Day 4 - Building Dynamic Frontend Components for Hekto E-commerce

NAME: Zakia Bashir

Roll No: 141704

Objective:

On Day 4, the goal is to design and develop dynamic frontend components that show marketplace data from Sanity CMS or APIs. The focus is on creating modular, reusable components to build scalable and responsive web applications.

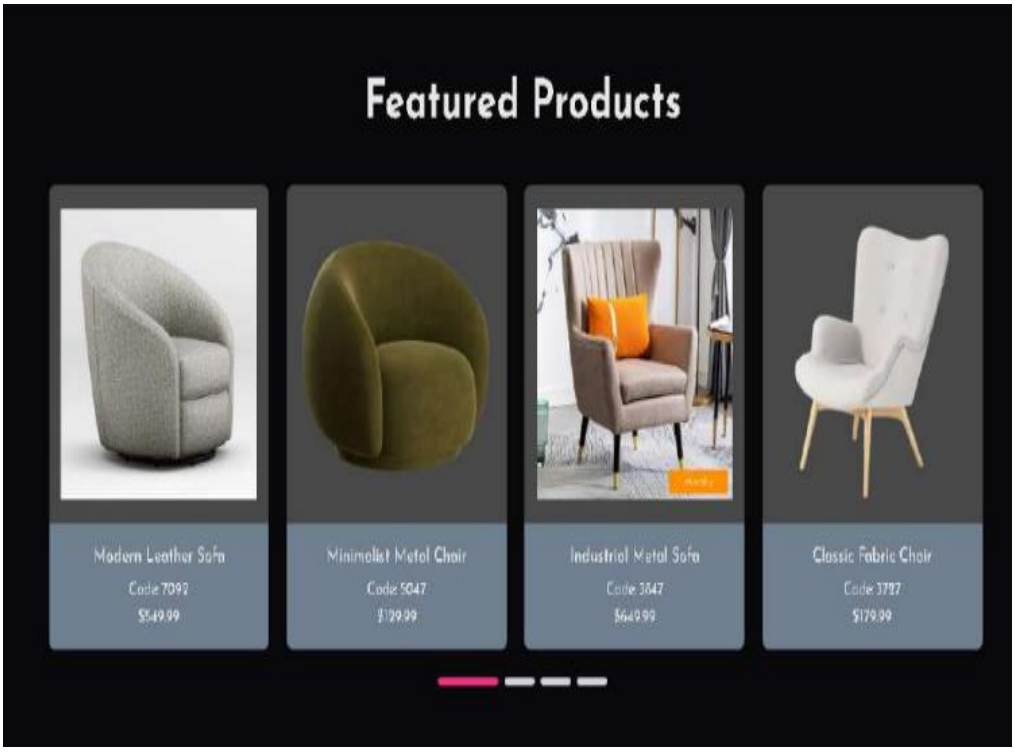
1. **Build Dynamic Frontend Components to Display Data from Sanity CMS:**
    - Connected the Next.js app with Sanity CMS.
    - Configured the Sanity client and used GROQ queries to fetch essential data like product ID, image, name, stock, and price.
    - Example query:

javascript

```
const query = `*[_type == "products"]{
_id,
productName,
productDescription,
price,
prevPrice,
stock,
```

```
productImage,
tag,
shipmentArray[] {
trackingId,
deliveryStatus,
estimatedDeliveryDate
}
}`;
```

- Mapped the fetched data into frontend components for flexible, dynamic displays that automatically update with the latest information.
- Used Sanity's real-time data to enhance the user experience with fresh and easily editable content.



2. Implement Reusable and Modular Components:

- Created components like Featured Products, Latest Products, Trending Products, product grids, and individual product displays.
- Ensured these components can be reused across multiple pages without rewriting the same code.
- By creating reusable Product components, individual products can be displayed anywhere within the app without duplicating code, keeping the codebase modular and easier to maintain.

3. Understand and Apply State Management:

- Applied state management techniques to handle the dynamic data and ensure smooth user interactions.

