On the other hand, customers can enjoy the convenience of browsing a wide variety of furniture options from the comfort of their homes. Furniro eliminates the hassle of visiting 30 to 50 shops by offering an extensive catalog, allowing users to compare prices, styles, and quality effortlessly.

Our goal is to bridge the gap between sellers and buyers, creating a seamless shopping experience that saves time, effort, and money.

Whether you're looking for budget-friendly options or premium furniture, Furniro is here to help you make the best choices without stepping out of your home.

Challenge of the day

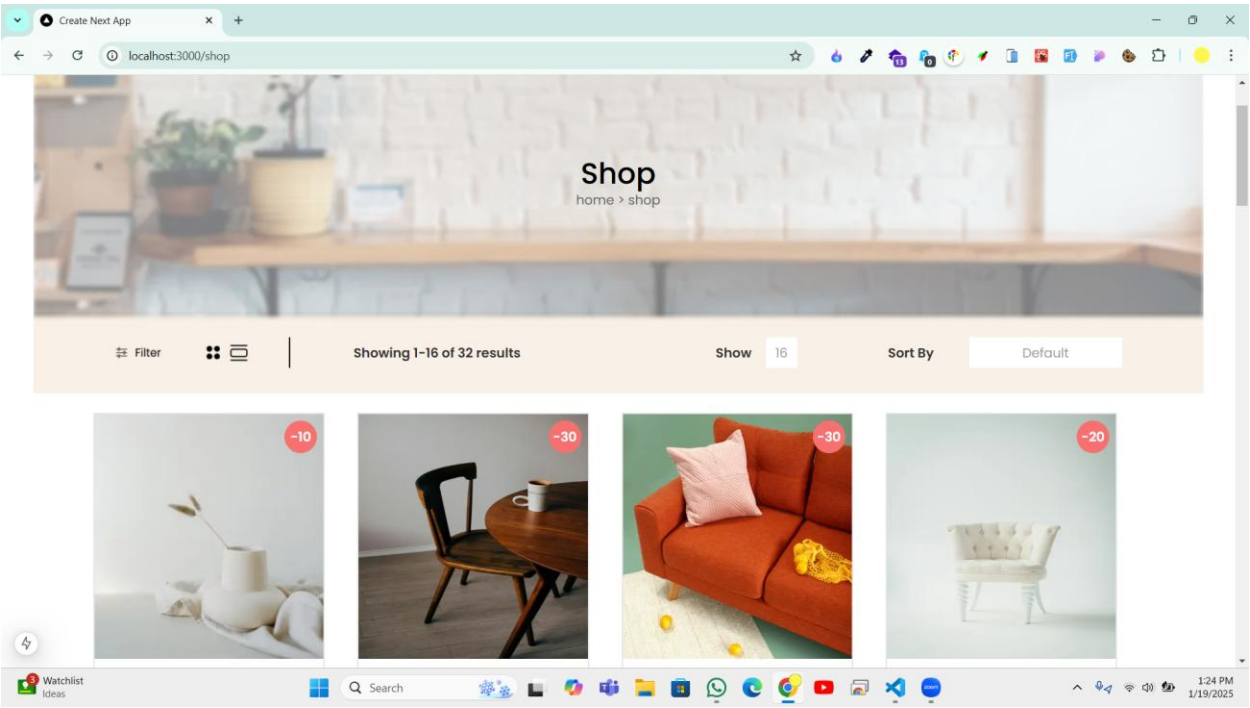
Today, I focused on developing the dynamic frontend components for my marketplace application as part of my assigned task for Day 4. The goal was to ensure the implementation of interactive and functional features that enhance the user experience. Below, I've detailed the progress and deliverables.

Main Components of the Day

The focus for Day 4 was on building and refining dynamic frontend components to enhance the marketplace application. The tasks included implementing core features such as product listing, detailed product pages, filtering options, Cart Functionality , Wishlist Sidebar and seamless navigation, ensuring a functional and user-friendly interface.

1. Product Listing Page

I began by implementing the product listing page, ensuring it dynamically fetched data from the API. Each product was displayed with accurate information, including the name, image, price, and description. The layout was designed to be responsive, providing a seamless experience across different devices.

