Low Level Design

ECOMMERCE WEBSITE DAISY DEALS 🎀

Written By	SNEHA KUMARI
Document Version	0.3
Last Revised Date	21-06-2024

Document Control

Change Record:

Version	Date	Author	Comments
0.1	16 – JUNE-2024	SNEHA	Introduction & Architecture defined
0.2	20 – JUNE-2024	SNEHA	Architecture & Architecture Description appended and updated
0.3	20 – JUNE-2024	SNEHA	Unit Test Cases defined and appended

Contents

1. Introduction

1.1. What is a Low-Level Design Document?

The goal of a Low-Level Design (LLD) document is to provide the internal logical design of the actual program code for the Daisy Deals e-commerce application. This document details the component structure, methods, and interactions within the application to guide programmers in writing the code.

1.2. Scope

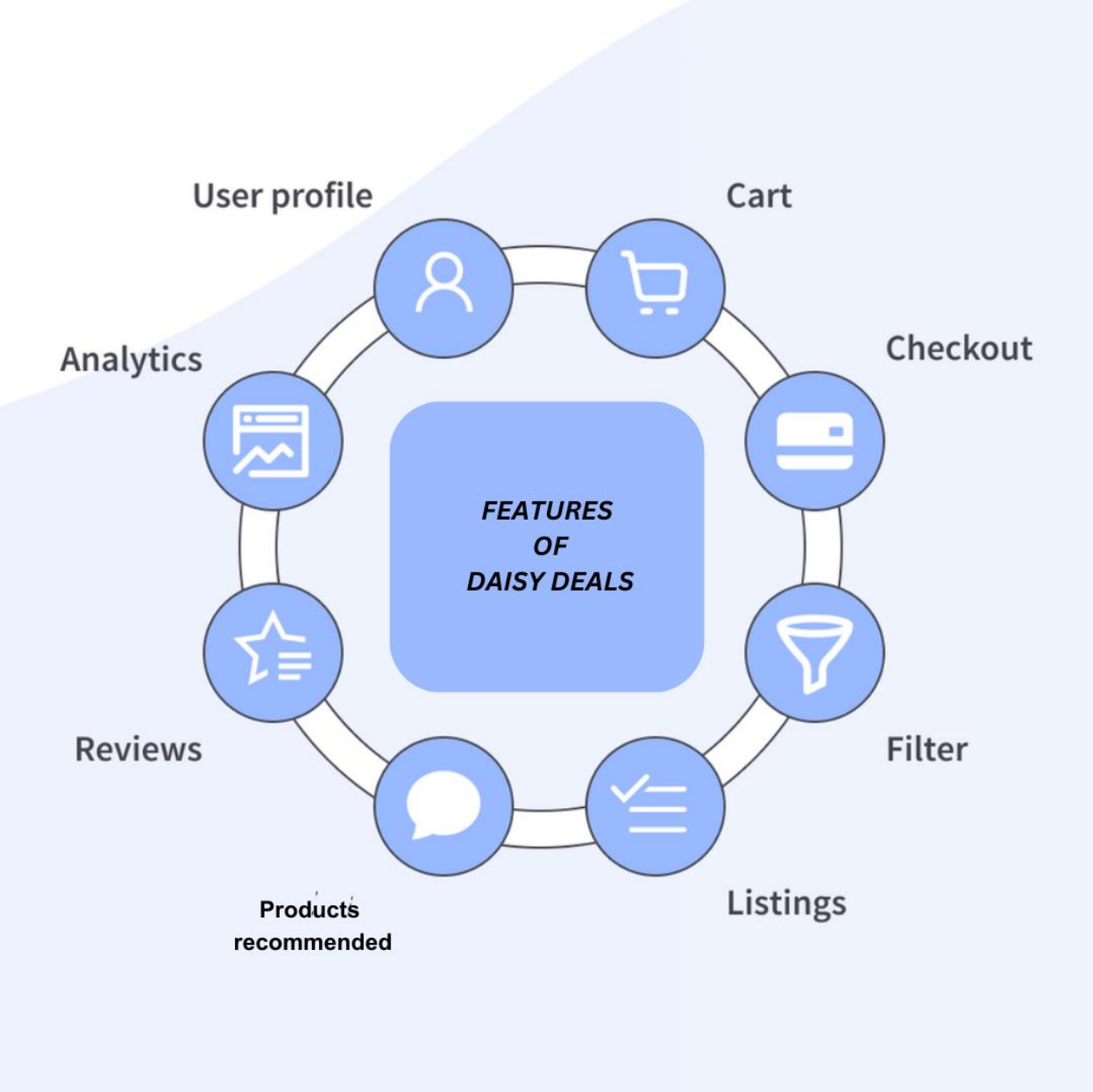
The LLD focuses on the frontend component of the Daisy Deals application, developed using JavaScript, React, and related technologies. It outlines the detailed design of user interface components, state management, routing, API integration, and styling, ensuring a cohesive and efficient implementation.