Understand and Apply State Management Techniques:

- Used React's useState and useEffect hooks to manage product data.
- useState declares a state variable to hold product data (like name, price, etc.).

javascript

```
const [products, setProducts] = useState<ProductType[]>([]);
```

- useEffect fetches product data from the backend (Sanity CMS) when the component mounts.

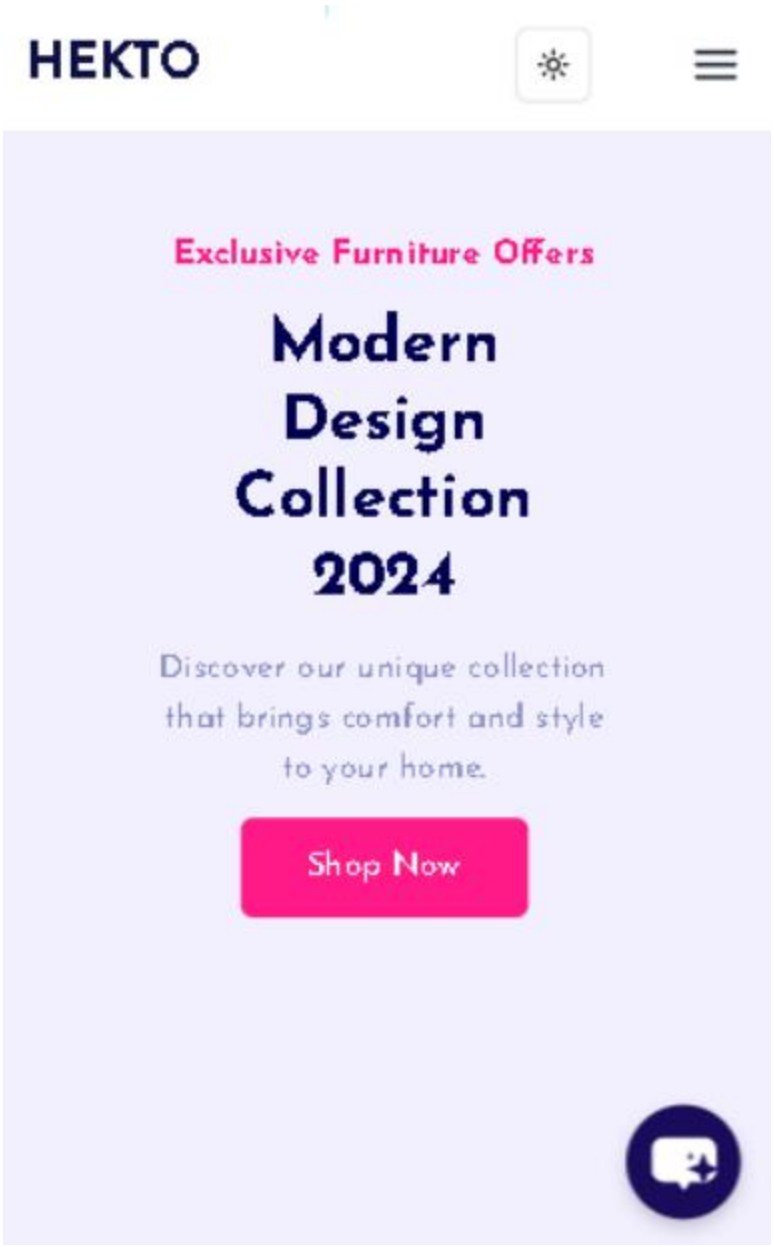
javascript

```
useEffect(() => {
  // Fetch products when the component mounts
  const getProducts = async () => {
    const fetchedProducts = await fetchProducts();
    const filteredProducts = fetchedProducts.filter(product => /* condition */);

    // Update the state with the fetched products
    setProducts(filteredProducts);
  };

  getProducts();
}, []); // Empty dependency array ensures this runs once when the component mounts
```

- When data is fetched, it updates the state, and React automatically re-renders the component with the new data.

4. Learn the Importance of Responsive Design and UX/UI Best Practices:

- Ensured that product listing components were responsive on all devices, providing an optimal user experience for both mobile and desktop users.



