**Project Documentation**

**“Mercato“**

**Created by:**

* **Mina Ashraf**
* **Hassan Mohsen**
* **Table of Contents**

1. [Project Overview](#project-overview)
2. [Technologies Used](#technologies-used)
3. [Installation](#installation)
4. [Folder Structure](#folder-structure)
5. [Context Providers](#context-providers)
   * [UserContext](#usercontext)
   * [CartContext](#cartcontext)
   * [WishlistContext](#wishlistcontext)
6. [Components](#components)
   * [Layout](#layout)
   * [Home](#home)
   * [Cart](#cart)
   * [Categories](#categories)
   * [Login](#login)
   * [Register](#register)
   * [Products](#products)
   * [ProductDetails](#productdetails)
   * [Wishlist](#wishlist)
7. [Styling](#styling)
8. [Routing](#routing)
9. [Error Handling](#error-handling)
10. [Testing](#testing)
11. [Future Improvements](#future-improvements)

* **Project Overview**

This project is an e-commerce application that allows users to browse products, manage a shopping cart, and maintain a wishlist. Users can log in, register, and perform various operations related to shopping.

**Technologies Used**

* **React**: Frontend framework for building the user interface.
* **Vite**: Build tool for fast development and production builds.
* **Tailwind CSS**: Utility-first CSS framework for styling.
* **Axios**: Library for making HTTP requests.
* **React Router**: Library for routing in React applications.
* **React Hot Toast**: Library for displaying toast notifications.
* **Font Awesome**: Icons for UI components.

**Installation:**

To run the project locally, follow these steps:

1. Clone the repository:

bash

Copy code

git clone <repository-url>

cd <project-folder>

1. Install dependencies:

bash

Copy code

npm install

1. Start the development server:

bash

Copy code

npm run dev

1. Open your browser and navigate to http://localhost:3000.

**Folder Structure:**

**bash**

**Copy code**

**src/**

**│**

**├── component/ # Components of the application**

**│ ├── Cart/**

**│ ├── Categories/**

**│ ├── Home/**

**│ ├── Login/**

**│ ├── Notfound/**

**│ ├── Products/**

**│ ├── Wishlist/**

**│ └── ...**

**│**

**├── Context/ # Context providers for state management**

**│ ├── UserContext.jsx**

**│ ├── CartContext.jsx**

**│ └── WishlistContext.jsx**

**│**

**├── App.jsx # Main application component**

**├── main.jsx # Entry point of the application**

**└── index.css # Global styles and Tailwind CSS**

## Context Providers:

### UserContext

* **Purpose**: Manages user authentication state.
* **Key Methods**:
  + setUserLogin: Updates the user login state.

### CartContext

* **Purpose**: Handles shopping cart operations.
* **Key Methods**:
  + addProductToCart(productId): Adds a product to the cart.
  + getLoggedUserCart(): Fetches the user's cart.
  + updateCartProductQuantity(productId, newCount): Updates product quantity.
  + deleteCartItem(productId): Removes a product from the cart.
  + clearCart(): Clears the entire cart.
  + checkout(cartId, url, formdata): Initiates checkout.

### WishlistContext

* **Purpose**: Manages wishlist operations.
* **Key Methods**:
  + addProductToWishlist(productId): Adds a product to the wishlist.
  + deleteProductFromWishlist(productId): Removes a product from the wishlist.
  + getWishlist(): Fetches the user's wishlist.

## Components:

### Layout

* Main layout component that wraps around other components.

### Home

* The homepage displaying featured products and categories.

### Cart

* Component for managing the shopping cart and displaying cart items.

### Categories

* Displays different product categories available in the store.

### Login

* Allows users to log in to their accounts.

### Register

* Allows new users to create an account.

### Products

* Displays a list of products available for purchase.

### ProductDetails

* Shows detailed information about a selected product.

### Wishlist

* Displays the user's saved wishlist items.

## Styling

* The application uses Tailwind CSS for utility-based styling. Custom styles can be added in the index.css file as needed.

## Routing

* React Router is used for navigation between different components. Protected routes are implemented to restrict access to certain components for logged-in users.

## Error Handling

* Error handling is implemented in the API calls, logging errors to the console and providing user feedback via toast notifications.

## Testing

* Consider implementing unit tests for components and integration tests for context providers using React Testing Library.

## Screenshots:

## 

## Future Improvements

* Implement unit tests and end-to-end tests.
* Enhance accessibility features.
* Optimize performance by using memorization where necessary.
* Implement SEO best practices for improved visibility.