2. Architecture

Overview

The frontend architecture of Daisy Deals follows a component-based structure using React. Key architectural components include:

- **Component Hierarchy**: Organizing UI components in a hierarchical manner.
- **State Management**: Using Redux for global state management.
- **Routing**: Using React Router for navigation.
- **API Integration**: Using Axios for API calls.
- **Styling**: Using CSS Modules and styled-components for styling.



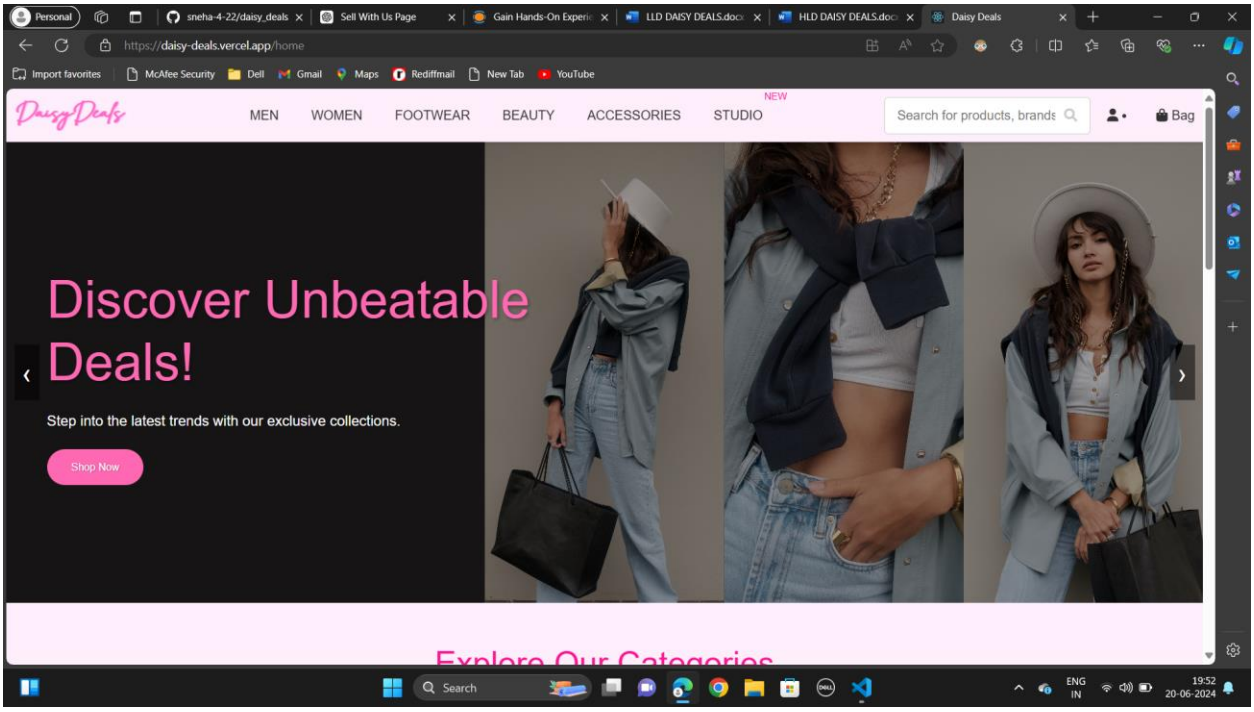
DEMO LINK - https://daisy-deals.vercel.app/

GITHUB - sneha-4-22/daisy_deals (github.com)

