"Day 4 - Dynamic Frontend Components- HomeAura"

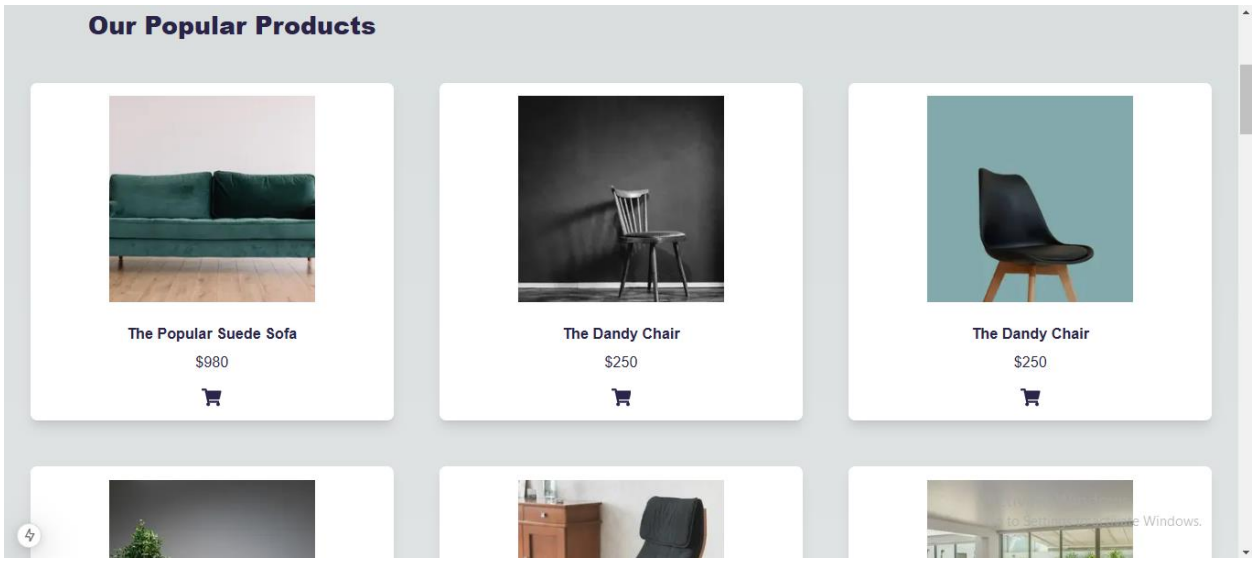
Here is a proper description on what I have done in my day 4 task of Hackathon..

Key Components Built:

1. Product Listing Component:

 I implemented a dynamic product listing that renders product data in a grid layout.
 The component displays essential fields such as:

- **Product Name**: The name of the product displayed for users.
- **Price**: The price of each product.
- **Image**: A product image to help users identify the product visually.
  This component helps to showcase a collection of products effectively in an organized manner.