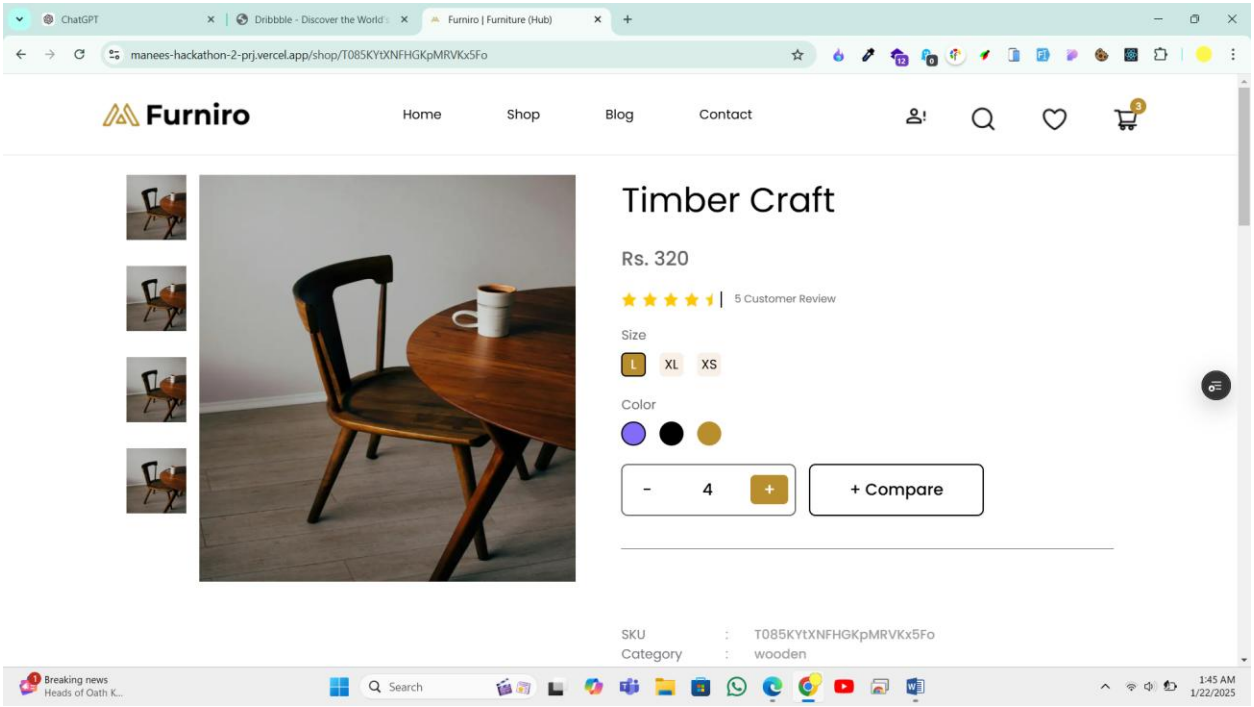


i) Designed a dynamic product listing page with data fetched via API integration.

ii)Displayed key details for each product, including name, image, price, and description, in a responsive layout.

2. Individual Product Detail Pages

I set up dynamic routing to ensure each product had its own detail page. These pages rendered product-specific data accurately, such as specifications and availability. Users can now navigate between the listing and detail pages effortlessly.

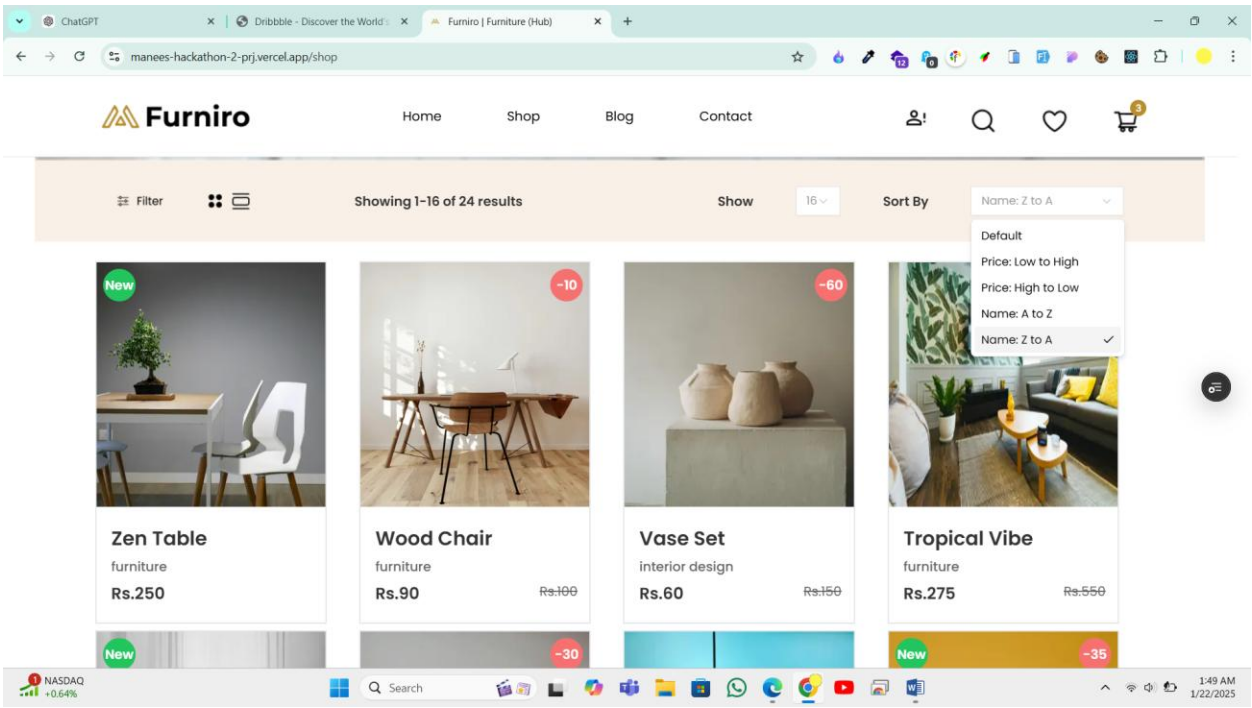


i) Implemented accurate routing to dynamically render detail pages for each product.

ii) These pages included comprehensive product information, ensuring users could easily access specific details.

