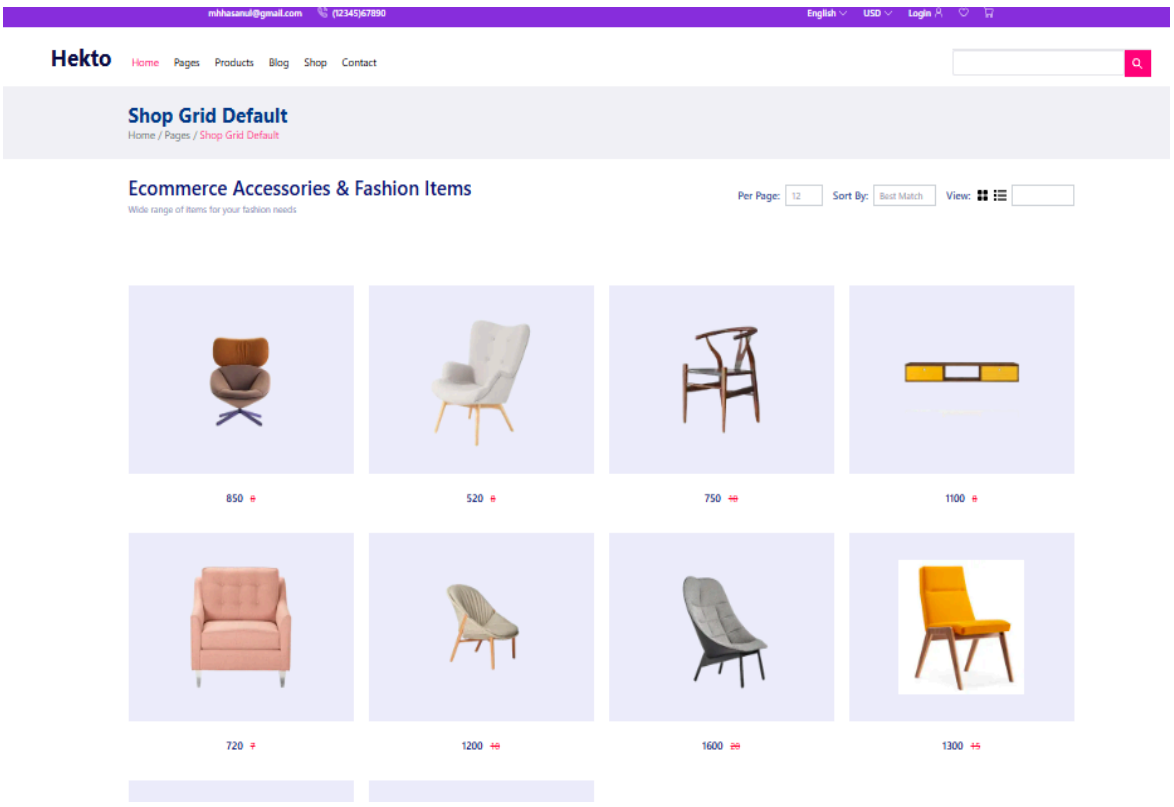
Day 4 - Dynamic Frontend Components - [General E-Commerce]