5. How I Replicated Professional Workflows for Real-World Client Projects:

- **Modular Components:**
    - Followed best practices by breaking the project into reusable components, making the code scalable and easier to maintain.
- **State Management with Hooks:**

    - Used React's useState and useEffect to manage the application state efficiently.
    - For larger projects, implemented Redux for state management to handle complex states across different components.
- **Error Handling and Edge Case Management:**
    - Incorporated error handling and managed edge cases for a production-ready application.

javascript

```
if (typeof window !== "undefined") {
  try {
    const serializedWishlist = localStorage.getItem('wishlist');
    return serializedWishlist ? JSON.parse(serializedWishlist) : undefined;
  } catch (error) {
    console.error('Error loading wishlist from localStorage:', error);
    return undefined;
  }
}
```

6. Create Individual Product Detail Pages Using Dynamic Routing in Next.js:

- Created a dynamic route in the src/app/FeaturedProduct/[id]/page.tsx file.
- This structure allows each product to have its own unique URL based on the product ID and dynamically loads product details from a backend API or CMS (such as Sanity).

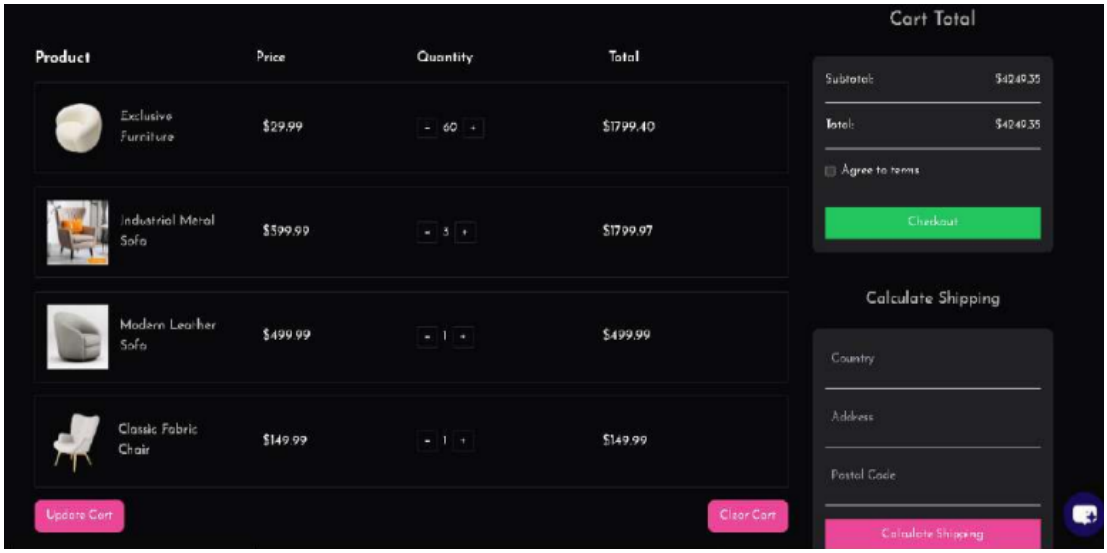
7. Cart Component:

I. Handling Stock and Cart Button State:

- The "Add to Cart" button is conditionally enabled or disabled based on the available stock.
- When stock is less than or equal to zero, the button becomes disabled to prevent users from adding more items than are available.
- Once a product is added to the cart, the stock count is decremented accordingly.