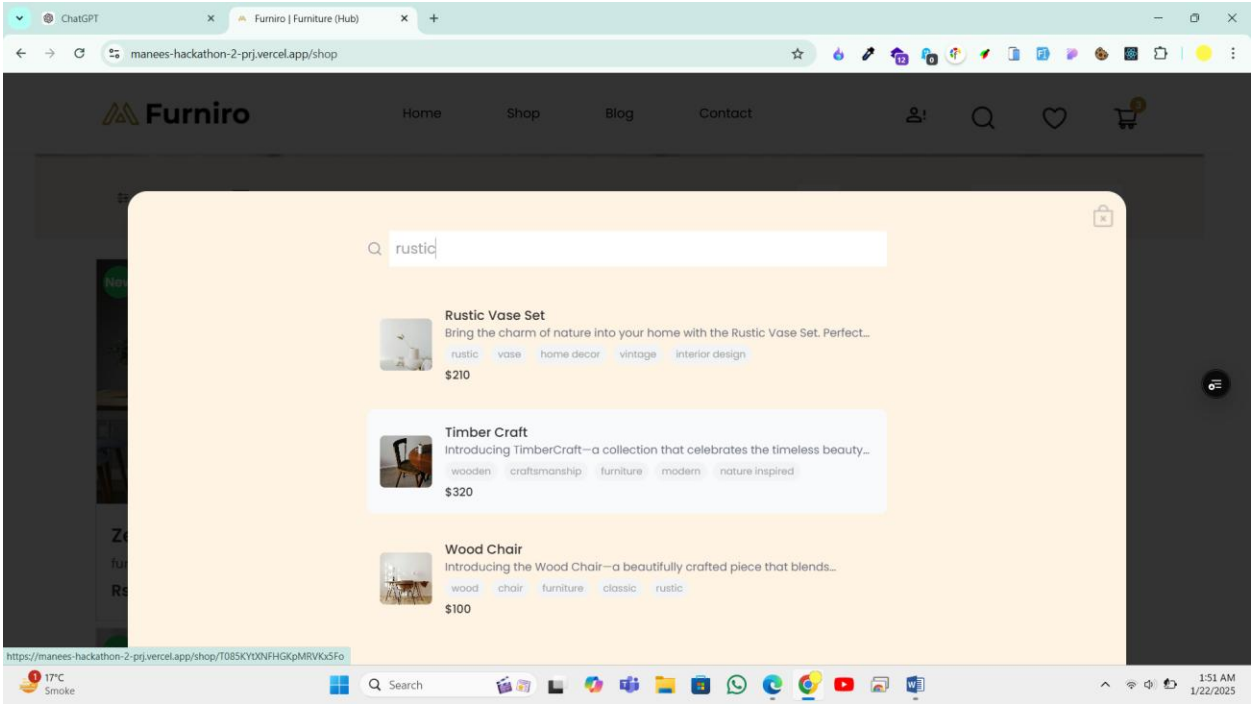
3. Category Filters

Developed a category filtering system that dynamically updated the product list based on user selections.



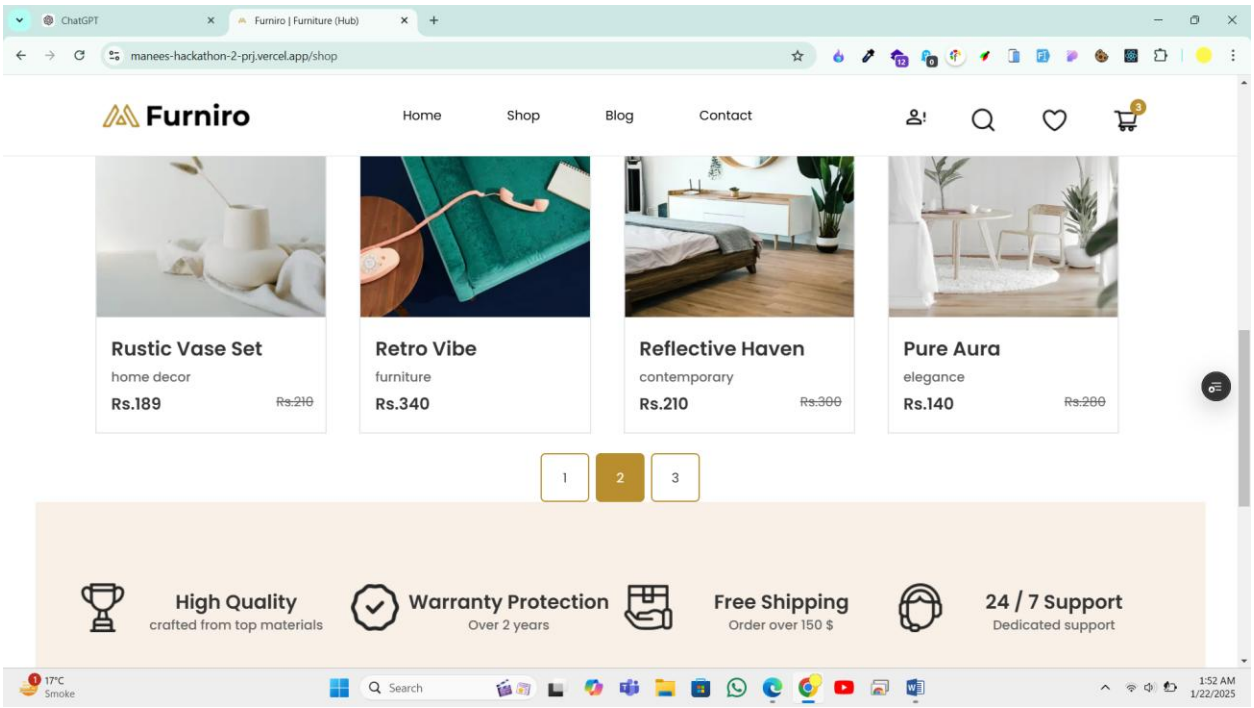
4. Search Bar

I implemented real-time search functionality, where users can type queries and see instant updates in the product list.



