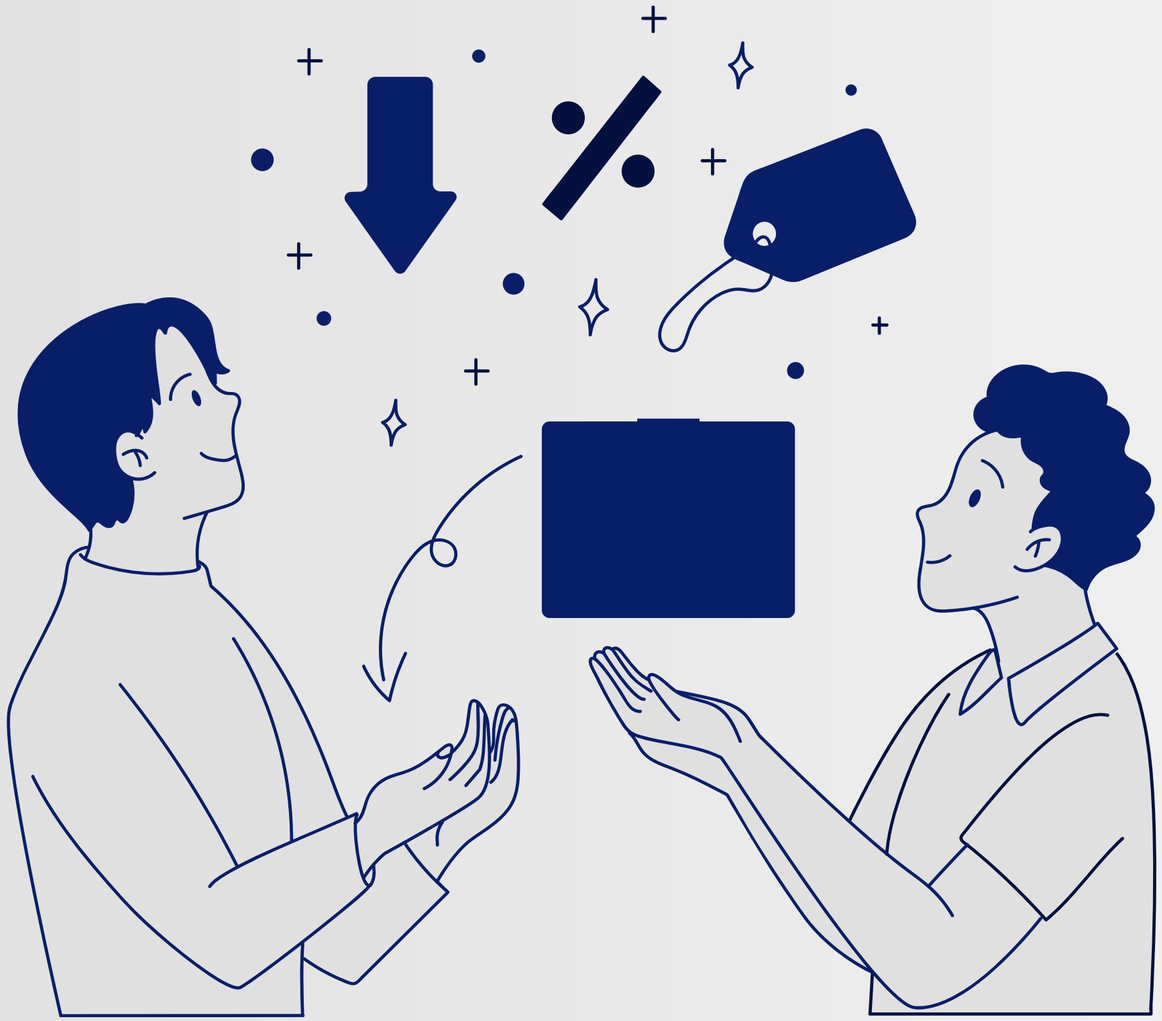
DAY 4 - DYNAMIC FRONTEND COMPONENTS SHOP.CO



OVERVIEW

On Day 4, I implemented key features to enhance the functionality of my e-commerce website, Shop.co, ensuring a dynamic and user-friendly experience. Below are the details of the work completed:

02

FEATURES IMPLEMENTED

Product Listing Page: Product Detail Page: Add to Cart Functionality with Redux: Pagination with ShadCN UI: Responsive Design with Tailwind css

TECHNICAL WORKFLOW

Product Listing Page Workflow

Fetched dynamic data from Sanity CMS to populate the product grid. Integrated category filters, a search bar, and pagination to allow users to browse and locate products seamlessly

Category Page:

Dynamically fetched products based on their categories using Sanity CMS. Rendered category-specific product listings with pagination for better user navigation

Product Detail Page:

Implemented individual product detail pages with accurate dynamic routing using Next.js.

Each detail page renders data dynamically from Sanity CMS, including

TECHNICAL WORKFLOW

Add to Cart Workflow:

Developed a slice using Redux Toolkit to manage cart state.
Configured store.ts to include persistence middleware for saving cart data in local storage.
Connected the toast notification component to trigger upon successful cart actions

Responsive Design:

Thoroughly tested and adjusted the layout for different screen sizes, ensuring consistent functionality and aesthetics

Toastify Notifications:

Created a client-rendered component to manage toast notifications efficiently.

Added success messages to confirm user actions, such as adding items to the cart

BEST PRACTICES FOLLOWED

Ensured code modularity by separating concerns into dedicated folderzzxzand files.

Followed clean code principles, making the project easy to maintain and extend. Implemented efficient state management using Redux Toolkit and redux-persist.

Tested the application thoroughly to ensure responsiveness and functionality across devices

BEST PRACTICES FOLLOWED

Ensured code modularity by separating concerns into dedicated folders and files. Followed clean code principles, making the project easy to maintain and extend. Implemented efficient state management using Redux Toolkit and redux-persist. Tested the application thoroughly to ensure responsiveness and functionality across devices