II. Displaying Added Items, Quantity, and Total Price:

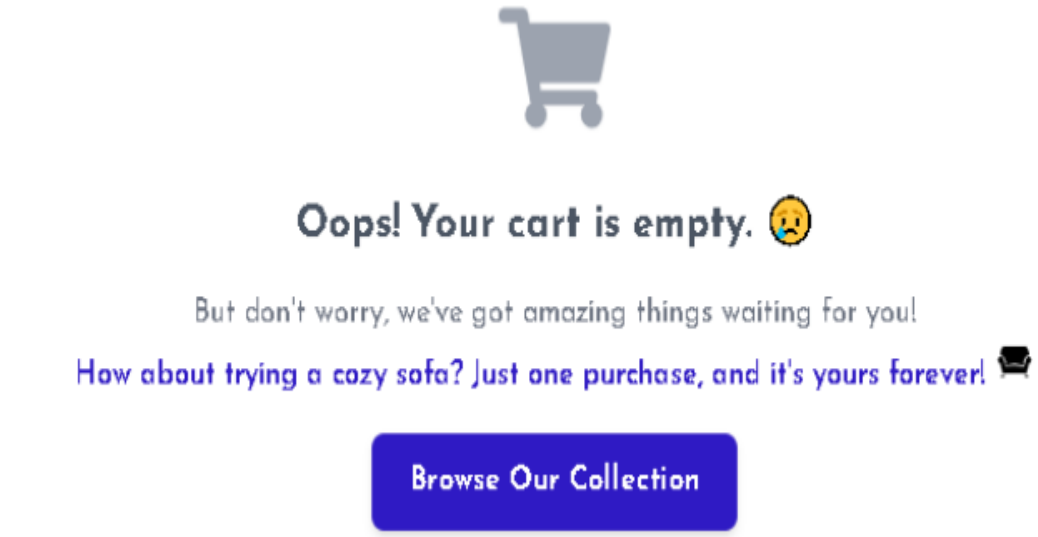
- The Cart component tracks the products added to the cart, showing key details such as product name, price, quantity, and total price.
- Displays products in a list format with options to modify the quantity or remove the product from the cart.



- Adjusting the quantity with the "+" or "–" buttons dynamically updates the total price displayed.

```
<div className="w-full min-h-[50vh] flex items-center justify-center">
  {cart.items.length === 0 ? (
    <div className="h-full flex flex-col justify-center items-center">
      <FaShoppingCart className="text-7xl text-gray-400 mb-4 animate-bounce" />
      <p className="text-2xl text-gray-600 dark:text-white font-semibold">
        Oops! Your cart is empty. <span className="text-yellow-500">But don&apos;t worry, we&apos;ve got amazing things waiting for you!</span>
      </p>
      <p className="text-gray-500 text-lg">
        How about trying a cozy sofa? Just one purchase, and it&apos;s yours!
        <span className="ml-2 text-2xl text-black inline-flex">
          <GiSofa />
        </span>
      </p>
```

```
      <Link href="/pages/shopLeft" className="px-6 py-3 bg-[#2F1AC4] text-white text-lg font-semibold">
        Browse Our Collection
      </Link>
    </div>
  ) : (
    // Render cart items here
  )}
</div>
```



8. Notification Component:

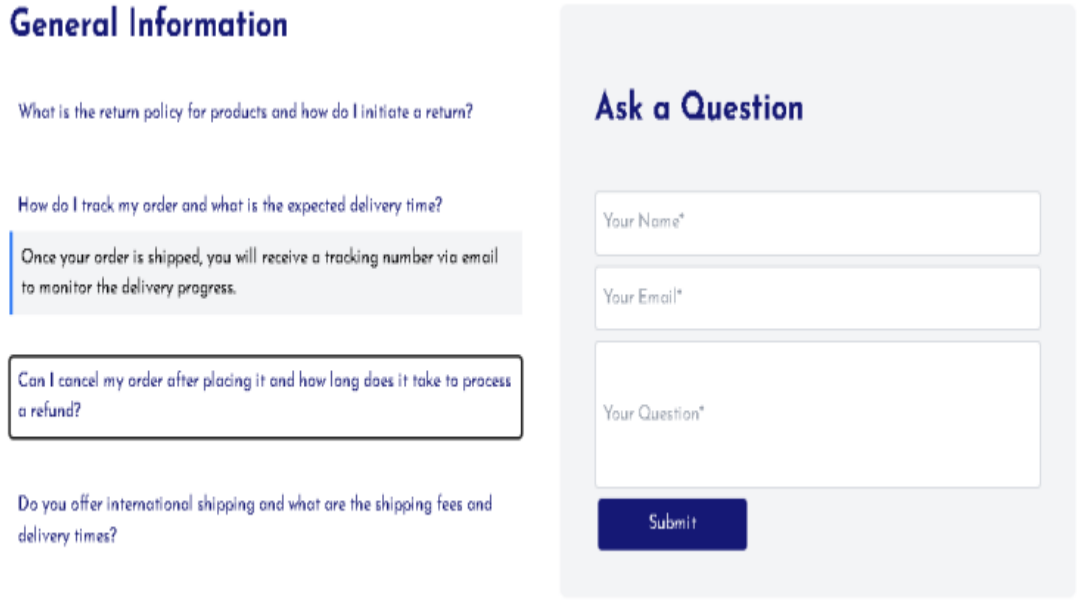
- Implemented a well-structured toast notification system that triggers after adding an item to the wishlist or cart.
- The notifications are user-friendly and provide instant feedback with clear messages, enhancing the overall user experience.

```
toast.success(
  <div className="flex items-center space-x-4">
    <FaYahoo size={24} className="text-[#FB2E86]" />
    <div>
```

```
      <h4 className="font-semibold text-lg text-green-500">Success!</h4>
      <p className="text-sm text-gray-200">
        You added <strong>{product.productName}</strong> to the cart.
      </p>
    </div>
  </div>
);
```