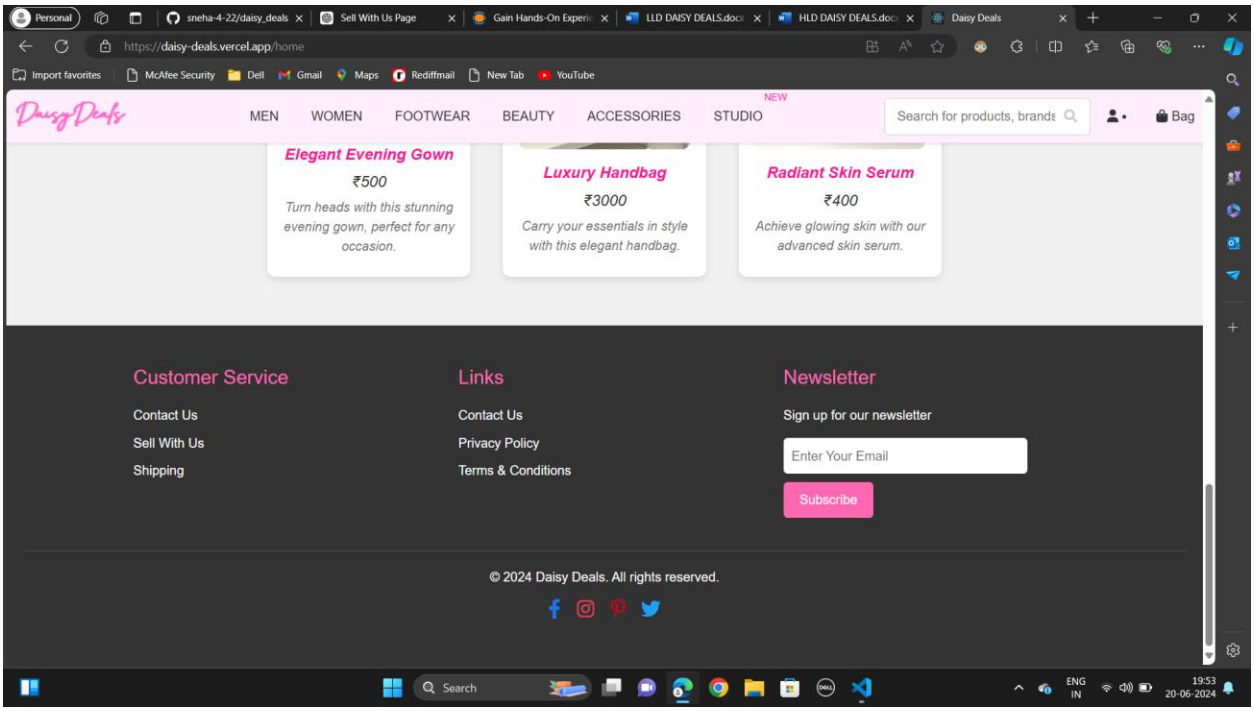
3. Architecture Description

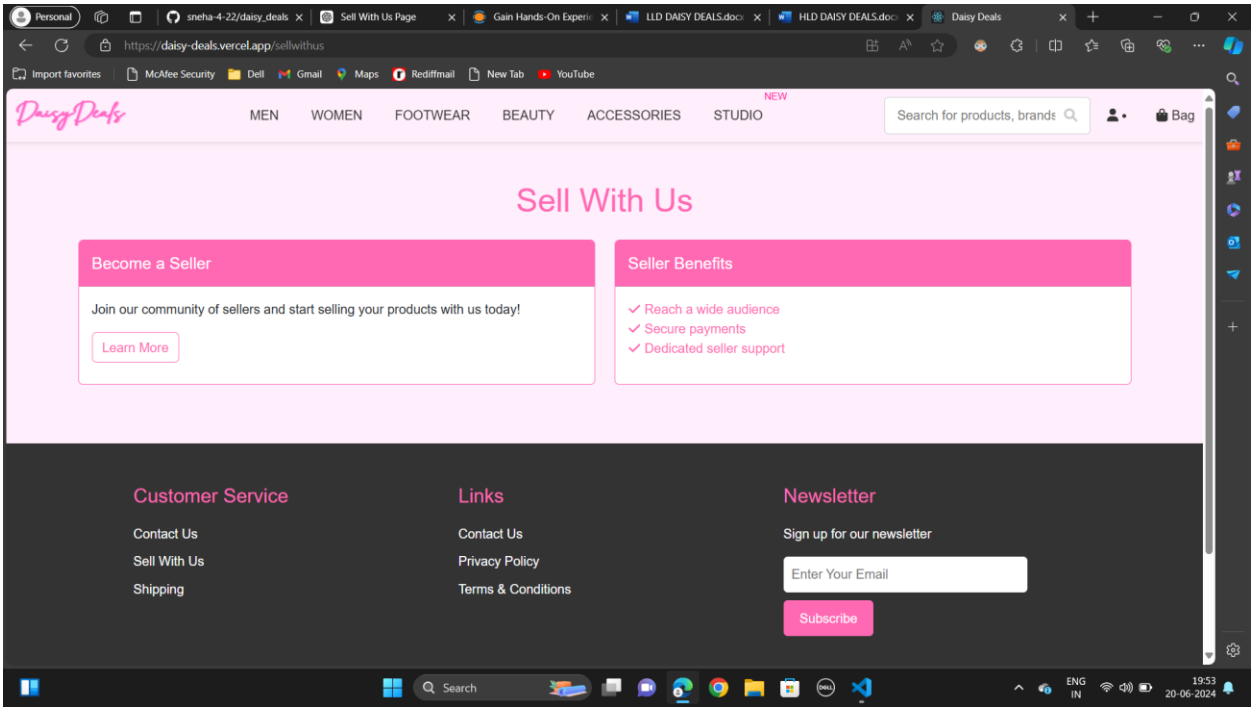
3.1. Component Description

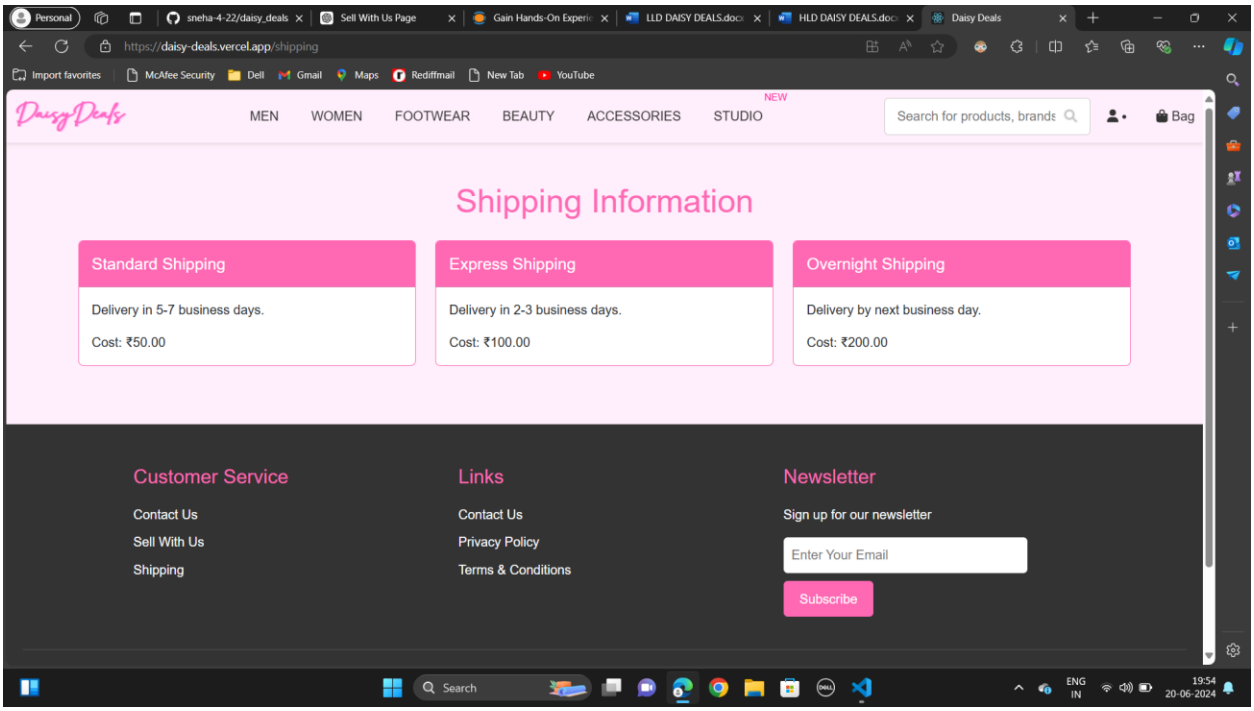
5. **App Component**
    - Entry point of the application.
    - Sets up routing and global context providers.
6. **Header Component**
    - Displays the logo, navigation links, and user actions (login, cart).



7. **Footer Component**
    - Contains links to company information, social media, and other relevant links.