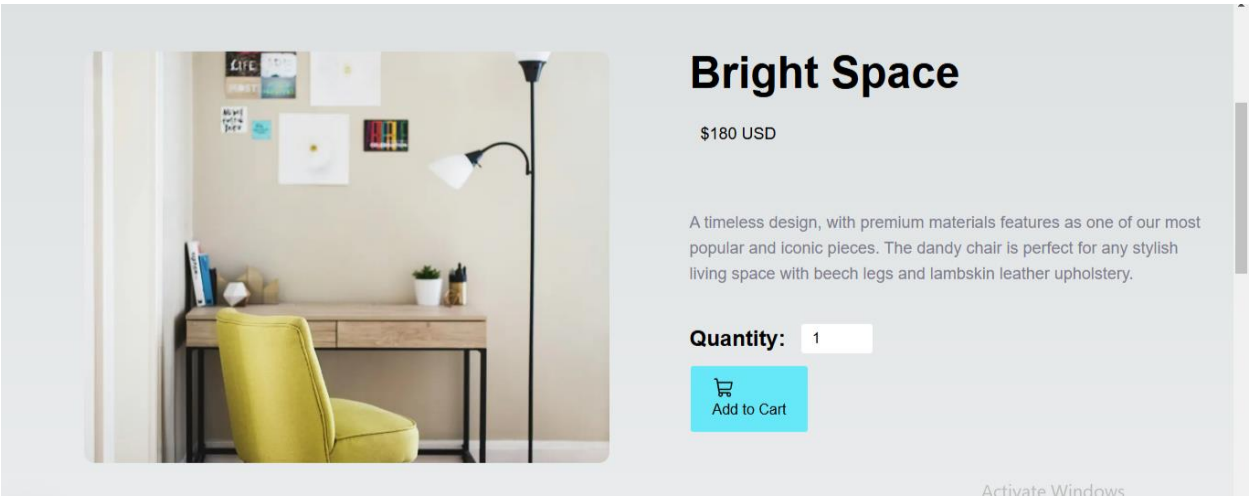


2. Product Detail Component:

 Each page provides detailed information about a specific product. The details include:

- Implemented this component by fetching product data from **Sanity CMS** using its API.
- The slug acts as a unique identifier for each product, ensuring that the correct product details are dynamically displayed on the corresponding page.

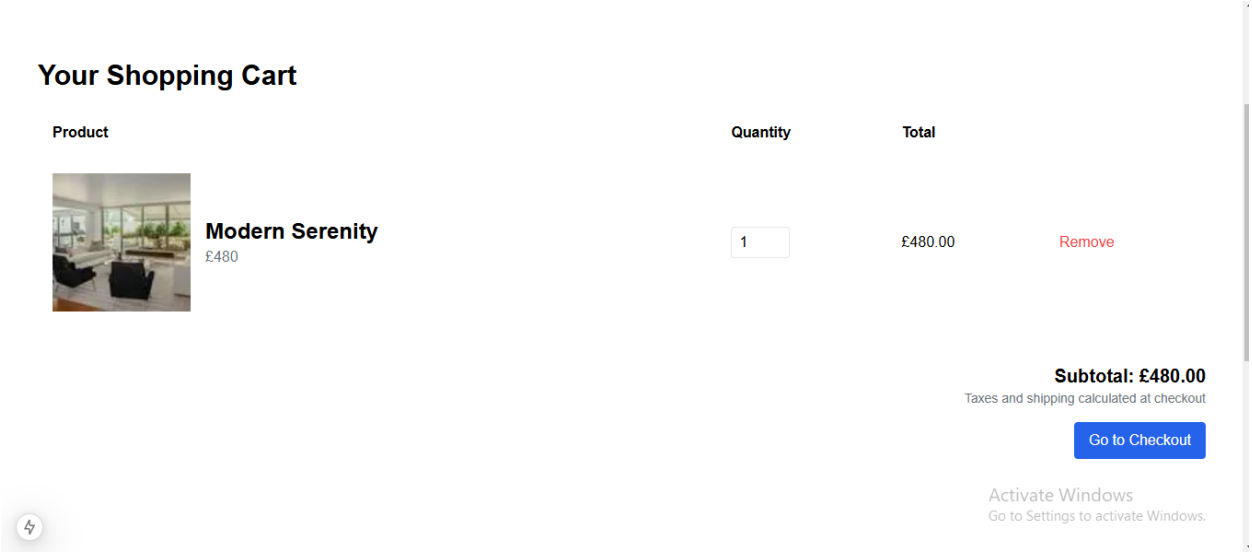
  a. **Product Description**: A detailed description that explains the features and specifications of the product.
  b. **Price**: The price of the specific product for purchasing consideration.
  This component allows users to view in-depth information about a product they are interested in.