NAME: Mahnoor Ansari

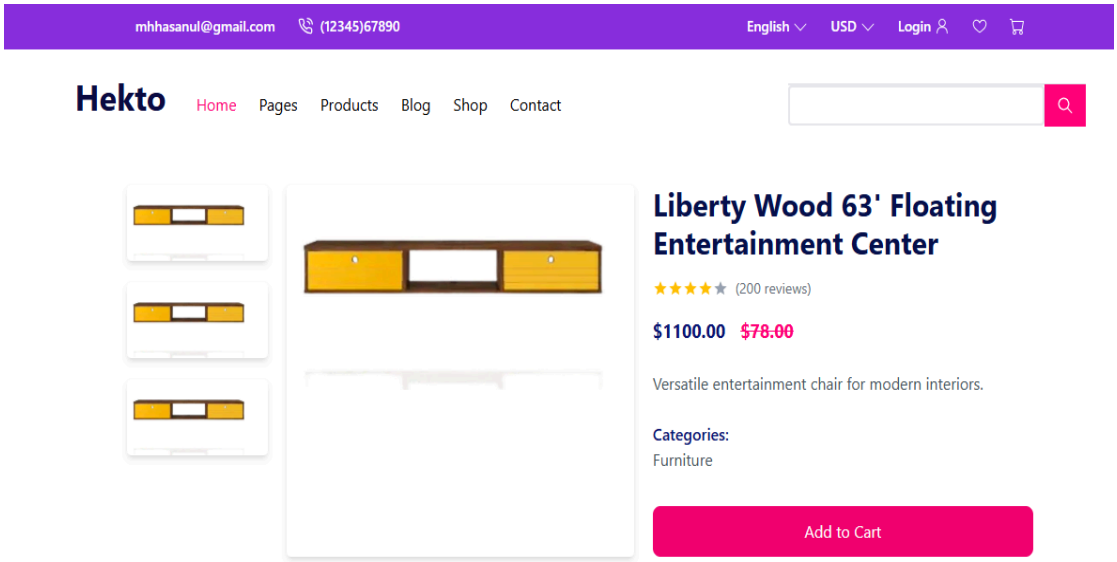
1. Functional Deliverables:

1. **The product listing page with dynamic data:**
    - The product listing page dynamically displays items fetched from the database.
    - It shows product details like name, price, and description.
    - Users can browse and interact with the products in real time.



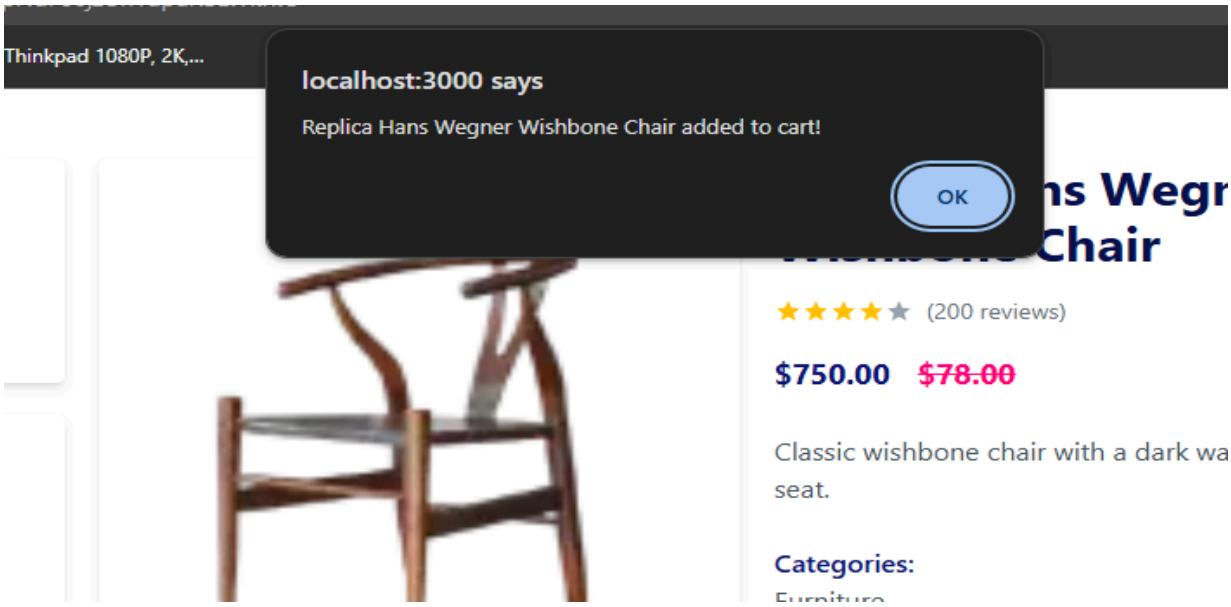
2.Individual product detail pages with accurate routing and data rendering:

- Each product has its own detail page with accurate routing.
- The page dynamically fetches and displays all product details like name, price, description, and images.
- Users can easily navigate to these pages by clicking on a product.
- The data is rendered in real-time for a seamless user experience.