9. FAQ and Help Center Component:

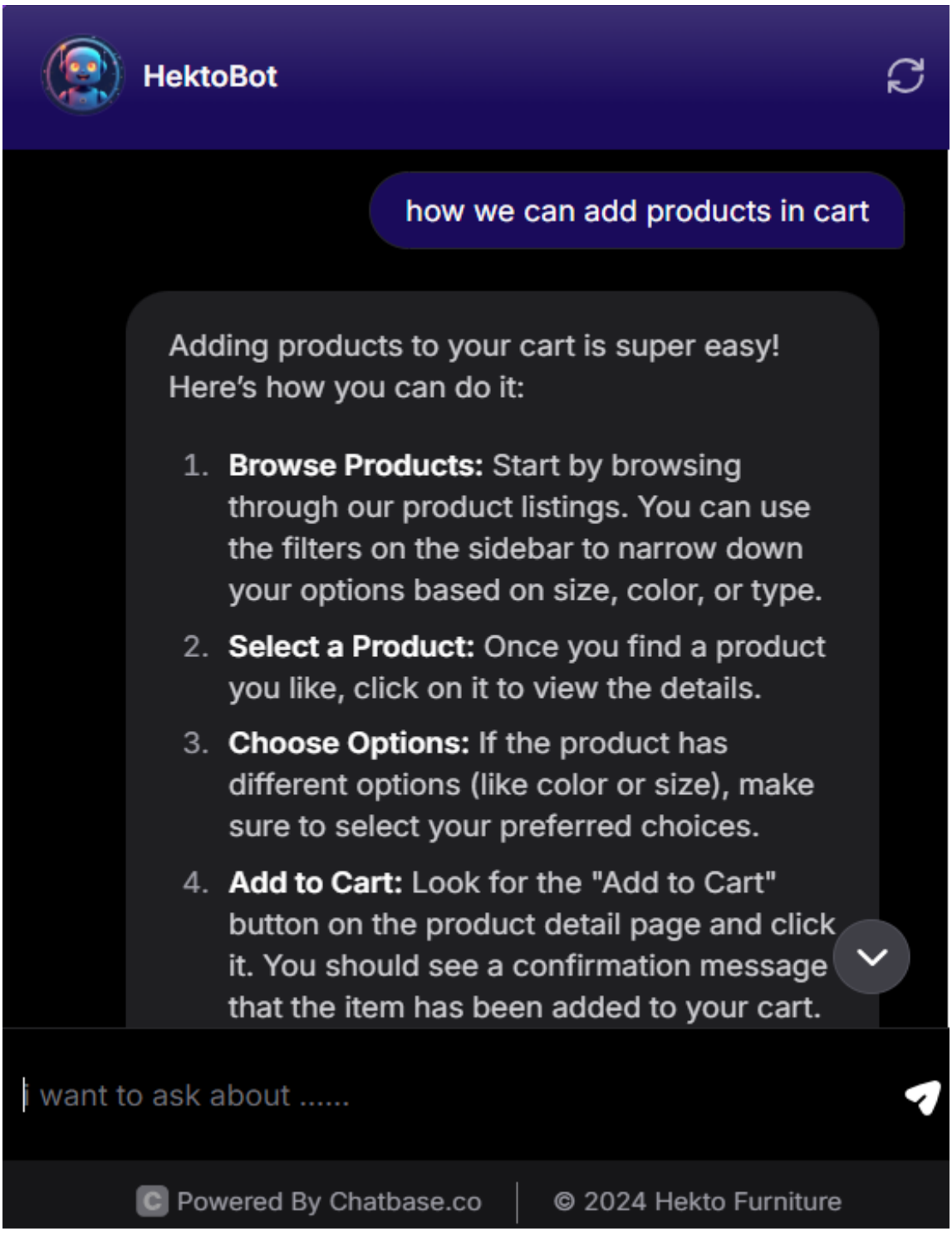
- Created a comprehensive and organized FAQ and Help Center to address common questions and provide users with quick, clear, and accessible solutions, ensuring a seamless and supportive experience.