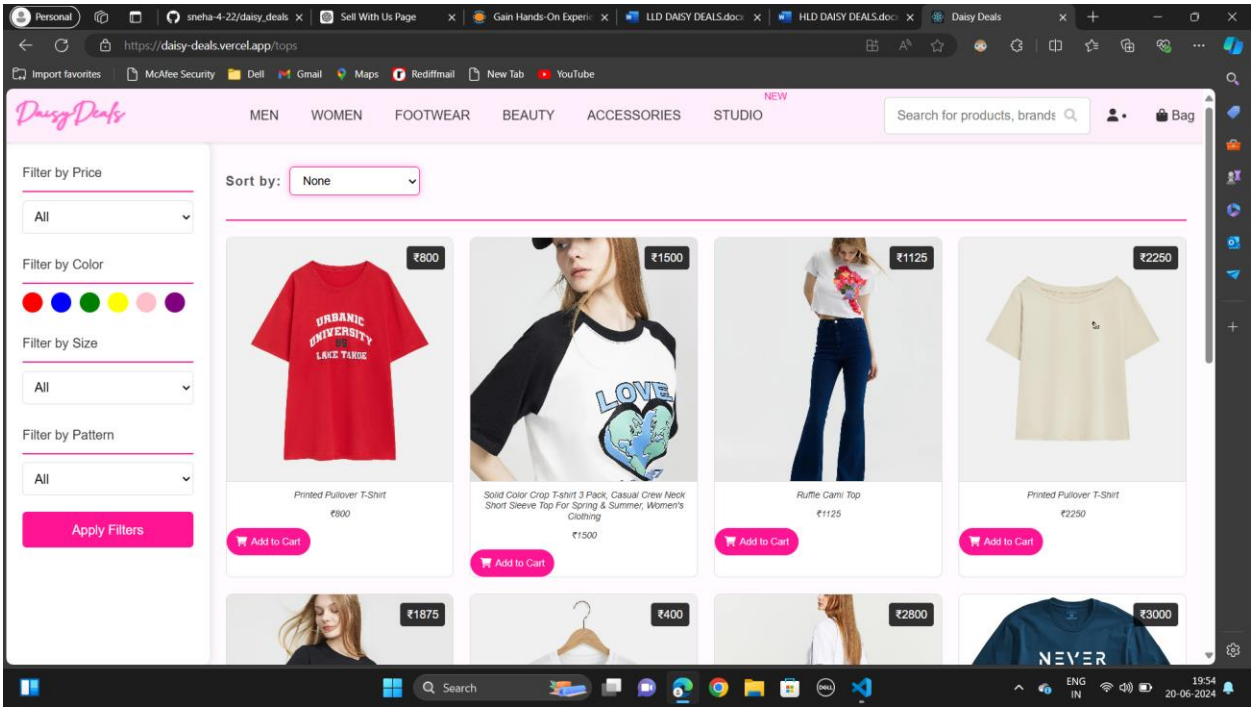

Daisy Deals
MEN
WOMEN
FOOTWEAR
BEAUTY
ACCESSORIES
STUDIO
NEW
Search for products, brands
Bag
Elegant Evening Gown
₹500
Turn heads with this stunning evening gown, perfect for any occasion.
Luxury Handbag
₹3000
Carry your essentials in style with this elegant handbag.
Radiant Skin Serum
₹400
Achieve glowing skin with our advanced skin serum.
Customer Service
Contact Us
Sell With Us
Shipping
Links
Contact Us
Privacy Policy
Terms & Conditions
Newsletter
Sign up for our newsletter
Enter Your Email
Subscribe
© 2024 Daisy Deals. All rights reserved.


Daisy Deals
MEN
WOMEN
FOOTWEAR
BEAUTY
ACCESSORIES
STUDIO
NEW
Search for products, brands
Bag
Sell With Us
Become a Seller
Join our community of sellers and start selling your products with us today!
Learn More
Seller Benefits
Reach a wide audience
Secure payments
Dedicated seller support
Customer Service
Contact Us
Sell With Us
Shipping
Links
Contact Us
Privacy Policy
Terms & Conditions
Newsletter
Sign up for our newsletter
Enter Your Email
Subscribe



8. HomePage Component

- Displays featured products and categories.



