Hackathon - Day 4 Documentation: Detailed Overview of Dynamic Components and Functionalities

Introduction

On Day 4 of the hackathon, the focus was on implementing dynamic components and enhancing the overall functionality of our application. This document provides a comprehensive overview of the components developed, their functionalities, and how they interact dynamically with user input and backend systems.

Dynamic Components Overview

1. Header Component

- **Location**: `components/header.tsx`
- **Description**: The `Header` component provides navigation and dynamic updates for wishlist and cart items.
- **Key Features**:
    - Dynamic cart badge: Displays the total number of items in the cart.
    - Wishlist icon: Changes to red when an item is added to the wishlist.

```tsx
"use client";
import React from "react";
import { CiSearch } from "react-icons/ci";
import { MdOutlineShoppingCart } from "react-icons/md";
import Link from "next/link";
import { useSelector } from "react-redux";
import { RootState } from "../Cart/redux/store";
import { FaRegHeart } from "react-icons/fa";
const Header = () => {
  // Accessing the state using RootState
  const cartItems = useSelector((state: RootState) => state.cart.items.length);

  return (
    <div className="relative max-w-full bg-white font-satoshi">
      {/* Header Section */}
      <header
        className="w-full h-[70px] flex items-center justify-between px-4 md:px-8 fixed top-0 left-0 z-50
      shadow-md bg-white"
      >
        {/* Left Section: Search Icon */}
        <div className="hidden md:block">
          <CiSearch size={24} className="text-gray-600 cursor-pointer" />
        </div>

        {/* Center Section: Brand Name */}
        <h1 className="text-lg md:text-xl font-normal text-black text-center font-[clash]">
          Avian
        </h1>

        {/* Right Section: Icons */}
        <div className="flex items-center space-x-4">
          {/* Search Icon for Mobile */}
          <CiSearch
            size={24}
            className="text-gray-600 cursor-pointer md:hidden"
          />

          {/* Shopping Cart Icon with Badge */}
          <Link href="/Cart" className="relative">
            <MdOutlineShoppingCart
              size={24}
              className="text-gray-600 cursor-pointer"
            />
            {cartItems > 0 && (
              <span
                className="absolute top-[-5px] right-[-5px] bg-red-500 text-white text-xs w-5 h-5
              rounded-full flex items-center justify-center"
              >
                {cartItems}
              </span>
            )}
          </Link>

          {/* User heart Icon */}
          <Link href="/wishlist">
            <FaRegHeart size={24} className="text-gray-600 cursor-pointer" />
          </Link>
        </div>
      </header>

      {/* Divider Line */}
      <hr className="w-full border-b-2 border-gray-300 mt-[70px]" />

      {/* Navigation Links */}
      <nav className="flex flex-wrap justify-center gap-4 lg:gap-8 py-4">
        <Link
          href="/"
          className="text-[#726E8D] hover:text-black transition-all py-2"
        >
          Home
        </Link>
        <Link
          href="/About"
          className="text-[#726E8D] hover:text-black transition-all py-2"
        >
          About
        </Link>
        <Link
          href="/products"
          className="text-[#726E8D] hover:text-black transition-all py-2"
        >
          Products
        </Link>
        <Link
          href="/Cart"
          className="text-[#726E8D] hover:text-black transition-all py-2"
        >
          Cart
        </Link>
      </nav>
    </div>
  );
};

export default Header;