10. Personal AI Chatbot Integration:

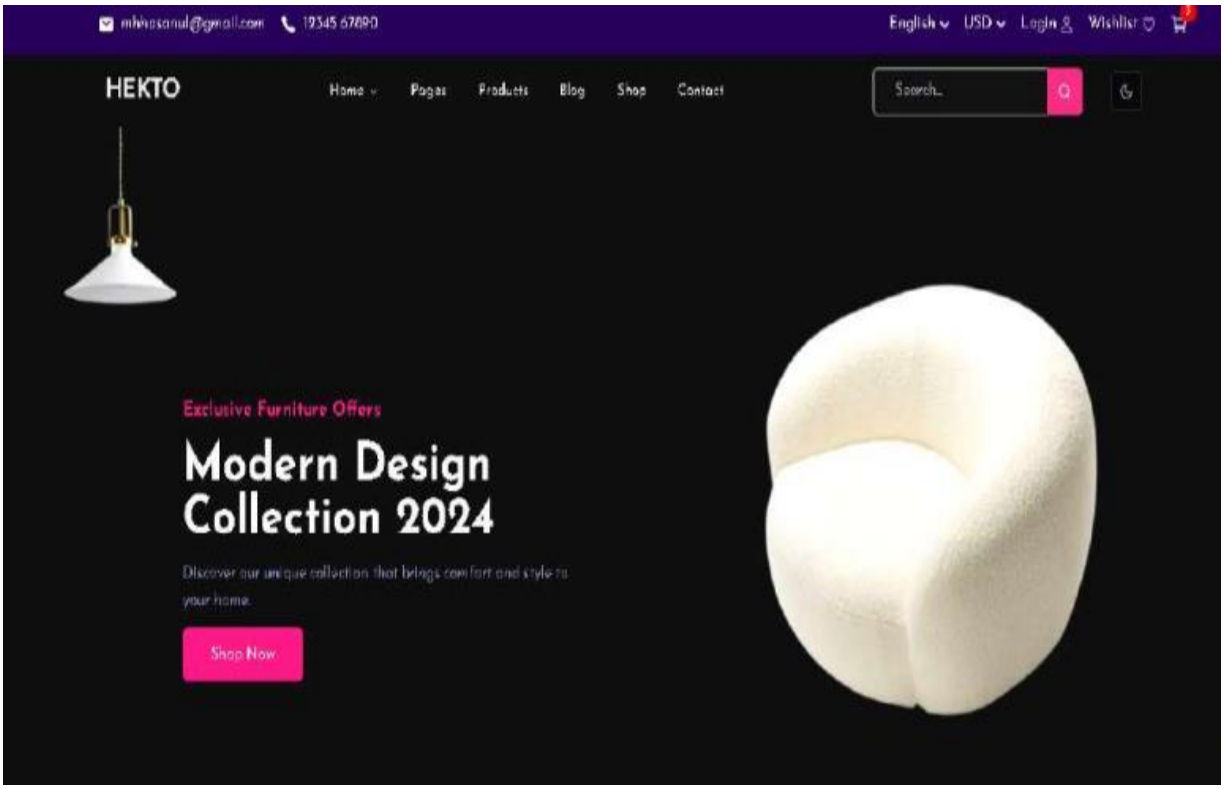
- Developed a personalized AI chatbot to enhance user engagement and streamline support.
- The chatbot provides instant responses, assists with queries, and delivers a tailored user experience.