9. ProductList Component

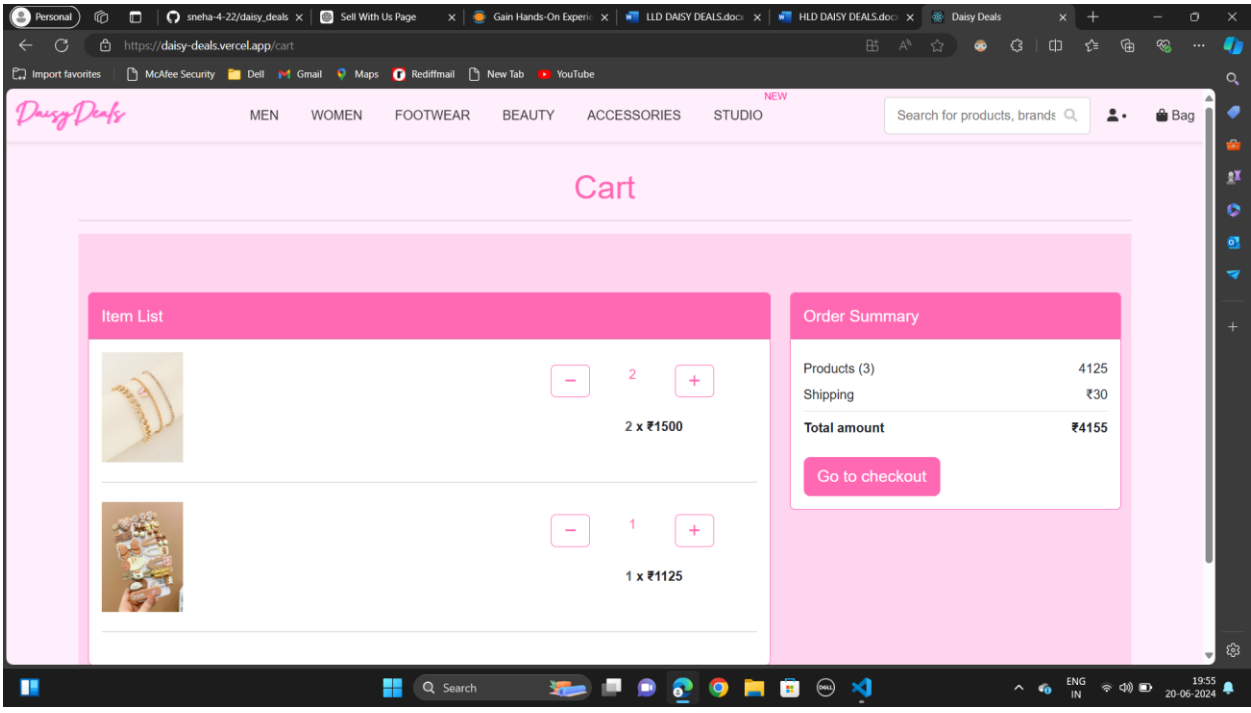
- Displays a list of products based on category or search query.

10. ProductDetail Component

- Shows detailed information about a single product.

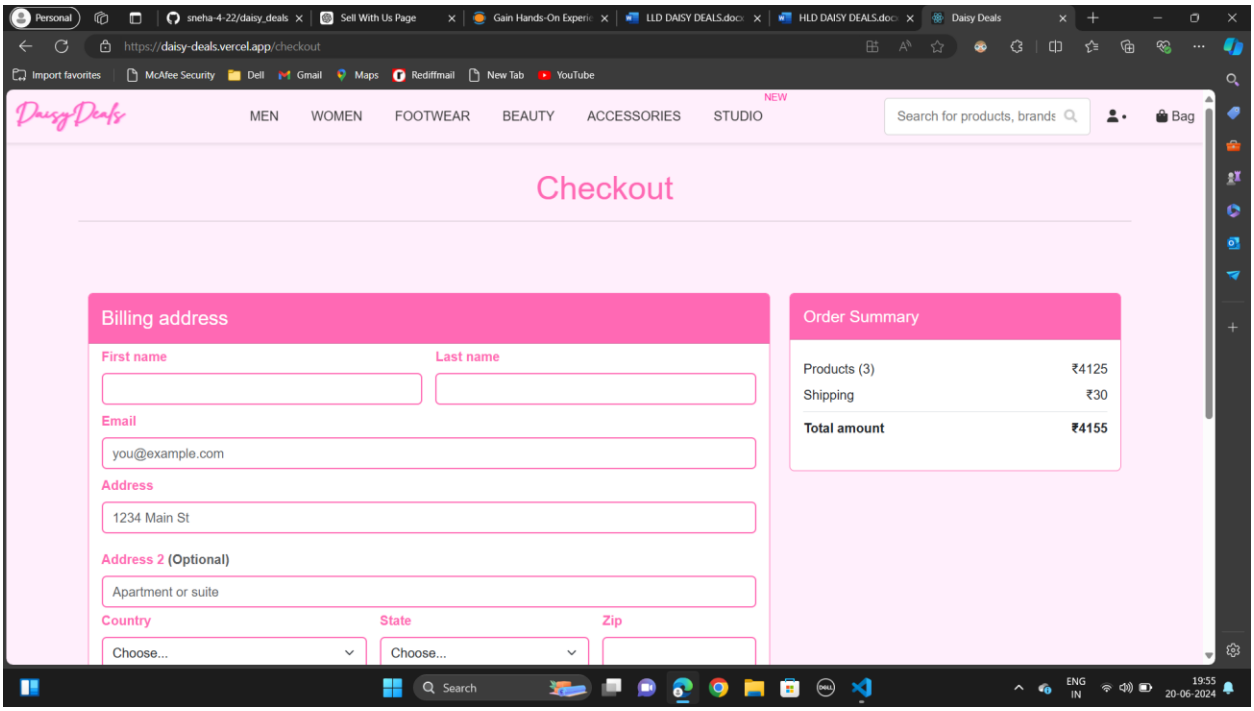
11. Cart Component

- Displays products added to the cart with options to modify quantities or remove items.

