Low Level Design (HLD)

“E-Commerce Application Clone”

Document Version Control

• Version	• Date	• Description	• Author
• **1.0**	• 20-Oct-2024	• **Introduction** • **Frontend Tools Breakdown**	• **Rajan Kumar**
• **1.1**	• 21-Oct-2024	• **Flow Chart** • **Detailed Component Interaction** • **State Management Flow (Frontend)** • **React Component Hierarchy**	• **Rajan Kumar**

Contents

Document Version Control.....2

1. Introduction.....4-5

1.1 What is Low Level Design Document?.....4

1.2 Scope.....4

1.3 Project Introduction.....5

1.4 Purpose of the Project.....5

2. Frontend Tools Breakdown.....6-7

2.1 React Component.....6

2.2 State Management.....6

2.3 Routing.....7

3. Flow Chart.....8

4. Detailed Component Interaction 9

5. State Management Flow (Frontend).....10

6. React Component Hierarchy.....11

1. Introduction

1.1 What is Low Level Design Document?

The goal of the Low-level design document (LLDD) is to give the internal logic design of the actual program code for the Heart Disease Diagnostic Analysis dashboard. LLDD describes the class diagrams with the methods and relations between classes and programs specs. It describes the modules so that the programmer can directly code the program from the document.

1.2 What is Scope?

Low-level design (LLD) is a component-level design process that follows a step-by-step refinement process. The process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work.

This LLD focuses on the frontend structure and flow of the E-Commerce Application Clone, utilizing only frontend tools (HTML, CSS, JavaScript, React). The core functionalities are managed via state and interactions on the client side.

1.3 Project Introduction:

E-Commerce Application Clone

The **E-Commerce Application Clone** is a client-side web application designed to mimic the core functionalities of an online shopping platform. The main objective of this project is to provide users with an interactive and seamless shopping experience, allowing them to browse through a collection of products, view individual product details, add products to their cart, and proceed to checkout for completing their purchases.

Key Features:

1. **Product Listing and Browsing:**
    - Users can view a homepage displaying a wide variety of products.
    - Products are neatly presented with essential information such as product name, price, and a thumbnail image.
2. **Category-Based Browsing:**
    - Users can filter products by category (e.g., electronics, fashion, etc.) to view a specific collection of items.
    - This allows easier navigation and product discovery based on preferences.
3. **Product Details Page:**

- Clicking on a product will take the user to a dedicated Product Details page, where they can explore detailed information like product description, price, reviews, and images.

4. **Cart Functionality:**
    - Users can add products to their cart from both the Homepage and Product Details page.
    - The cart displays all the products selected, with options to modify the quantity or remove items.
    - The cart total price is dynamically updated based on user modifications.
5. **Checkout Process:**
    - The Checkout page collects the user's shipping and payment information.
    - Users can review their order details and complete the purchase with a confirmation.

1.4 Purpose of the Project:

The purpose of this E-Commerce Application Clone is to simulate the experience of an online shopping platform with a focus on front-end development. It provides the essential features that enable users to engage with a product catalog, manage their shopping cart, and complete purchases.

2. Frontend Tools Breakdown

2.1 React Components

The application will be built with React.js to handle state and dynamic interactions across the application.

- **Header (Navbar) Component:**
  - Contains links to the Home, Categories, Cart, and Checkout pages.
  - Displays the cart count (using a state or context API).
- **Homepage Component:**
  - Fetches and displays the collection of products.
  - Each product will have an "Add to Cart" button to manage cart interactions.
  - It will use props to pass product data down to ProductCard components.
- **ProductCard Component:**
  - Displays individual product details such as image, name, and price on the homepage.
  - Will trigger "Add to Cart" actions when clicked.
- **Product Details Component:**
  - Displays more detailed information about a selected product.
  - It will be navigated to from the ProductCard on the Homepage or Category page.
- **CategoryPage Component:**
  - Displays products from a specific category.
  - Similar to the Homepage layout but will show products based on category selection (props or URL params).
- **Cart Component:**
  - Shows products that have been added to the cart.
  - Allows the user to modify the quantity or remove products.
  - Calculates and displays the total price.
- **Checkout Component:**
  - Displays a form to enter shipping and payment details.
  - Confirms the order and shows an order summary.
- **Footer Component:**
  - Simple footer with basic links (Privacy Policy, Contact Us, etc.).