3. Cart Component:

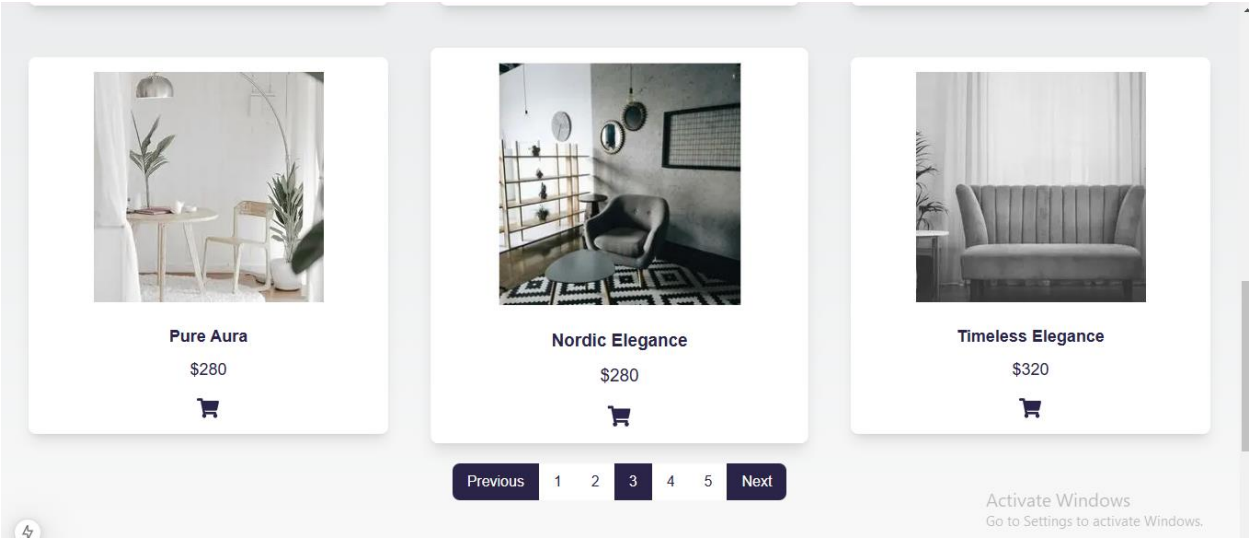
- Developed the cart component to display the products that a user adds to their shopping cart.
- It shows:

- **Items Added**: The products the user has added to their cart.
- **Quantity**: The number of units of each product added.
- **Total Price**: The total amount for the items in the cart, including any updates when the quantity changes.
- The component uses state management to track and update cart items dynamically, making the shopping experience seamless.



4. Pagination Component:

- I developed the pagination component to improve the usability of product listings.
- This component helps in breaking down large product lists into smaller, more manageable pages.
- I included features like:
    - **Previous and Next Buttons**: For easy navigation between pages.
    - **Numbered Pagination**: To allow users to jump directly to a specific page in the list.
      Pagination ensures that users can navigate through the product catalog without being overwhelmed by too many items on a single page.



5. Footer and Header Components:

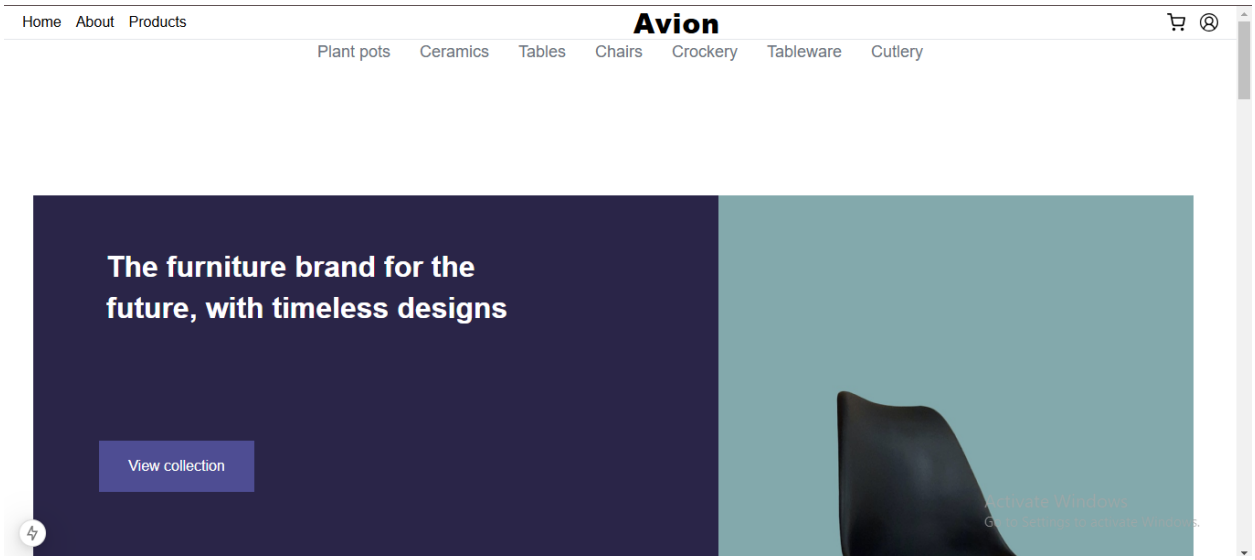
 I created consistent footer and header components that are present across all pages of the site.

