DAY 5 - TESTING, ERROR HANDLING, AND BACKEND INTEGRATION REFINEMENT

**Test Core Features:**

CODE:  
async function getProductById(id) {

  const query = `\*[\_type == "product" && id == $id][0]{

    \_id,

    name,

    description,

    price,

    discountPercentage,

    tags,

    image{

      asset->{

        url

      }

    },

    rating,

    rateCount

  }`;

  const product = await client.fetch(query, { id });

  return product;

}

export default function ProductDetailPage({ params }) {

  const { id } = React.use(params);

  const [product, setProduct] = useState();

  const [loading, setLoading] = useState(true);

  const dispatch = useDispatch();

  useEffect(() => {

    async function fetchProduct() {

      try {

        const productData = await getProductById(id);

        setProduct(productData);

      } catch (error) {

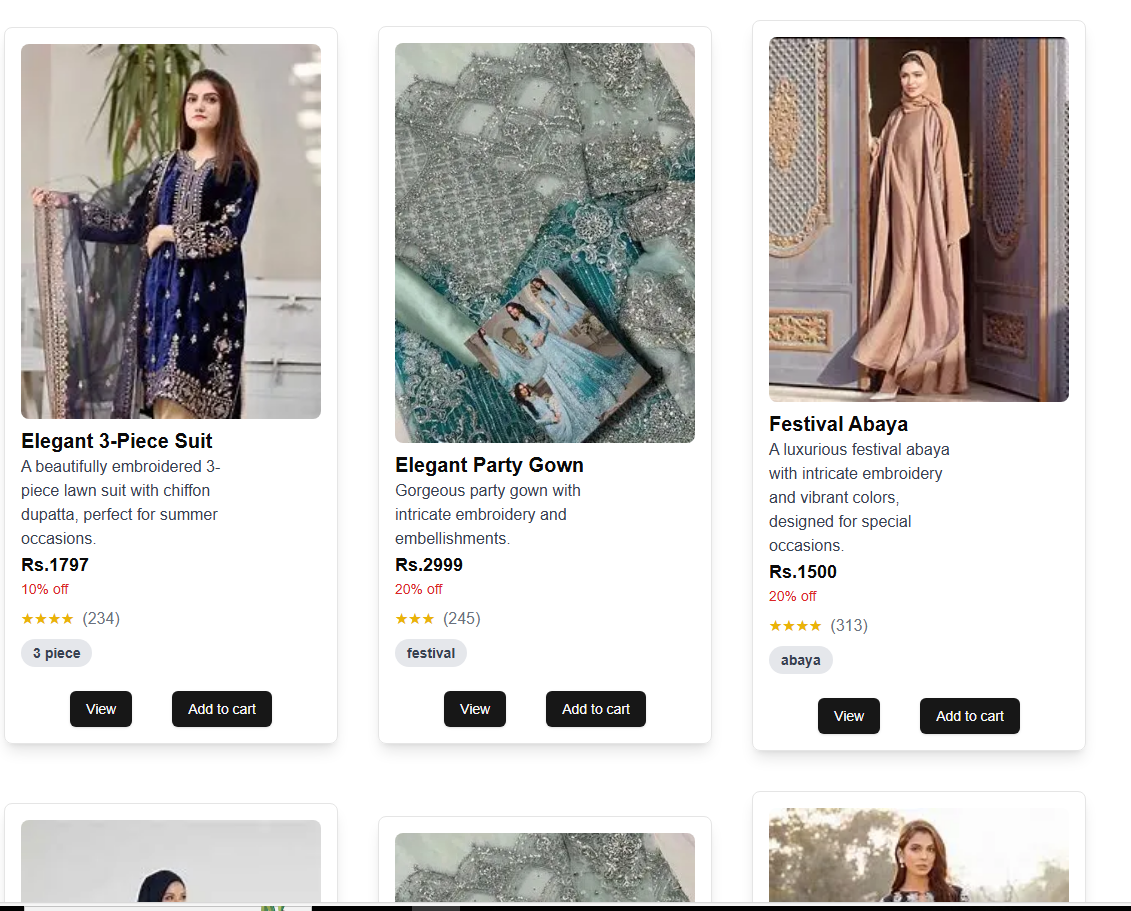
        console.error("Error fetching product:", error);

      } finally {

        setLoading(false);} }

    fetchProduct();

  }, [id]);