2.2 State Management

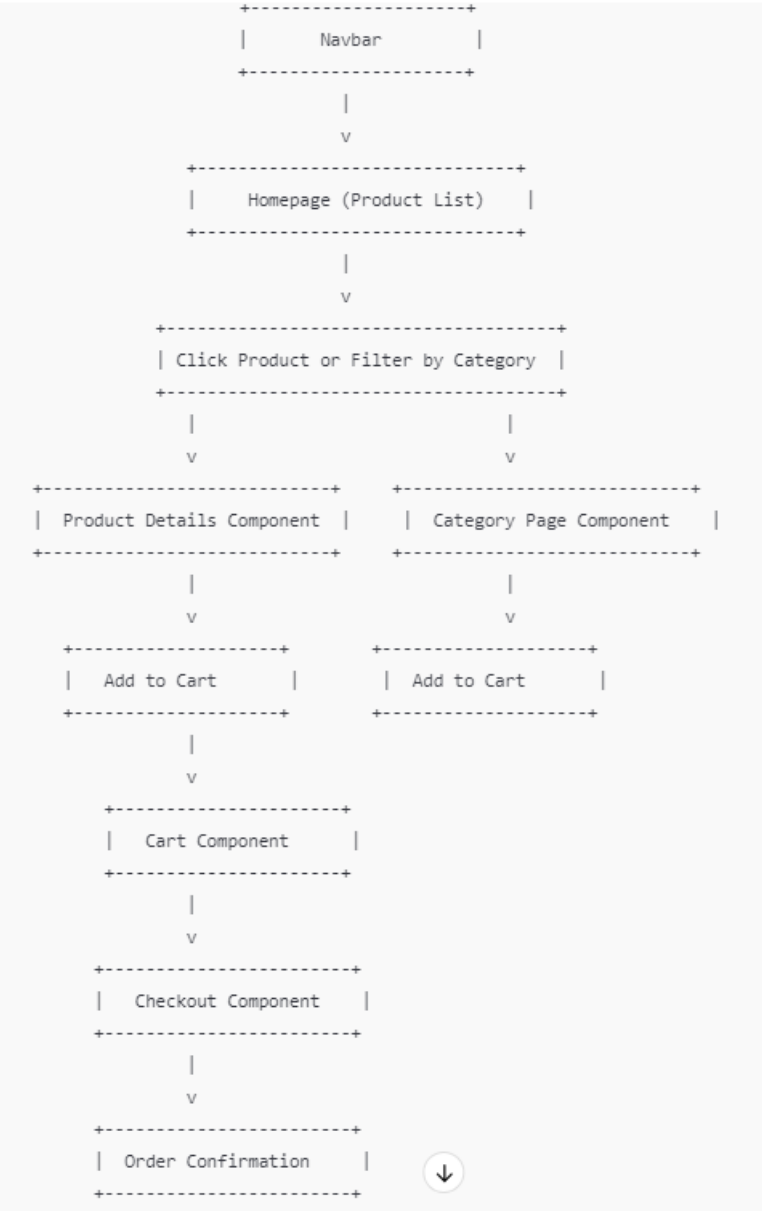
- **React's useState and useContext Hooks:**
    - useState will be used to manage product lists, cart items, and UI updates.
    - useContext will be used to globally manage the state of the cart (cart items, total price) across different components.

2.3 Routing

- **React Router:**
    - Routing will be implemented using React Router to navigate between the Homepage, Product Details, Category Page, Cart, and Checkout.

3. Flow Chart

Below is the frontend flow chart, showing how data and user actions flow across different components.



4. Detailed Component Interaction

1. **Homepage**
    - Fetches the product list and renders it using the ProductCard component.
    - **State:**
        - Products are stored in the component's state using useState().
        - When a user clicks on a product, it navigates to the Product Details page.
2. **Category Page**
    - Displays products filtered by a selected category.
    - Products can be added to the cart from this page.
    - Uses React Router to pass the category as a parameter.
3. **Product Details**
    - Displays detailed information about a specific product.
    - An "Add to Cart" button allows users to add the product directly from this page.
    - **State:**
        - Individual product details are fetched from the product list based on the selected product's ID.
4. **Cart Page**
    - Shows the products that the user has added to the cart.
    - Allows for quantity modification or product removal.
    - **State:**
        - The cart state is managed globally using useContext so that the cart can be accessed from any component.
5. **Checkout Page**
    - Contains a form for shipping details and payment information.
    - Confirms the final order and clears the cart once the order is placed.
    - **State:**
        - Captures user input for shipping and payment via controlled form elements.

5. State Management Flow (Frontend)

State in this frontend design will be managed primarily by React's useState and useContext. Here's a breakdown of how the state flows across different components:

1. **Global State (useContext)**
    - **Cart Context:**
        - Tracks the items in the cart and the total price.
        - Accessible from any component, such as Navbar, Cart, and Checkout.
        - Actions:
            - Add item to cart
            - Remove item from cart
            - Update quantity
2. **Local State (useState)**
    - **Product List (Homepage/CategoryPage):**
        - Fetches and stores the list of products to display.
        - Passed down as props to the ProductCard component.
    - **Product Details:**
        - Stores detailed information about the selected product when the user clicks on it.
    - **Checkout Form:**
        - Stores user input (shipping address, payment information) before confirming the order.

6. **React Component Hierarchy**

```
App.js
|
|-- Navbar.js (Cart Count Display)
|
|-- Routes (React Router)
    |
    |-- / (Homepage.js)
    |     |-- ProductList.js
    |     |   |-- ProductCard.js (Add to Cart)
    |
    |-- /category/:id (CategoryPage.js)
    |     |-- ProductList.js (Filtered by Category)
    |
    |-- /product/:id (ProductDetails.js)
    |     |-- Add to Cart Button
    |
    |-- /cart (Cart.js)
    |     |-- CartItem.js (Modify Quantity/Remove)
    |
    |-- /checkout (Checkout.js)
          |-- Shipping & Payment Form
```

Summary of Frontend Flow:

1. **Homepage:**
   Fetches and displays products. Users can navigate to product details or add products to the cart.
2. **Category Page:**
   Displays products from a selected category.
3. **Product Details Page:**
   Displays detailed information about a single product and allows adding to the cart.
4. **Cart Page:**
   Displays the products in the cart and allows modifying the cart's contents.
5. **Checkout Page:**
   Final step where users enter shipping/payment details and place the order