- It seamlessly integrates with websites or platforms, making interactions efficient and accessible for users.



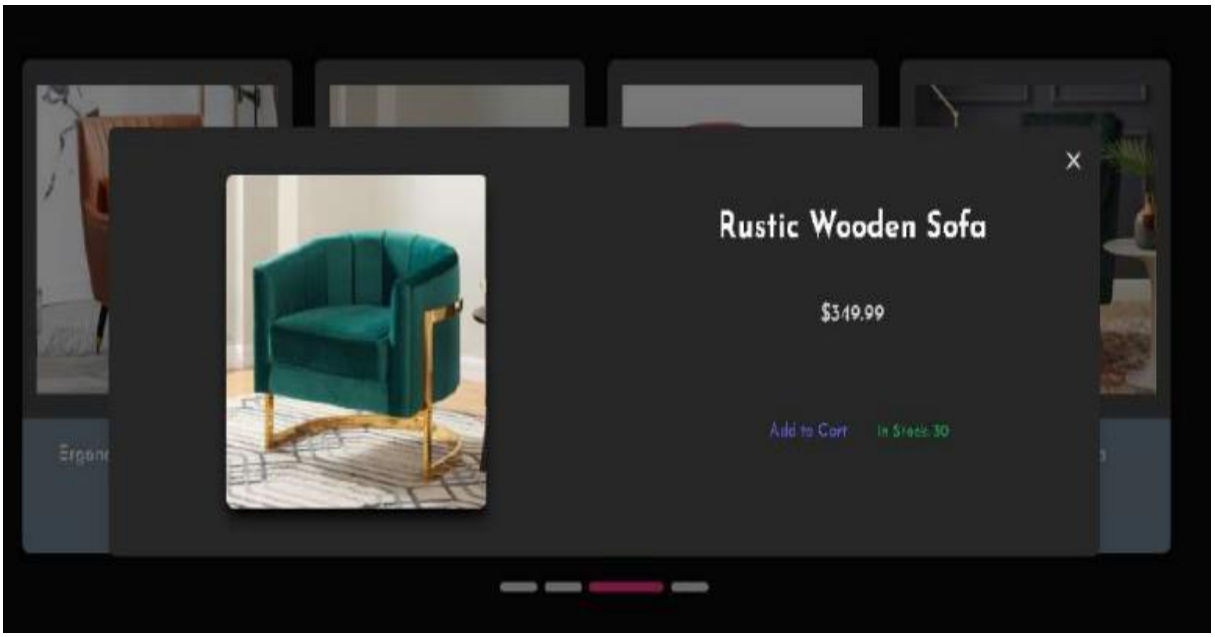
11. Seamless Theme Toggle: Light and Dark Modes:

- Implemented a smooth theme toggle feature, allowing users to switch between light and dark modes effortlessly.
- This enhances user experience by catering to individual preferences and providing optimal readability in any lighting condition.
- The transition is visually appealing, ensuring a cohesive and modern design across both themes.



12. Interactive Zoom Feature for Enhanced Viewing:

- Integrated a dynamic zoom functionality to enhance the product viewing experience.
- On hover, a sleek zoom icon (🔎) appears, indicating the feature's availability.
- With a simple click, users can view the product in greater detail without leaving the current page.
- This interactive feature provides a seamless and immersive way to explore product details, ensuring convenience and clarity.



Conclusion:

- Gained a better understanding of the importance of modular and scalable design in frontend development.
- Created dynamic components and ensured they're responsive across devices, establishing a strong foundation for building a professional marketplace.
- Integrating data from APIs and CMSs was a hands-on experience that will be invaluable in real-world projects.
- Working with Redux on the add-to-cart feature provided a clearer understanding of state management, making the marketplace more interactive and user-friendly.
- Focused on building reusable components, applying UX/UI best practices, and ensuring the design worked seamlessly across different devices.
- This experience has prepared me to tackle real-world client projects, as I've replicated professional workflows and learned essential frontend development practices that will be crucial moving forward.