Product Requirements & Specification Document

Project Name

RetailX - Personalized E-Commerce Experience

Description

RetailX is a next-generation e-commerce frontend delivering personalized product recommendations, advanced search, and seamless checkout. Built with React, Redux Toolkit, and Tailwind CSS, RetailX offers a performant, responsive, and futuristic user interface. Key features include dynamic product filtering, user authentication, and integration with payment APIs.

---

1. Goals & Objectives

Goal	Description
Personalization	Deliver tailored product recommendations per user
Advanced Search & Filtering	Enable fast, dynamic product discovery
Seamless Checkout	Provide a frictionless, secure checkout experience
Responsive, Futuristic UI	Ensure modern, mobile-first design with startup appeal
Integration	Connect with payment APIs and backend services

---

2. Core Features

Feature	Description
Personalized Recommendations	AI-driven suggestions based on user behavior and preferences
Advanced Search	Full-text, faceted search with autocomplete and typo tolerance
Dynamic Filtering	Real-time filtering by category, price, brand, rating, etc.
User Authentication	Sign up, login, password reset, OAuth (Google, Facebook)
Product Catalog	Responsive grid/list views, quick view, and product detail pages
Shopping Cart	Add/remove/update items, persistent cart state
Seamless Checkout	Multi-step checkout, address management, order summary, payment integration
Order History	View past orders, order details, and status tracking
Responsive Design	Optimized for mobile, tablet, and desktop
Accessibility	WCAG 2.1 compliance, keyboard navigation, ARIA roles

---

3. Technical Specifications

3.1 Frontend Stack

Technology	Purpose
React	UI framework
Redux Toolkit	State management
Tailwind CSS	Utility-first styling
TypeScript	Type safety
JavaScript	Core scripting
HTML/CSS	Markup and base styling

3.2 Architecture

- **Component-based** React structure
- **Redux slices** for user, products, cart, and UI state
- **API layer** for backend and payment integration
- **Responsive layouts** via Tailwind CSS
- **TypeScript** for all business logic and components

3.3 Integration

Service	Integration Method
Product API	REST/GraphQL
Auth API	JWT/OAuth2
Payment API	Stripe/PayPal SDK

---

4. User Flows

4.1 Browsing & Search

```
flowchart LR
    A[Home] --> B[Search/Filter]
    B --> C[Product List]
    C --> D[Product Detail]
    D --> E[Add to Cart]
```

4.2 Authentication

```
flowchart LR
    A[Landing] --> B[Login/Signup]
    B --> C[Authenticated Session]
```

4.3 Checkout

```
flowchart LR
    A[Cart] --> B[Address]
    B --> C[Payment]
    C --> D[Order Confirmation]
```

---

5. Non-Functional Requirements

Requirement	Specification
Performance	<2s page load, lazy loading, code splitting
Security	HTTPS, JWT, input validation, XSS/CSRF safe
Accessibility	WCAG 2.1 AA compliance
Scalability	Support 10k+ concurrent users
Responsiveness	Mobile-first, supports all major browsers
Maintainability	Modular, well-documented codebase

---

6. Acceptance Criteria

Feature	Criteria
Recommendations	Personalized list updates per user session
Search & Filtering	<500ms response, supports multiple filters
Authentication	Secure login, OAuth, password reset
Checkout	End-to-end flow, payment confirmation, error handling
Responsiveness	Renders correctly on mobile, tablet, desktop
Accessibility	Passes automated and manual accessibility tests

---

7. Milestones & Timeline

Milestone	Deliverable	Timeline
UI/UX Design	Wireframes, prototypes	Week 1-2
Core Architecture	Project setup, routing, state	Week 3
Product Catalog/Search	Product listing, search/filter	Week 4-5
Auth & User Profile	Auth flows, profile management	Week 6
Cart & Checkout	Cart, checkout, payment integration	Week 7-8
Personalization	Recommendation engine integration	Week 9

QA & Launch	Testing, bug fixes, deployment	Week 10

---

8. Risks & Mitigations

Risk	Mitigation
API Instability	Mock APIs, error handling, retry logic
Payment Integration Delays	Early integration, fallback payment options
Personalization Accuracy	A/B testing, user feedback loops
Performance Bottlenecks	Profiling, code splitting, caching

---

9. Appendix

- **Design References:** Figma links (to be provided)
- **API Documentation:** Swagger/OpenAPI (to be provided)
- **Brand Guidelines:** Futuristic, startup-inspired palette and typography

---

End of Document