Fashionista - E-Commerce Website Documentation

Project Overview

Project Name: Fashionista - Fashion E-Commerce Store **Technology:** React.js (Frontend Only) **Developer:** Vikas Sharma **Experience Level:** 3.5+ Years Frontend Development

---

Table of Contents

---

1. Project Introduction

Fashionista is a fully functional fashion e-commerce website built using React.js. This project demonstrates advanced frontend development skills including state management, component architecture, routing, and responsive design.

Purpose

- Showcase frontend development expertise
- Demonstrate React.js proficiency
- Display understanding of e-commerce user flows
- Exhibit responsive design implementation

Key Highlights

- Pure React.js implementation (no backend required)
- Complete shopping experience from browsing to checkout
- Persistent cart and wishlist using localStorage
- Mobile-first responsive design

---

2. Technology Stack

Technology	Version	Purpose
React.js	19.x	UI Library
React Router DOM	7.x	Client-side Routing
React Icons	5.x	Icon Components
Vite	7.x	Build Tool & Dev Server
CSS3	-	Styling
LocalStorage	-	Data Persistence

Why These Technologies?

React.js: Industry-standard library for building user interfaces with component-based architecture.

React Router: Enables single-page application (SPA) navigation without page reloads.

Vite: Modern build tool offering fast development server and optimized production builds.

CSS Variables: Maintainable theming system with consistent design tokens.

---

3. Project Structure

```
fashionista-ecommerce/
├── public/
│   └── vite.svg
├── src/
│   ├── components/
│   │   ├── common/
│   │   │   ├── Navbar.jsx          # Navigation bar
│   │   │   └── Footer.jsx          # Site footer
│   │   ├── product/
│   │   │   ├── ProductCard.jsx     # Product display card
│   │   │   └── ProductFilters.jsx  # Filter sidebar
│   │   ├── cart/
│   │   │   └── CartItem.jsx        # Cart item row
│   │   └── auth/
│   │       └── (future components)
│   ├── pages/
│   │   ├── Home.jsx                # Landing page
│   │   ├── Products.jsx            # Product listing
│   │   ├── ProductDetail.jsx       # Single product view
│   │   ├── Cart.jsx                # Shopping cart
│   │   ├── Wishlist.jsx            # Saved items
│   │   ├── Checkout.jsx            # Checkout process
│   │   ├── Auth.jsx                # Login/Register
│   │   └── Orders.jsx              # Order history
│   ├── context/
│   │   └── ShopContext.jsx         # Global state management
│   ├── data/
│   │   └── products.js             # Product catalog data
│   ├── App.jsx                     # Root component
│   ├── App.css                     # Global styles
│   ├── index.css                   # Base styles
│   └── main.jsx                    # Entry point
├── index.html
├── package.json
├── vite.config.js
└── PROJECT_DOCUMENTATION.md
```

Directory Explanation

Directory	Purpose
`components/`	Reusable UI components organized by feature
`pages/`	Full page components mapped to routes

`context/`	React Context for global state management
`data/`	Static data files (products, categories)

---

4. Features Overview

4.1 Product Browsing

- **Product Listing:** Grid display of all products
- **Category Filtering:** Men, Women, Shoes, Accessories
- **Price Range Filter:** Min/Max price selection
- **Sort Options:** Featured, Price (Low-High, High-Low), Rating
- **Search Functionality:** Search by product name or description

4.2 Product Details

- **Image Display:** High-quality product images
- **Size Selection:** Multiple size options per product
- **Color Selection:** Color variants
- **Quantity Selector:** Increment/decrement quantity
- **Add to Cart:** With selected options
- **Add to Wishlist:** Save for later
- **Product Reviews:** Rating and review section
- **Related Products:** Similar item suggestions

4.3 Shopping Cart

- **View Cart Items:** List of added products
- **Update Quantity:** Modify item quantities
- **Remove Items:** Delete from cart
- **Price Calculation:** Subtotal, shipping, tax, total
- **Free Shipping:** Orders over $100
- **Clear Cart:** Remove all items

4.4 Wishlist

- **Save Products:** Heart icon to save
- **View Saved Items:** Dedicated wishlist page
- **Move to Cart:** Easy transfer to cart
- **Remove Items:** Unsave products

4.5 User Authentication

- **Registration:** Create new account
- **Login:** Sign in existing users
- **Logout:** End session
- **Persistent Session:** Remember logged-in state

4.6 Checkout Process

- **Step 1 - Shipping:** Address information
- **Step 2 - Payment:** Card details (demo)
- **Step 3 - Review:** Order confirmation
- **Order Placement:** Complete purchase

4.7 Order Management

- **Order History:** View past orders
- **Order Details:** Items, shipping, total
- **Order Status:** Confirmed, Shipped, etc.