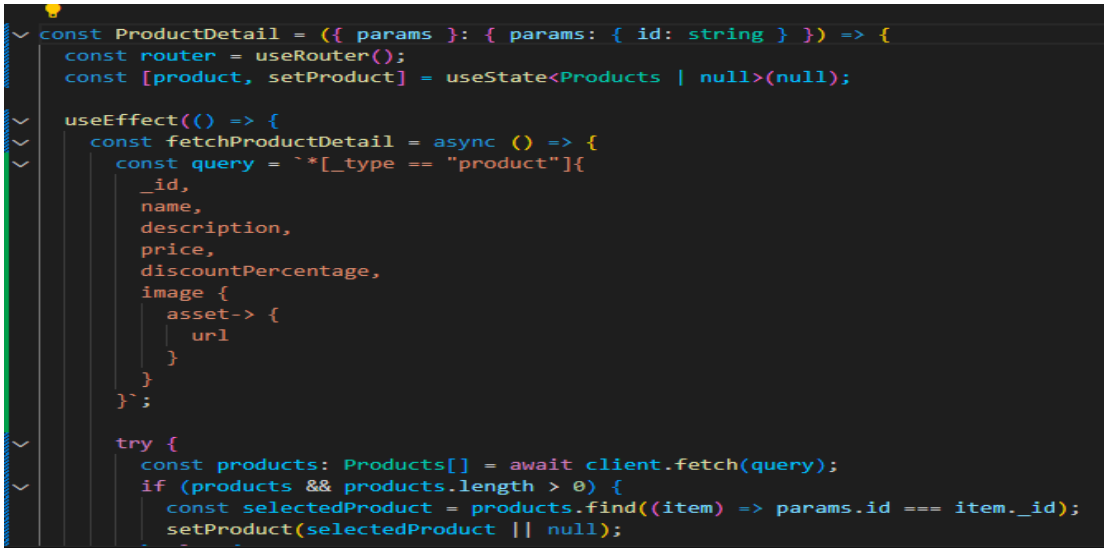
Notifications Component:

- The Notifications Component displays alerts or messages to users.
- It shows updates like success, errors, or warnings in real-time.
- Users can quickly see important information through pop-ups or banners.



Scripts or logic for API integration and dynamic routing:

- Scripts handle API integration to fetch and send data between the frontend and backend.
- Dynamic routing ensures each page is generated based on the product or data being accessed.
- This allows users to view specific content, like product details, by navigating to unique URLs.
- The logic ensures smooth interaction and accurate data display.



```
const ProductDetail = ({ params }: { params: { id: string } }) => {
  const router = useRouter();
  const [product, setProduct] = useState<Products | null>(null);

  useEffect(() => {
    const fetchProductDetail = async () => {
      const query = `*[_type == "product"]{
        _id,
        name,
        description,
        price,
        discountPercentage,
        image {
          asset-> {
            url
          }
        }
      }`;

      try {
        const products: Products[] = await client.fetch(query);
        if (products && products.length > 0) {
          const selectedProduct = products.find((item) => params.id === item._id);
          setProduct(selectedProduct || null);
```

Documentation:

1.Steps taken to build and integrate components:

- The components by defining their purpose and features.
- The code for each component with proper structure and styling.
- Test the components individually to ensure they work as expected.
- Integrate the components into the application by importing and placing them in the right locations.
- Continuously test and debug the integrated components for a smooth user experience.

2.Challenges faced and solutions implemented:

- Faced challenges in fetching data from the API, which was resolved by debugging the API endpoints.
  Encountered layout issues in the design, fixed by adjusting Tailwind CSS and testing responsiveness.
- Experienced errors in routing, solved by reviewing dynamic routing logic and correcting paths.
- Had trouble managing state across components, addressed by implementing context or state management tools.

        - Fixed performance issues by optimizing API calls and reducing unnecessary re-renders.

3.Best practices followed during development:

- Followed proper folder structure for better project organization.
- Tested components regularly to ensure they worked as expected.
- Used responsive design techniques to make the application work on all devices.

NOTE:

This document will be updated later as needed.