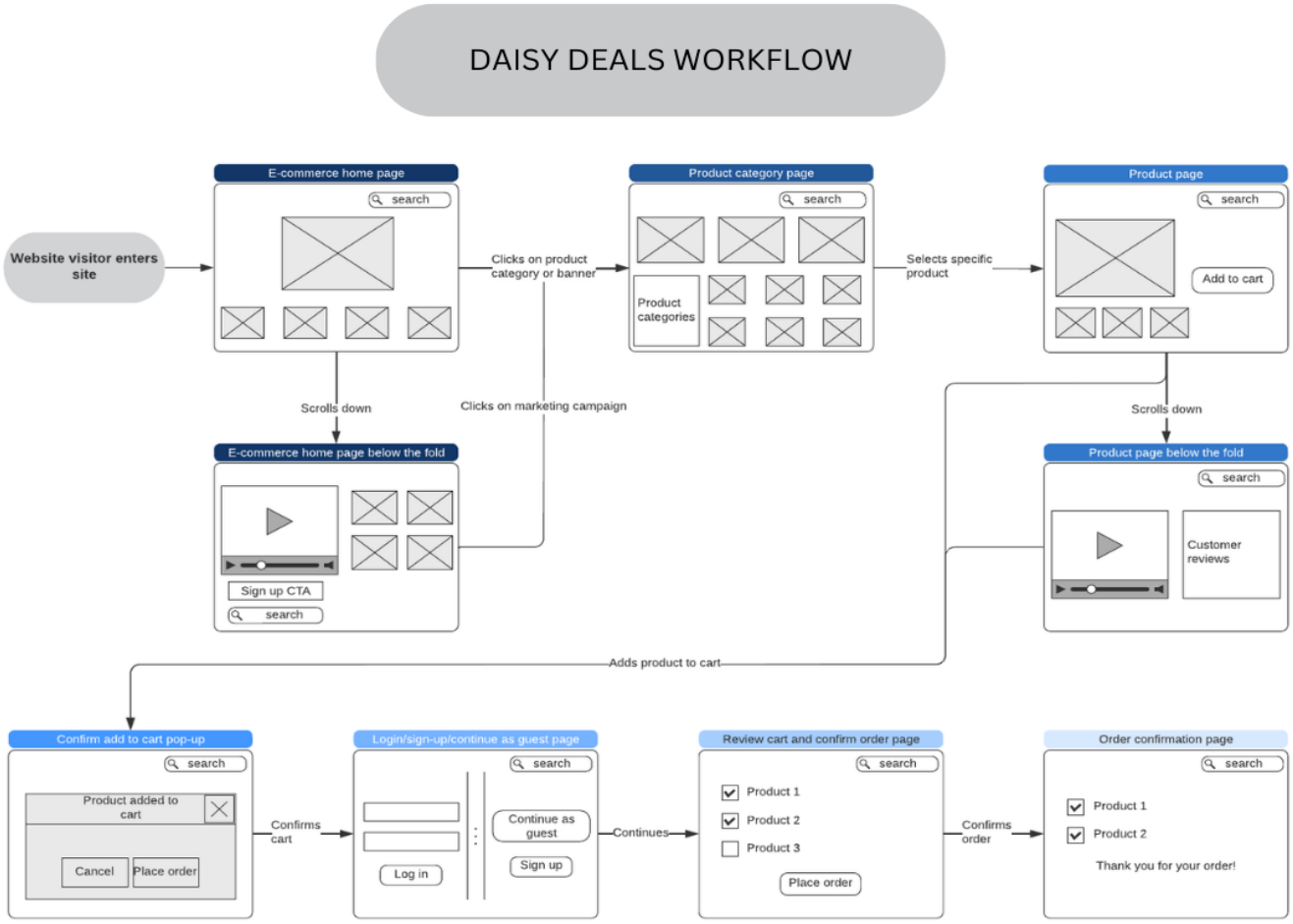


12. Checkout Component

- Collects shipping and payment information to complete the purchase.