5. **Pagination**:

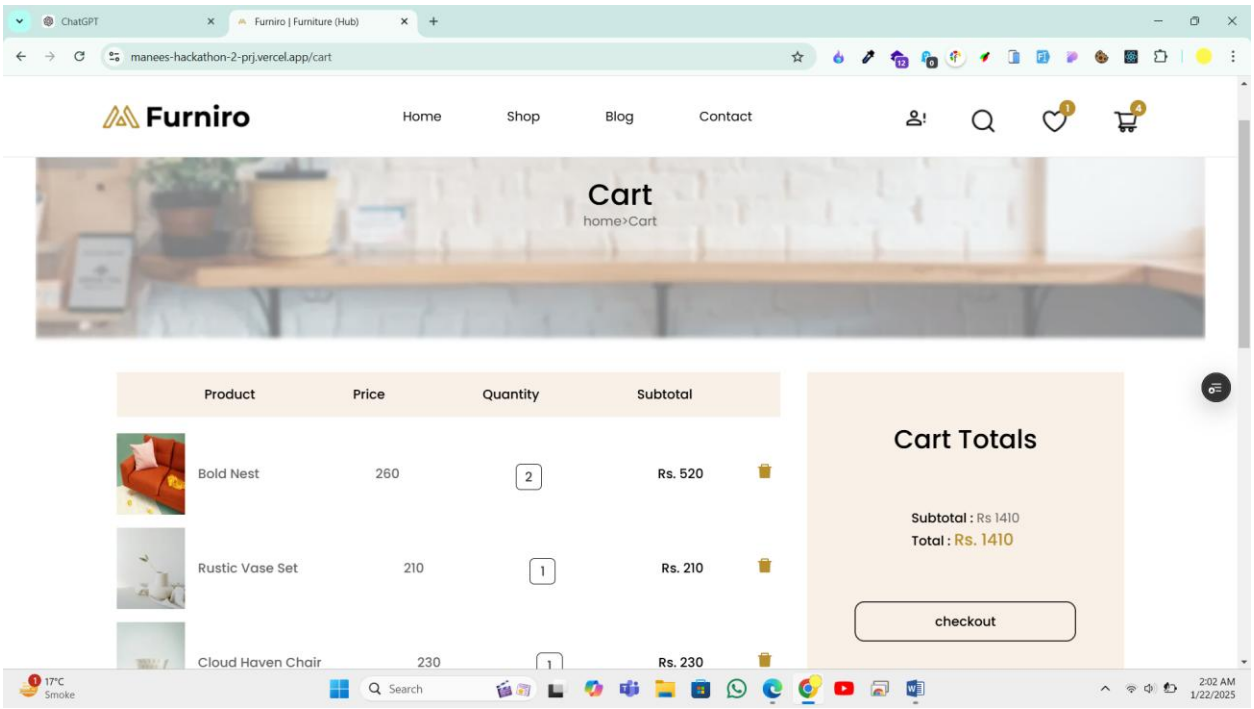
For datasets with numerous entries, pagination was introduced to allow smooth browsing without overwhelming the interface.



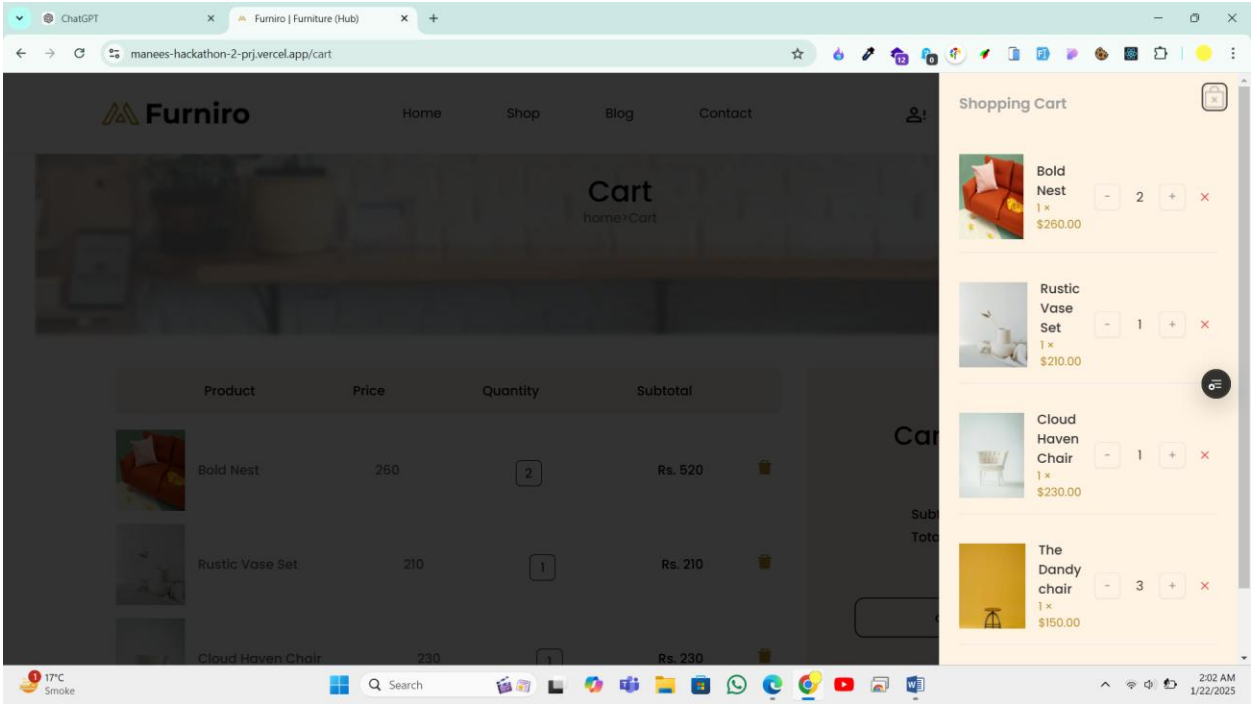
6. Add-to-Cart Route and sidebar.

1)Implemented an **Add to Cart** route and Sidebar to handle cart operations efficiently.

2)Ensured that products could be added to the cart dynamically and displayed on the cart page using Context API for state management.