 The header includes essential navigation links to the main pages of the site, such as:

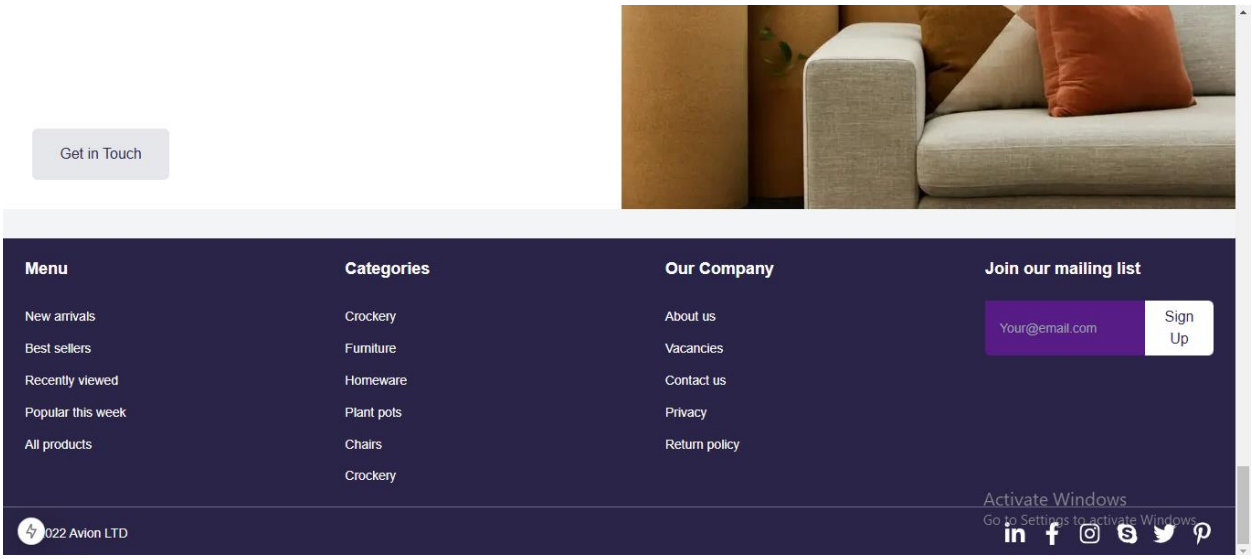
- **Home, About, Products**: Directs users to key areas of the site.