**Cart operations: Add, update, and remove items from the cart:  
CODE:**const handleCheckout = async () => {

    try {

      const cartItemsForCheckout = cartItems.map((item) => ({

        name: item.name,

        price: item.price,

        quantity: item.quantity,

      }));

      const response = await fetch("/api/create-checkout-session", {

        method: "POST",

        headers: {

          "Content-Type": "application/json",

        },

        body: JSON.stringify({ cartItems: cartItemsForCheckout }),

      });

      const { id } = await response.json();

      const stripe = await loadStripe(process.env.NEXT\_PUBLIC\_STRIPE\_PUBLISHABLE\_KEY);

      await stripe.redirectToCheckout({ sessionId: id });

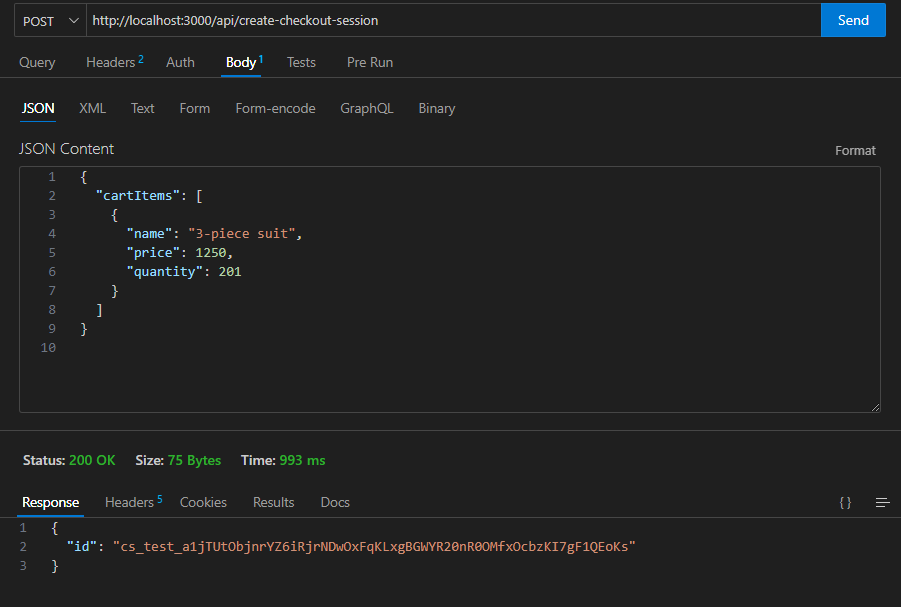
    }

catch (error) {

      console.error("Error during checkout:", error);

    }

  };



**Testing Tools:**

* **Thunder client:** For API response testing to verify data retrieval and submission.
* **React Testing Library:** For testing UI component behavior and state management.

**Error Handling**

**1. Add Error Messages:**

* Implement try-catch blocks to handle API errors effectively.

**Example:**

try {

        const productData = await getProductById(id);

        setProduct(productData);

      } catch (error) {

        console.error("Error fetching product:", error);

      } finally {

        setLoading(false);

      }

**2. Fallback UI:**

* Display alternative content when data is unavailable.
* Example: "No items found" message for an empty product list.