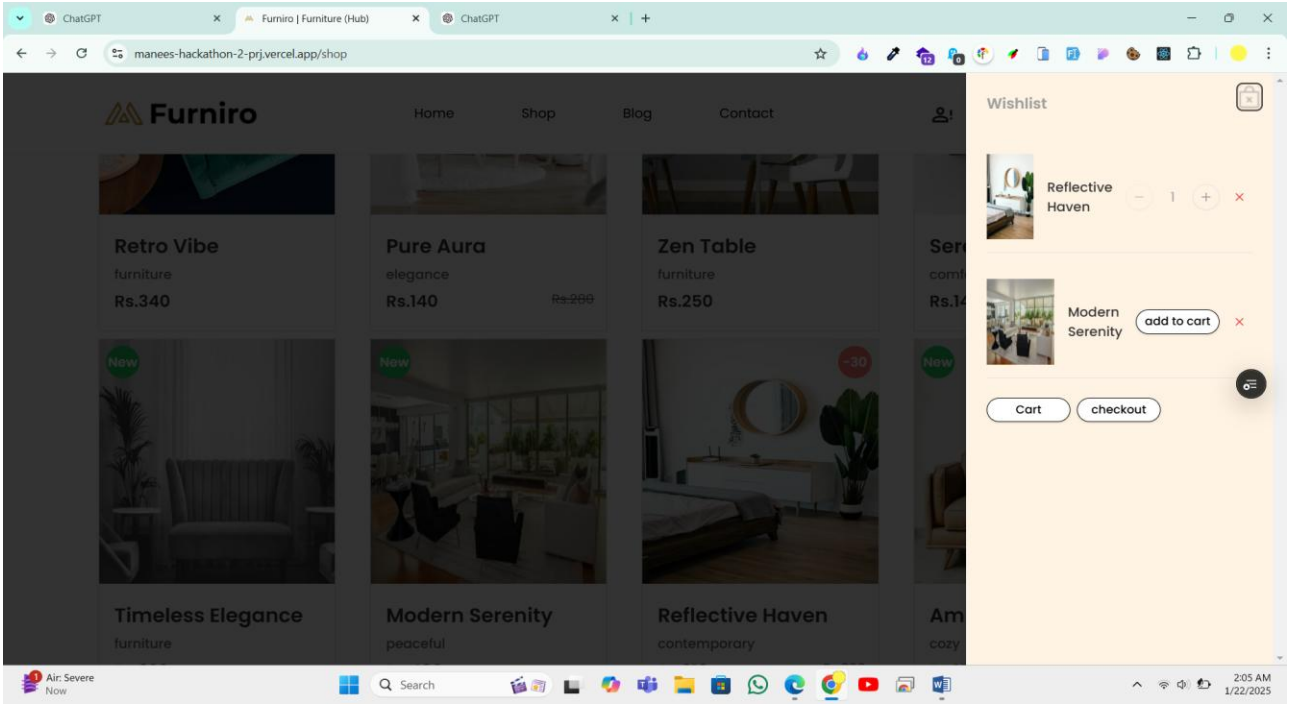


Below is the Visualization of the Cart Sidebar



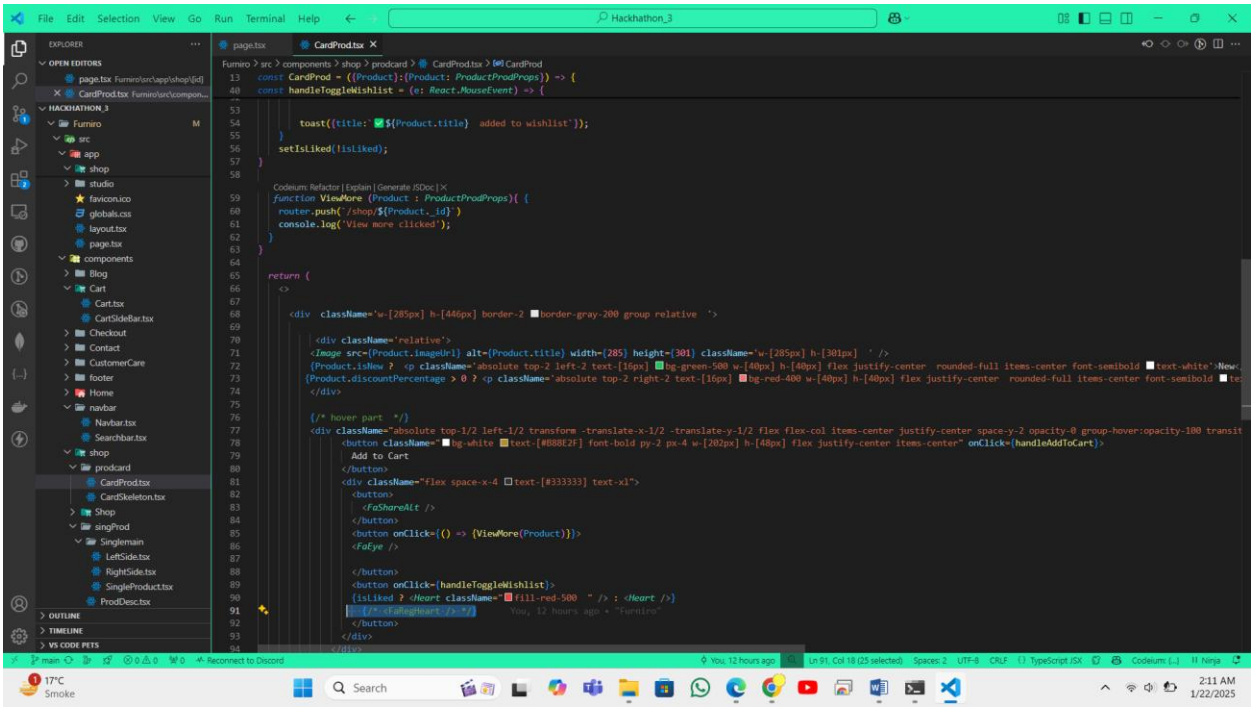
6. Wishlist Side Bar using Context api.