---

5. Component Architecture

Component Hierarchy

```
App
├── ShopProvider (Context)
│   └── Router
│       ├── Navbar
│       ├── Routes
│       │   ├── Home
│       │   │   ├── Hero Section
│       │   │   ├── Features
│       │   │   ├── Categories
│       │   │   ├── ProductCard (multiple)
│       │   │   └── Promo Banner
│       │   ├── Products
│       │   │   ├── ProductFilters
│       │   │   └── ProductCard (multiple)
│       │   ├── ProductDetail
│       │   │   └── ProductCard (related)
│       │   ├── Cart
│       │   │   └── CartItem (multiple)
│       │   ├── Wishlist
│       │   │   └── ProductCard (multiple)
│       │   ├── Checkout
│       │   ├── Auth
│       │   └── Orders
│       └── Footer
```

Component Types

Type	Examples	Purpose
Layout	Navbar, Footer	Consistent page structure
Page	Home, Products, Cart	Route-level components
Feature	ProductCard, CartItem	Specific functionality
UI	Buttons, Inputs	Basic UI elements

---

6. State Management

Context API with useReducer

The application uses React's Context API combined with useReducer for centralized state management.

State Structure

```
const initialState = {
  products: [...],           // Product catalog
  cart: [],                  // Shopping cart items
  wishlist: [],              // Saved products
  user: null,                // Current user
  orders: [],                // Order history
  filters: {                 // Active filters
    category: 'all',
    priceRange: [0, 500],
    sortBy: 'featured',
    searchQuery: ''
  }
};
```

Available Actions

Action	Description
`ADD_TO_CART`	Add product with size/color to cart
`REMOVE_FROM_CART`	Remove item from cart
`UPDATE_CART_QUANTITY`	Change item quantity
`CLEAR_CART`	Empty the cart
`TOGGLE_WISHLIST`	Add/remove from wishlist
`SET_FILTERS`	Update filter settings
`LOGIN`	Set user data
`LOGOUT`	Clear user data
`ADD_ORDER`	Create new order
`ADD_REVIEW`	Add product review

Context Hook Usage

```
// In any component
import { useShop } from '../context/ShopContext';

const Component = () => {
  const {
    cart,
    wishlist,
    dispatch,
    getCartTotal,
    getCartCount,
    isInWishlist
  } = useShop();

  // Use state and dispatch actions
};
```

Data Persistence

Cart, wishlist, user, and orders are automatically saved to localStorage:

```
useEffect(() => {
  localStorage.setItem('cart', JSON.stringify(state.cart));
}, [state.cart]);
```

7. Routing Configuration

Route Definitions

```
<Routes>
  <Route path="/" element={<Home />} />
  <Route path="/products" element={<Products />} />
  <Route path="/product/:id" element={<ProductDetail />} />
  <Route path="/cart" element={<Cart />} />
  <Route path="/wishlist" element={<Wishlist />} />
  <Route path="/checkout" element={<Checkout />} />
  <Route path="/auth" element={<Auth />} />
  <Route path="/orders" element={<Orders />} />
</Routes>
```

Route Parameters

Route	Parameter	Description
`/product/:id`	`id`	Product ID for detail page
`/products?category=men`	`category`	Filter by category

8. Key Components Explained

8.1 ShopContext.jsx

Purpose: Global state management for the entire application.

Key Features:

- useReducer for complex state updates
- Helper functions (getCartTotal, getFilteredProducts)
- LocalStorage persistence
- Context Provider pattern

```
// Core reducer function
function shopReducer(state, action) {
  switch (action.type) {
    case 'ADD_TO_CART':
      // Handle adding items with variants
    case 'TOGGLE_WISHLIST':
      // Toggle item in wishlist
    // ... more cases
  }
}
```

8.2 ProductCard.jsx

Purpose: Display individual product in grid layouts.