 The footer includes additional information and links relevant to the site, ensuring a consistent experience.
 Both components are fully responsive, ensuring a smooth user experience on different screen sizes and devices. I also focused on making them accessible to all users, following best practices in web design

Screenshot of Header Component:

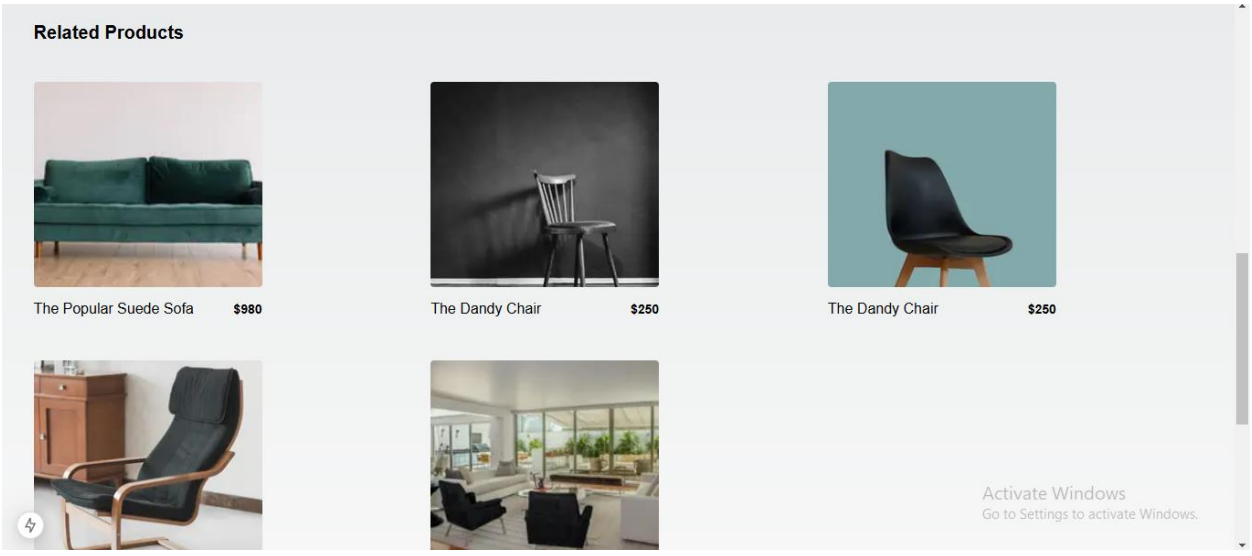


Screenshot of Footer Component:



6. Related Products Component:

- Shown related products below the product details page to users find their related products easily from here
- They can just click on the product they want and the product with full detail page will be open.

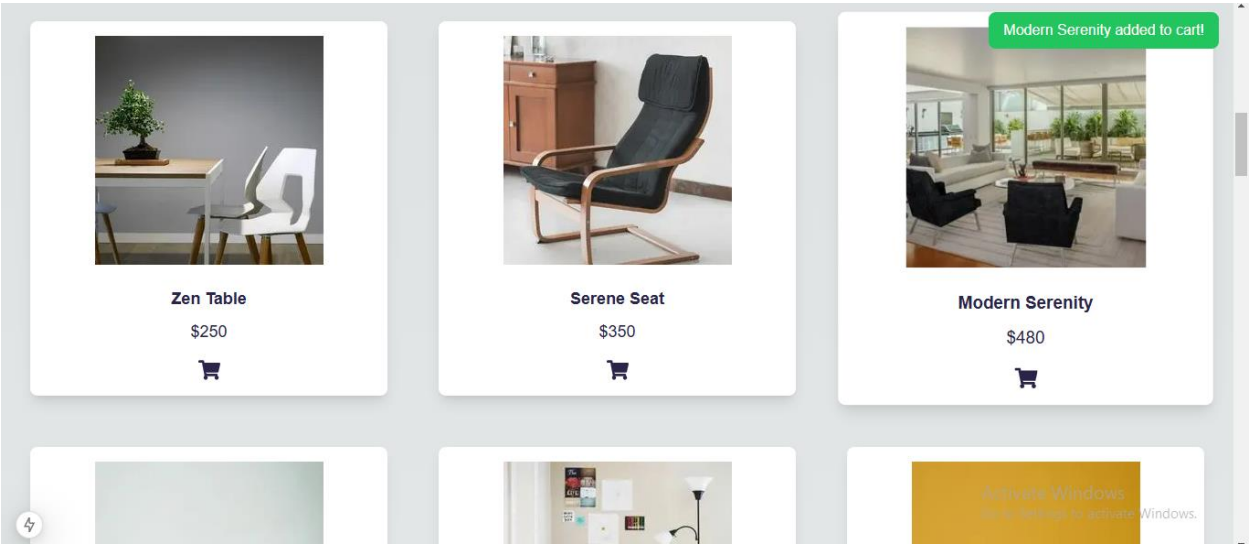


7. Products and Toast Integration:

- Implemented a **toast notification** system to enhance user interaction and provide instant feedback.