- Developed a wishlist feature to allow users to save favorite products for later.
- Added a sidebar component for wishlist items, dynamically updated using Context API.

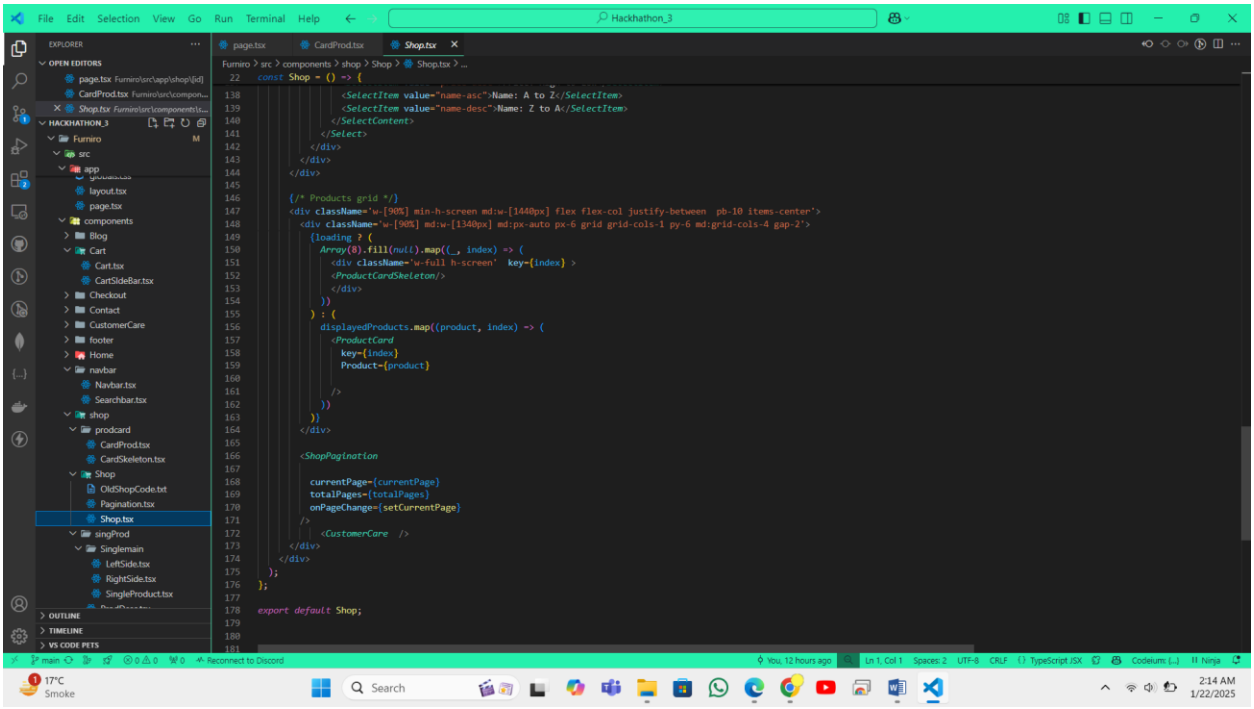


Code snippets for key components

Product Card Code.