```

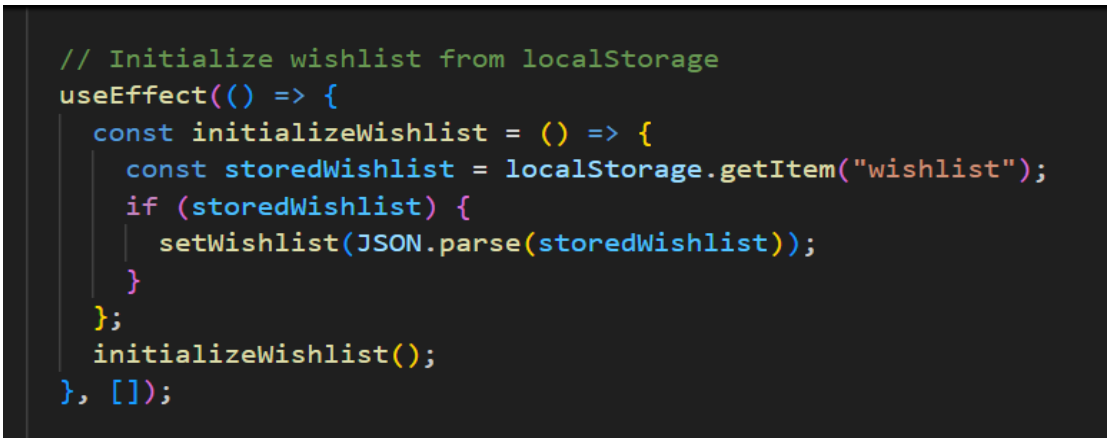
2. Product Page Component

- **Location**: `pages/[product].tsx`
- **Description**: Dynamically renders the product details based on the product ID.
- **Functionalities**:
    - Fetches product details from the backend using the product slug.
    - Handles adding items to the cart and wishlist.
    - Stores wishlist data in `localStorage` for persistence.
-

```
// Add/Remove Item from Wishlist
const handleAddToWishList = (id: string) => {
  if (wishlist.includes(id)) {
    const updatedWishlist = wishlist.filter((itemId) => itemId !== id);
    setWishlist(updatedWishlist);
    localStorage.setItem("wishlist", JSON.stringify(updatedWishlist));
  } else {
    const updatedWishlist = [...wishlist, id];
    setWishlist(updatedWishlist);
    localStorage.setItem("wishlist", JSON.stringify(updatedWishlist));
  }
};
```

3. Wishlist Route

- **Location**: Wishlist / page.tsx
- **Description**: Displays all items added to the wishlist.
- **Functionalities**:
    - Fetches wishlist data from `localStorage`.
    - Allows users to remove items from the wishlist dynamically.



```
// Initialize wishlist from localStorage
useEffect(() => {
  const initializeWishlist = () => {
    const storedWishlist = localStorage.getItem("wishlist");
    if (storedWishlist) {
      setWishlist(JSON.parse(storedWishlist));
    }
  };
  initializeWishlist();
}, []);
```

4. Cart Slice

- **Location**: `redux/cartslice.ts`
- **Description**: Manages cart state using Redux.
- **Functionalities**:
    - Adds items to the cart.
    - Removes items from the cart.
    - Updates item quantities dynamically.

```tsx
"use client"

import { createSlice, PayloadAction } from "@reduxjs/toolkit";

export interface CartItem {
  id: number | string;
  title: string;
  price: number | string;
  image: string;
  name: string;
  description:string
}
interface CartState {
  items: CartItem[];
}

const initialState: CartState = {
  items: [],
};

const cartSlice = createSlice({
  name: "cart",
  initialState,
  reducers: {
    // Remove item by ID
    remove(state, action: PayloadAction<number>) {
      state.items = state.items.filter((item) => item.id !== action.payload);
    },
    // Add item to the cart
    add(state, action: PayloadAction<CartItem>) {
      state.items.push(action.payload);
    },
  },
});

export const { remove, add } = cartSlice.actions;

export default cartSlice.reducer;

```

Functionalities Implemented

1. Dynamic Badge Update

- **Description**: Automatically updates cart and wishlist badges in the `Header` component based on the state.

- **Implementation**:
  - Used `useSelector` to listen for changes in the Redux store.
  - Re-rendered the badges dynamically whenever the state changed.

2. Persistent Wishlist

- **Description**: Wishlist data is stored in `localStorage` to ensure it persists across page reloads.
- **Steps**:
  - On component mount, fetch wishlist data from `localStorage`.
  - Update `localStorage` whenever the wishlist state changes.

3. Dynamic Product Details

- **Description**: Dynamically renders product details based on the product ID passed in the URL.
- **Steps**:
  - Fetched product details from Sanity CMS using the slug parameter.
  - Displayed data dynamically in the product details component.

4. Interactive Wishlist Button

- **Description**: Changes the heart icon to red when an item is added to the wishlist.
- **Steps**:
  - Used a conditional class to change the icon color dynamically.

Conclusion

Day 4 focused on building a robust and dynamic user experience through components that adapt to user interactions. By leveraging tools like Redux, `localStorage`, and conditional rendering, the application now delivers a highly interactive and seamless experience.