Personal
sneha-4-22/daisy_deals
Sell With Us Page
Gain Hands-On Experie
LLD DAISY DEALS.docx
HLD DAISY DEALS.docx
Daisy Deals
https://daisy-deals.vercel.app/checkout
Import favorites
McAfee Security
Dell
Gmail
Maps
Rediffmail
New Tab
YouTube
DaisyDeals
MEN
WOMEN
FOOTWEAR
BEAUTY
ACCESSORIES
STUDIO
NEW
Search for products, brands
Bag
Checkout
Billing address
First name
Last name
Email
you@example.com
Address
1234 Main St
Address 2 (Optional)
Apartment or suite
Country
Choose...
State
Choose...
Zip
Order Summary
Products (3)
₹4125
Shipping
₹30
Total amount
₹4155
Search
ENG
IN
19:55
20-06-2024

USER I/O WORKFLOW



3.2. User Interface Components

13. Button Component

- Reusable button with props for text, click handlers, and styles.

14. ProductCard Component

- Displays product image, name, price, and an add-to-cart button.

15. **SearchBar Component**
    - Allows users to search for products by name or category.
16. **CategoryMenu Component**
    - Displays product categories for navigation.

3.3. State Management

- **Redux Store**
  - Centralized store to manage global state (cart, user information, product data).
- **Reducers**
  - CartReducer: Manages cart state (add/remove products, update quantities).
  - ProductReducer: Manages product list and details state.
- **Actions**
  - Cart Actions: addProductToCart, removeProductFromCart, updateProductQuantity.
  - Product Actions: fetchProducts, fetchProductDetails.

3.4. Routing

- **React Router Configuration**
  - `BrowserRouter` for wrapping the app.
  - `Route` components for mapping URL paths to components.

- Routes:
    - `/`: HomePage
    - `/products`: ProductList
    - `/product/:id`: ProductDetail
    - `/cart`: Cart
    - `/checkout`: Checkout

3.5. API Integration

- **Axios Setup**
    - Create an Axios instance for API calls with base URL and default headers.
- **API Calls**
    - fetchProducts: GET request to retrieve product list.
    - fetchProductDetails: GET request to retrieve product details by ID.

3.6. Styling

- **CSS Modules**
    - Scoped CSS to prevent class name collisions.
- **Styled-Components**
    - Utilized for dynamic and reusable styles.
- **Global Styles**
    - Define global styles and theme using CSS variables or a theme provider.

4. Unit Test Cases

4.1. App Component

- **Test Case 1**: Verify that the App component renders without crashing.
    - **Pre-Requisite**: Application setup is complete.
    - **Expected Result**: App component renders successfully.

4.2. Header Component

- **Test Case 1**: Verify that the Header displays navigation links.
    - **Pre-Requisite**: Header component is rendered.
    - **Expected Result**: Navigation links are displayed correctly.
- **Test Case 2**: Verify that the Header shows the cart icon.
    - **Pre-Requisite**: Header component is rendered.
    - **Expected Result**: Cart icon is displayed correctly.

4.3. HomePage Component

- **Test Case 1**: Verify that the HomePage displays featured products.
    - **Pre-Requisite**: HomePage component is rendered.
    - **Expected Result**: Featured products are displayed on the homepage.

4.4. ProductList Component

- **Test Case 1**: Verify that the ProductList displays a list of products.

    - **Pre-Requisite**: ProductList component is rendered.
    - **Expected Result**: A list of products is displayed.
- **Test Case 2**: Verify that the ProductList handles empty product list.
    - **Pre-Requisite**: ProductList component is rendered with an empty product list.
    - **Expected Result**: Appropriate message is displayed when no products are available.

4.5. ProductDetail Component

- **Test Case 1**: Verify that the ProductDetail displays product information.
    - **Pre-Requisite**: ProductDetail component is rendered with product data.
    - **Expected Result**: Product information is displayed correctly.
- **Test Case 2**: Verify that the add-to-cart button works.
    - **Pre-Requisite**: ProductDetail component is rendered with product data.
    - **Expected Result**: Clicking the add-to-cart button adds the product to the cart.

4.6. Cart Component

- **Test Case 1**: Verify that the Cart displays added products.
    - **Pre-Requisite**: Products are added to the cart.
    - **Expected Result**: Cart component displays the added products.
- **Test Case 2**: Verify that the Cart allows quantity updates.

    - **Pre-Requisite**: Cart component is rendered with products.
    - **Expected Result**: User can update the quantity of products in the cart.

4.7. Checkout Component

- **Test Case 1**: Verify that the Checkout collects shipping information.
    - **Pre-Requisite**: Checkout component is rendered.
    - **Expected Result**: User can input shipping information.
- **Test Case 2**: Verify that the Checkout processes the order.
    - **Pre-Requisite**: Checkout component is rendered with valid input.
    - **Expected Result**: Order is processed and confirmation is displayed.