Features:

- Product image with hover effects
- Quick add to cart/wishlist buttons

- Price display with discount calculation
- Rating stars
- Link to product detail

```
const ProductCard = ({ product }) => {
  const { dispatch, isInWishlist } = useShop();

  const handleQuickAdd = (e) => {
    e.preventDefault();
    dispatch({
      type: 'ADD_TO_CART',
      payload: { ...product, size: product.sizes[0], ... }
    });
  };

  return (
    <Link to={`/product/${product.id}`}>
      {/* Card content */}
    </Link>
  );
};
```

8.3 ProductFilters.jsx

Purpose: Sidebar filters for product listing.

Features:

- Category selection buttons
- Price range inputs
- Sort dropdown
- Real-time filtering

8.4 Checkout.jsx

Purpose: Multi-step checkout process.

Features:

- Progress indicator (3 steps)
- Form validation
- Order summary sidebar
- Order creation on submit

```
const [step, setStep] = useState(1);

// Step 1: Shipping Information
// Step 2: Payment Details
// Step 3: Review & Confirm
```

---

9. Styling Approach

CSS Architecture

CSS Variables for Theming:

```
:root {
  --primary: #2563eb;
  --primary-dark: #1d4ed8;
  --secondary: #f97316;
  --dark: #1f2937;
  --light: #f9fafb;
  --gray: #6b7280;
  --success: #10b981;
  --danger: #ef4444;
  --shadow: 0 4px 6px -1px rgba(0, 0, 0, 0.1);
  --radius: 8px;
  --transition: all 0.3s ease;
}
```

Responsive Breakpoints

Breakpoint	Target Devices
1024px	Tablets landscape
768px	Tablets portrait
480px	Mobile phones

```
@media (max-width: 768px) {
  .products-grid {
    grid-template-columns: repeat(2, 1fr);
  }
  .nav-links {
    /* Mobile menu styles */
  }
}
```

Key Styling Patterns

1. **Grid Layouts:** CSS Grid for product grids
2. **Flexbox:** Navigation and component alignment
3. **Transitions:** Smooth hover effects
4. **Box Shadows:** Depth and hierarchy
5. **Border Radius:** Modern, rounded design

---

10. How to Run

Prerequisites

- Node.js (v18 or higher)
- npm (v9 or higher)

Installation Steps

```
# 1. Navigate to project directory
cd Personal-Portfolio-

# 2. Install dependencies
npm install
```

```
# 3. Start development server
npm run dev

# 4. Open in browser
# http://localhost:5173
```

Build for Production

```
# Create production build
npm run build

# Preview production build
npm run preview
```

Project Scripts

Script	Command	Description
dev	`npm run dev`	Start dev server
build	`npm run build`	Production build
preview	`npm run preview`	Preview build
lint	`npm run lint`	Run ESLint

---

11. Future Enhancements

Potential Improvements

1. **Backend Integration**
    - RESTful API connection
    - Real database for products
    - User authentication with JWT
2. **Payment Gateway**
    - Stripe/Razorpay integration
    - Real payment processing
3. **Additional Features**
    - Product image gallery
    - Size guide modal
    - Recently viewed products
    - Product comparison
    - Email notifications
4. **Performance**
    - Image lazy loading
    - Code splitting
    - Service worker for offline
5. **Testing**
    - Unit tests with Jest

    - E2E tests with Cypress

Skills Demonstrated

This project showcases the following frontend development skills:

React.js Expertise

- Functional Components
- React Hooks (useState, useEffect, useReducer, useContext, useRef)
- Custom Hooks
- Context API
- Component Composition
- Props and State Management

Modern JavaScript

- ES6+ Features
- Array Methods (map, filter, reduce)
- Destructuring
- Spread Operator
- Template Literals
- Arrow Functions

CSS/Styling

- CSS Variables
- Flexbox Layout
- CSS Grid
- Responsive Design
- Media Queries
- Transitions & Animations

Development Practices

- Component-based Architecture
- Separation of Concerns
- Clean Code Principles
- File Organization
- Git Version Control

Contact Information

Developer: Vikas Sharma **Experience:** 3.5+ Years Frontend Development **Specialization:** React.js, JavaScript, TypeScript **GitHub:** https://github.com/Vikas85109 **LinkedIn:** https://www.linkedin.com/in/vikas-sharma-4bb27418a/

This documentation was created for the Fashionista E-Commerce project to demonstrate frontend development capabilities and serve as a reference for project understanding.