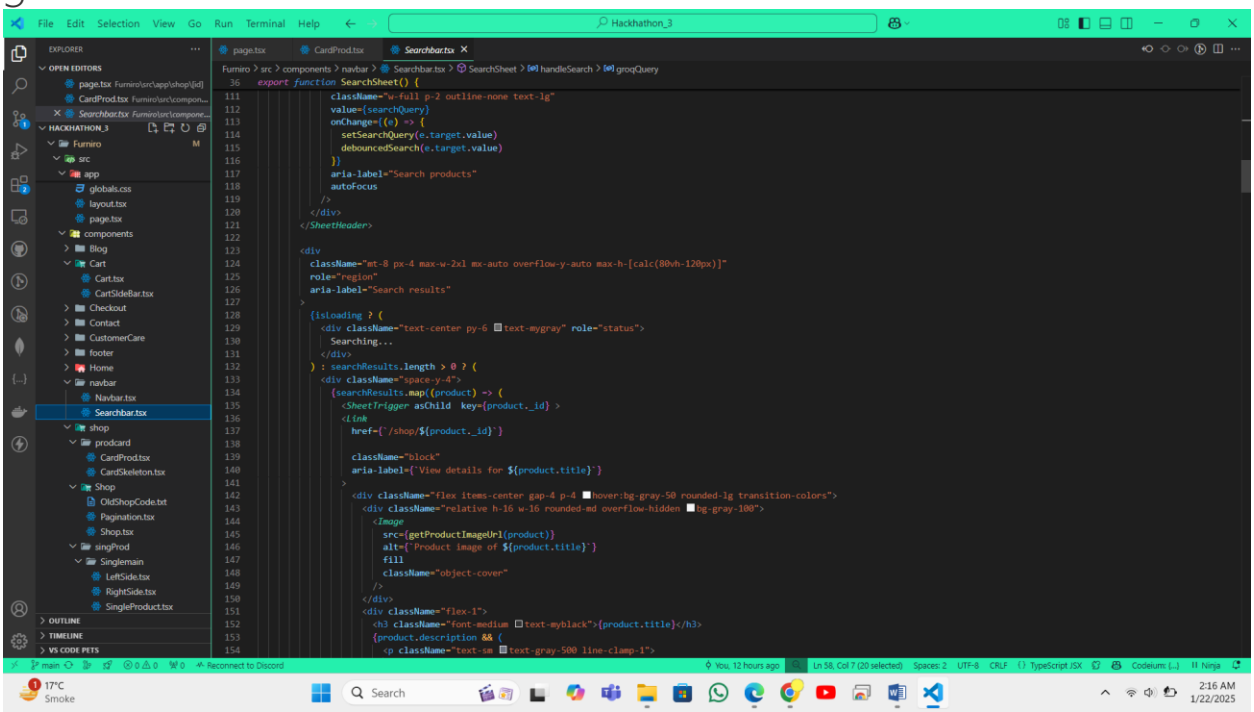


2. Product Listing Code



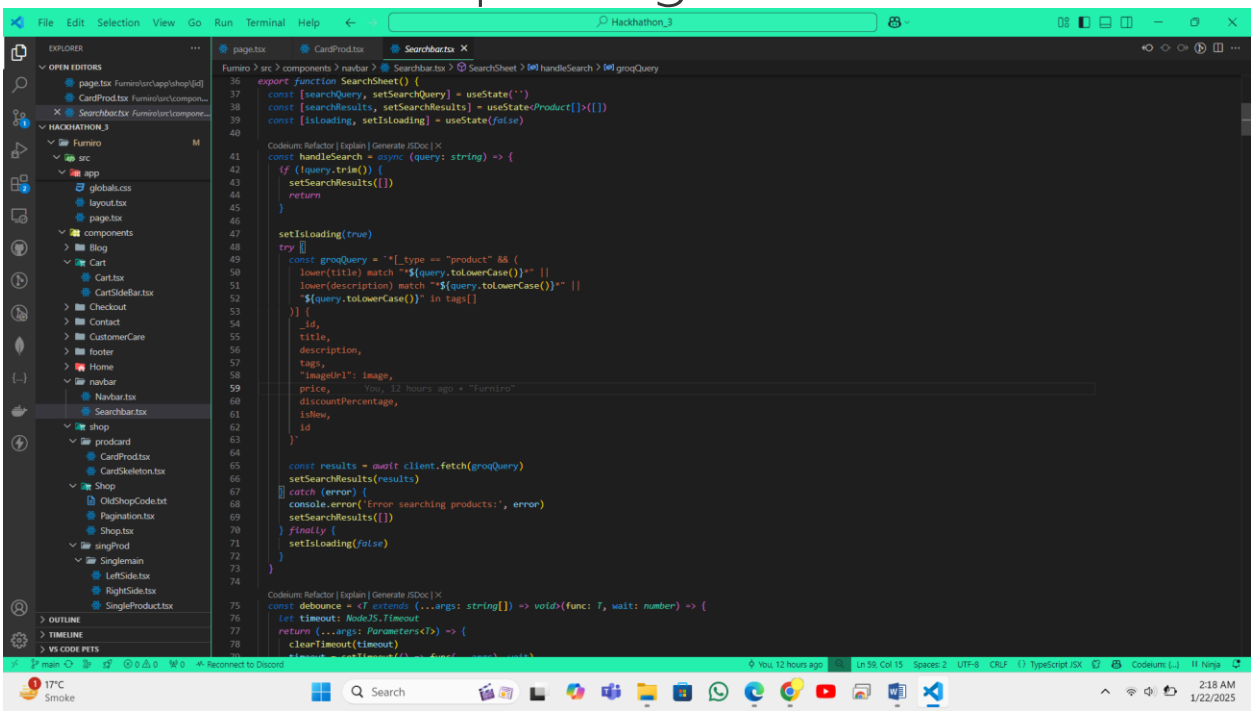
```
const Shop = () => {
                <SelectItem value="name-asc">Name: A to Z</SelectItem>
                <SelectItem value="name-desc">Name: Z to A</SelectItem>
              </SelectContent>
            </Select>
          </div>
        </div>
      </div>

      {/* Products grid */}
      <div className='w-[90%] min-h-screen md:w-[1440px] flex flex-col justify-between  pb-10 items-center'>
        <div class Name='w-[90%] md:w-[1340px] md:px-auto px-6 grid grid-cols-1 py-6 md:grid-cols-4 gap-2'>
          {loading ? (
            Array(8).fill(null).map((_, index) => (
              <div class Name='w-full h-screen'  key={index} >
              <ProductCardSkeleton/>
              </div>
            ))
          ) : (
            displayedProducts.map((product, index) => (
              <ProductCard
                key={index}
                Product={product}

              />
            ))
          )}
        </div>

        <ShopPagination

          currentPage={currentPage}
          totalPages={totalPages}
          onPageChange={setCurrentPage}
        />
            <CustomerCare  />
      </div>
    </div>
  );
};

export default Shop;
```