- Added functionality to display a toast message whenever a user:
  a. Clicks on the **"Add to Cart"** button for any product.
  b. Interacts with the **cart icon**, confirming actions like adding items to the cart.
- The toast message displays a confirmation (e.g., *"Added to Cart"*) to notify the user that their action was successful.

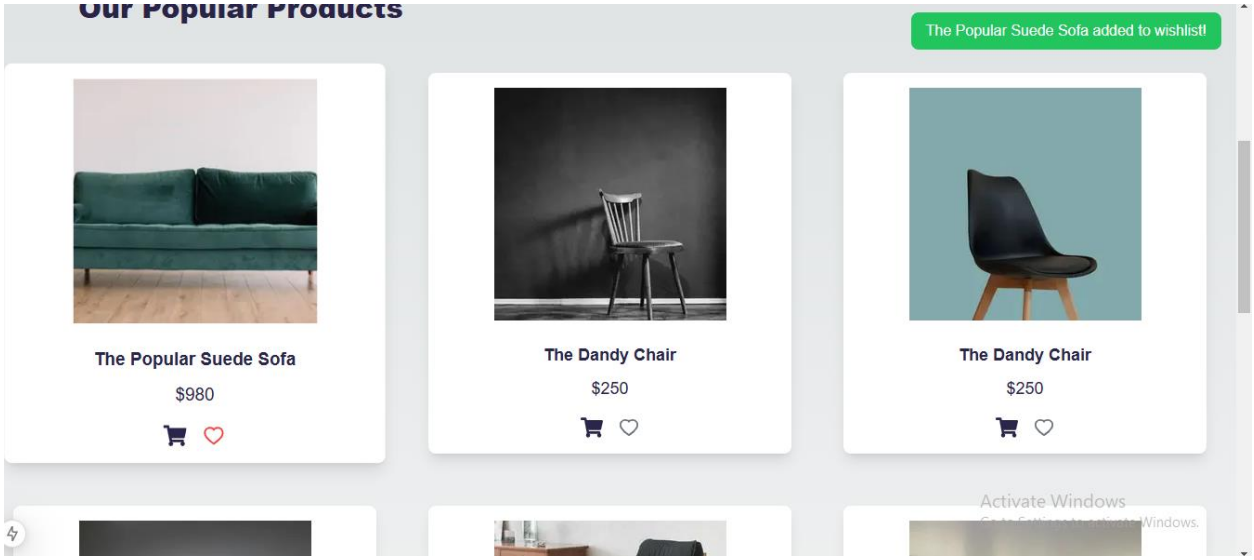
Screenshot Of Toast Integration:



8. Wishlist Component:

- Developed the wishlist component to allow users to save their favorite products for future reference or purchase.
  • It shows:
  • Items Saved: The products the user has added to their wishlist.
  • Product Details: Key information like price and description of each saved item.
  • Option to Add to Cart: Users can move products from the wishlist to the shopping cart with a single click.
  • The component uses state management to track saved items and dynamically update the wishlist, improving the user experience.