**Cross-Browser and Device Testing**

1. **Browser Testing:**

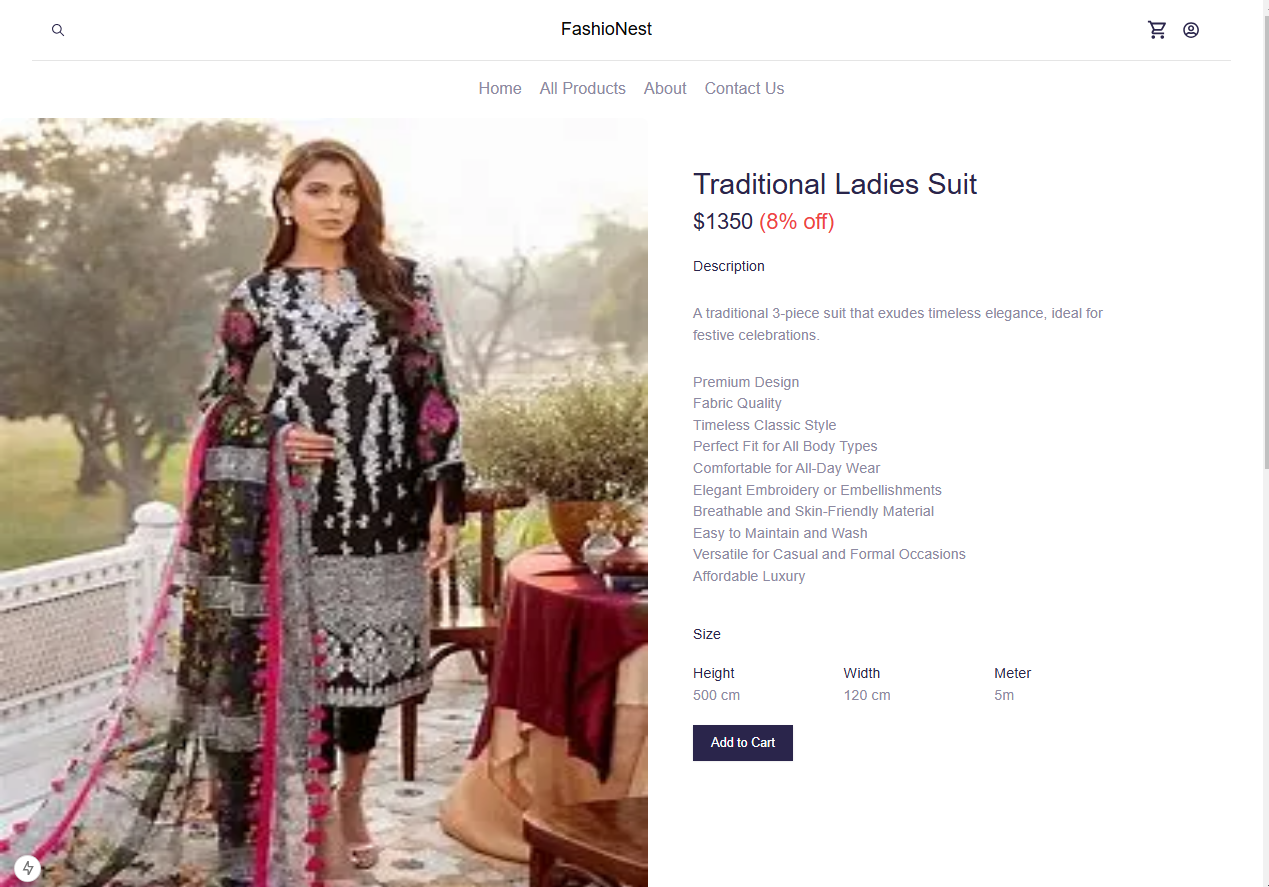
o Test on Chrome, and Edge.

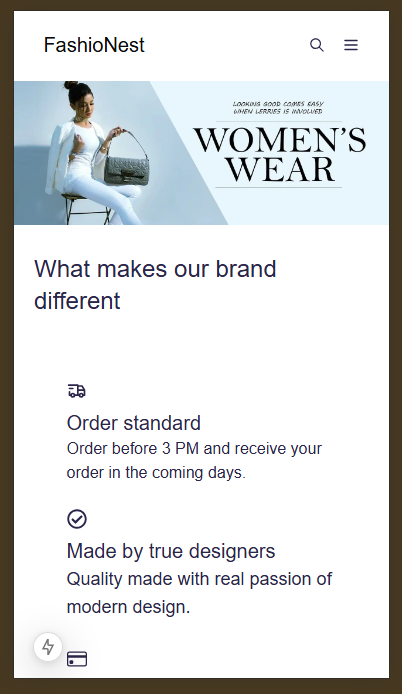
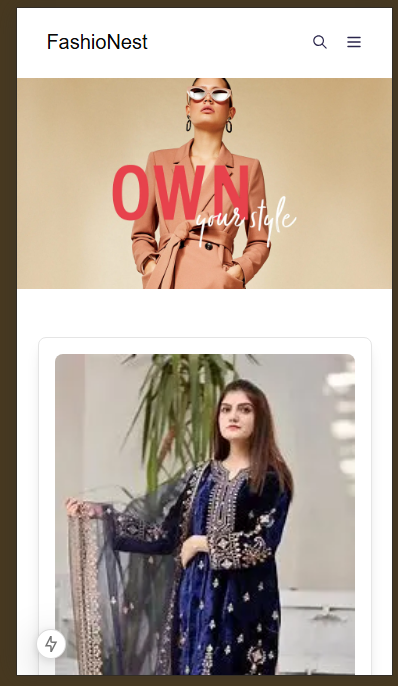
o Verify consistent rendering and functionality.

1. **Device Testing:**

o Use responsive design tools.

o Manually test on at least one physical mobile device.



**Testing Results**

**Key Issues Found and Resolutions**

**Issue 1: Product Not Found on Vercel**

* **Description**: The product detail page (/Productlist/[id]) worked on localhost but returned "Product not found" on the Vercel deployment.
* **Root Cause**:
  + Environment variables (e.g., Sanity project ID) were not properly configured in Vercel.
  + The id parameter in the URL was not matching the id in the Sanity dataset.
* **Resolution**:
  + Added missing environment variables in the Vercel dashboard.
  + Verified that the id in the URL matches the id in Sanity.
  + Added logging to debug data fetching in production.