3.Search Bar Code

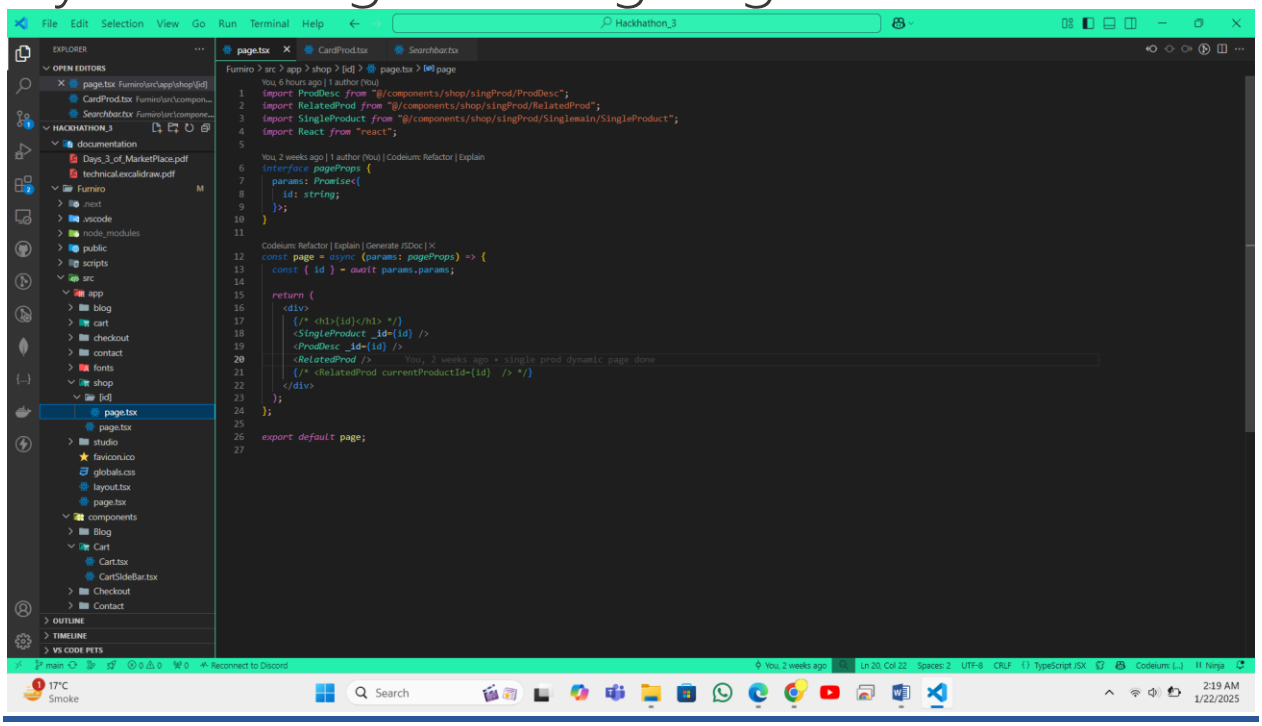


```
export function SearchSheet() {
              ClassName="w-full p-2 outline-none text-lg"
              value={searchQuery}
              onChange={(e) => {
                setSearchQuery(e.target.value)
                debouncedSearch(e.target.value)
              }}
              aria-label="Search products"
              autoFocus
            />
          </div>
        </SheetHeader>

        <div
          ClassName="mt-8 px-4 max-w-2xl mx-auto overflow-y-auto max-h-[calc(80vh-120px)]"
          role="region"
          aria-label="Search results"
        >
          {isloading ? (
            <div ClassName="text-center py-6 text-mygray" role="status">
              Searching...
            </div>
          ) : searchResults.length > 0 ? (
            <div ClassName="space-y-4">
              {searchResults.map((product) => (
                <SheetTrigger asChild  key={product._id} >
                <Link
                  href={`/shop/${product._id}`}

                  ClassName="block"
                  aria-label={`View details for ${product.title}`}
                >
                  <div ClassName="flex items-center gap-4 p-4 hover:bg-gray-50 rounded-lg transition-colors">
                    <div ClassName="relative h-16 w-16 rounded-md overflow-hidden bg-gray-100">
                      <Image
                        src={getProductImageUrl(product)}
                        alt={`Product image of ${product.title}`}
                        fill
                        ClassName="object-cover"
                      />
                    </div>
                    <div ClassName="flex-1">
                      <h3 ClassName="font-medium text-myblack">{product.title}</h3>
                      {product.description && (
                        <p ClassName="text-sm text-gray-500 line-clamp-1">
```

A overview of api integration code



Dynamic Page Coding Logic.



Steps Taken to Build and Integrate Components

1. **Dynamic Product Listing Page**
   - Initiated API integration to fetch product data and displayed it dynamically in a grid layout.
   - Utilized reusable components like ProductCard to ensure consistency and modularity across the application.
2. **Individual Product Detail Pages**
   - Configured dynamic routing to render product-specific detail pages.
   - Designed these pages to display comprehensive product details, including descriptions, specifications, and availability.
3. **Category Filters and Search Bar**
   - Implemented category-based filtering to dynamically update the product listing.
   - Developed a real-time search bar using controlled inputs, ensuring instant results.
4. **Pagination System**
   - Added pagination to handle large datasets efficiently, improving page performance and user experience.
5. **Add-to-Cart Route**
   - Designed and implemented the Add-to-Cart feature, ensuring smooth functionality and state persistence.
   - Managed the cart state using Context API, allowing global access to the cart data.
6. **Sidebar and Wishlist Sidebar with Context API**
   - Developed a collapsible sidebar for navigation, with state managed using Context API.
   - Created a wishlist sidebar to store and display favorite products dynamically.

Challenges Faced and Solutions Implemented

1. **Challenge: State Management for Filters and Cart**
   - **Issue**: Managing state across components, especially for filters, cart, and wishlists.
   - **Solution**: Utilized Context API for centralized state management, ensuring seamless data flow and synchronization.
2. **Challenge: API Latency**
   - **Issue**: Slow response times during data fetching led to delays in displaying content.
   - **Solution**: Implemented loading indicators and debouncing for search functionality to enhance user experience.

3. **Challenge: Large Dataset Pagination**
    - **Issue**: Rendering large datasets caused performance lags.
    - **Solution**: Implemented pagination to fetch and display data in smaller,

Best Practices Followed During Development

1. **Modular Design**
    - Components like ProductCard, ProductList, and SearchBar were designed to be reusable, improving maintainability.
2. **Responsive Design**
    - Ensured all pages and components adapted seamlessly to various screen sizes for a consistent user experience.
3. **Optimized State Management**
    - Leveraged Context API to efficiently manage and share state across components without prop-drilling.
4. **Efficient API Integration**
    - Used optimized API calls and caching strategies to reduce latency and improve performance.
5. **Clean and Scalable Codebase**
    - Followed clean coding principles and ensured the project was scalable for future enhancements.