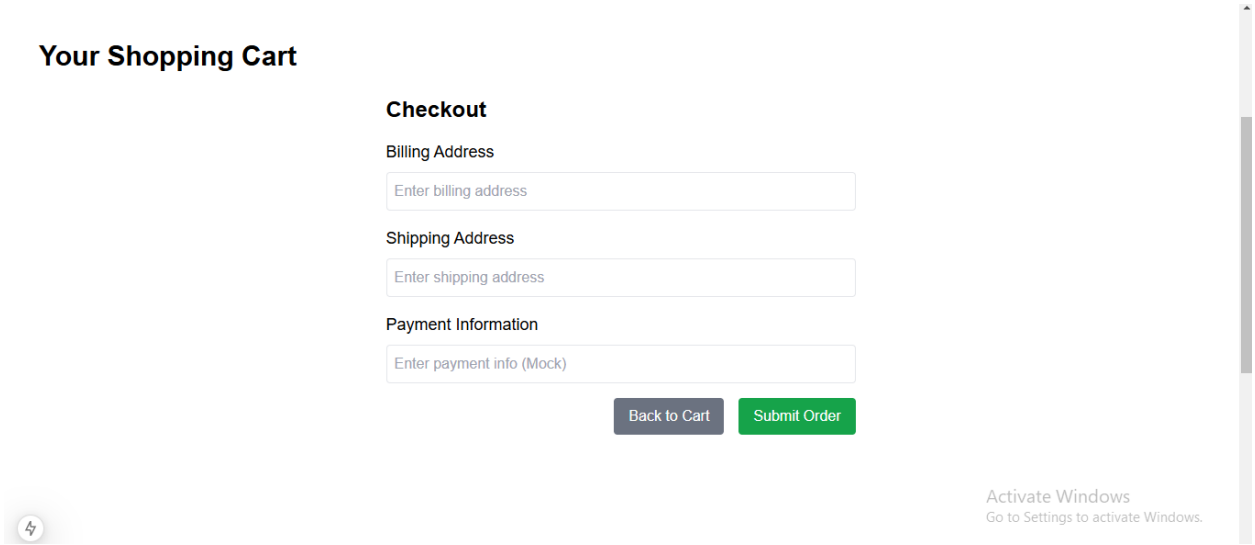
Screenshot of wishlist:



9. Checkout Flow Component:

- A multi-step form to collect customer details, billing and shipping addresses, and payment information, with conditional steps for easy navigation.
- An order summary page that displays customer details, addresses, and payment method after submission, along with cart validation to ensure items are added before proceeding.

Screenshot of Checkout:



Technical Report Summary:

- **Steps taken to build and integrate components:**

Components were first designed based on the project requirements.

- Then, the integration of components was done by following the specific layout structure.

- Each component was tested for functionality before moving on to the next phase.

- Data for the components was fetched from **Sanity CMS** using its API to ensure dynamic and real-time updates.

- • Each component was thoroughly tested for functionality and responsiveness before moving on to the next phase.

• Challenges faced and solutions implemented:

- **Challenge:** Ensuring responsiveness on different screen sizes.
  Solution: Utilized Tailwind CSS media queries to handle responsiveness.
- Managing state effectively across components, especially for features like the cart.
  Solution: Utilized React state management tools such as useState and useContext to track and manage data efficiently.
- **Challenge:** Fetching and displaying dynamic product data while maintaining performance.
  Solution: Integrated **Sanity CMS** to fetch real-time data using GROQ queries and ensured optimal API usage for dynamic routing in Next.js.
- **Challenge:** Handling unique product slugs for dynamic routing and rendering.

• Best practices followed during development:

- Keeping code modular and reusable.
- Writing clear and concise comments for better understanding.
- Following accessibility standards for a more inclusive web.
- Managing state effectively across components, especially for features like the cart.
  Solution: Utilized React state management tools such as useState and useContext to track and manage data efficiently.
- **Challenge:** Fetching and displaying dynamic product data while maintaining performance.
  Solution: Integrated **Sanity CMS** to fetch real-time data using GROQ queries and ensured optimal API usage for dynamic routing in Next.js.
- **Challenge:** Handling unique product slugs for dynamic routing and rendering.

Checklist for Day 04:

Self-Validation Checklist

Frontend Component Development ✔

Styling and Responsiveness ✔

Code Quality ✔

Documentation and Submission ✔

By: Duaa Pirzada