Low Level Design Document

Introduction

This Low Level Design (LLD) document outlines the implementation details for **RetailX - Personalized E-Commerce Experience**. RetailX is a next-generation e-commerce frontend built with React, Redux Toolkit, and Tailwind CSS, providing personalized product recommendations, advanced search, dynamic filtering, user authentication, and seamless checkout with payment API integration.

---

1. System Components

Component	Description	Key Responsibilities
ProductCatalog	Displays products, supports filtering/search	Fetch, filter, and render product list
RecommendationEngine	Shows personalized recommendations	Fetch and display user-specific suggestions
SearchBar	Advanced product search	Handle input, trigger search actions
AuthModule	User authentication (login/signup/logout)	Manage auth state, token storage
Cart/Checkout	Cart management and payment integration	Add/remove items, process payments
Redux Store	Global state management	Store user, cart, product, and UI state
UI Components	Reusable UI elements (buttons, cards, modals, etc.)	Consistent, responsive UI

---

2. Class/Interface Overview

Name	Type	Relationships/Usage	Key Methods/Attributes
`Product`	Interface	Used in ProductCatalog, Cart, Recommendations	`id`, `name`, `price`, `tags`, `imageUrl`
`User`	Interface	Used in AuthModule, Recommendations	`id`, `name`, `email`, `token`
`CartItem`	Interface	Used in Cart/Checkout	`productId`, `quantity`
`AppDispatch`	Type	Redux actions	`dispatch(action)`
`RootState`	Type	Redux state	`user`, `cart`, `products`, `ui`

Example: Product Interface (TypeScript)

```
interface Product {
  id: string;
```

```
  name: string;
  price: number;
  tags: string[];
  imageUrl: string;
}
```

---

3. Data Structure Overview

Model	Fields	Notes
Product	id, name, price, tags, imageUrl, description	Used for display/filtering
User	id, name, email, token	Auth & personalization
CartItem	productId, quantity	Cart state
Order	id, userId, items, total, status, paymentInfo	Checkout & payment

---

4. Algorithms / Logic

Personalized Recommendation Flow (Pseudocode):

```
onUserLogin(userId):
  userProfile = fetchUserProfile(userId)
  history = fetchUserPurchaseHistory(userId)
  recommendations = getRecommendations(userProfile, history)
  display(recommendations)
```

Dynamic Product Filtering:

```
onFilterChange(filters):
  filteredProducts = products.filter(p => matches(p, filters))
  updateProductCatalog(filteredProducts)
```

Checkout Flow:

```
onCheckout(cart, paymentDetails):
  if validateCart(cart) and validatePayment(paymentDetails):
    response = callPaymentAPI(paymentDetails)
    if response.success:
      createOrder(cart, user)
      showSuccess()
    else:
      showPaymentError()
```

---

5. Error Handling

Scenario	Handling Approach
API/network failure	Show error toast, retry option
Invalid user input (auth/forms)	Inline validation, error messages
Payment failure	Display error, allow retry
Unauthorized access	Redirect to login, clear sensitive state
Data fetch error	Show fallback UI, log error

End of Document