Requirements Document

Functional Requirements Assessment

FR-001: User Registration

- **Description**: New users can register with a username (5-30 characters, alphanumeric and underscores only) and password (5-100 characters), with real-time validation, bcrypt hashing, and storage in MongoDB.
- **Status**: Adequately met
- **Details**: The registration page (register.html) implements enhanced validation with 5-character minimums, format restrictions, real-time feedback with color-coded inputs, and dynamic error messages. User data is securely hashed and stored in MongoDB, with duplicate username handling and redirection to the login page upon success.

FR-002: User Authentication

- **Description**: Registered users log in with validated credentials, generating a JWT token stored in localStorage, with real-time feedback and redirection to the shop page.
- **Status**: Adequately met
- **Details**: The login page (login.html) features 5-character minimum validation, real-time feedback, and enhanced error messaging. JWT tokens are generated and stored in localStorage, with successful redirection to shop.html upon valid authentication.

FR-003: Session Management

- **Description**: Protected pages check for a valid JWT token, redirecting unauthenticated users to the login page.
- **Status**: Adequately met
- **Details**: Shop.html, checkout.html, and confirmation.html enforce JWT validation, redirecting to login.html if no valid token is found. JWT middleware protects API endpoints, ensuring secure access control.

FR-004: User Logout

- **Description**: Authenticated users can log out, clearing the JWT token from localStorage and redirecting to the login page with a confirmation prompt.
- **Status**: Adequately met

- **Details**: The logout button on shop.html triggers a confirmation dialog, clears the JWT token, and redirects to login.html, ensuring secure session termination.

FR-005: Product Display

- **Description**: Authenticated users view a product catalog in a grid layout with name, price, description, category, and hover effects, loaded from MongoDB via GET /products API.
- **Status**: Adequately met
- **Details**: Shop.html displays a dynamic product grid with all required details, loaded from MongoDB. Hover effects enhance UX, and the page is fully integrated with backend APIs.

FR-006: Shopping Cart Management

- **Description**: Authenticated users can add products to a persistent cart via POST /cart/add, with real-time counter updates and confirmation messages.
- **Status**: Adequately met
- **Details**: The cart system supports adding products with backend persistence, real-time updates to the cart counter, and confirmation messages. Quantity updates are handled appropriately.

FR-007: Cart Viewing

- **Description**: Users view cart contents with product name, quantity, unit price, total price, and cart subtotal.
- **Status**: Adequately met
- **Details**: The cart sidebar on shop.html displays all required details, including real-time updates and subtotal calculations, synchronized with MongoDB via GET /cart API.

FR-008: Cart Modification

- **Description**: Users can modify cart contents (quantity changes or item removal) with real-time updates and total recalculation.
- **Status**: Adequately met
- **Details**: Users can increase/decrease quantities (PUT /cart/update) or remove items (DELETE /cart/remove), with real-time updates to the cart display and totals. Zero-quantity items are removed automatically.

FR-009: Checkout Process

- **Description**: Users complete purchases via a checkout form with shipping information, order summary, and simulated payment, generating a unique order ID, storing the order in MongoDB, and clearing the cart.
- **Status**: Partially met
- **Details**: The checkout page (checkout.html) includes a refined interface with a city dropdown (20 US cities), phone number validation with (XXX) XXX-XXXX formatting, and real-time feedback. Orders are processed via POST /orders API, with confirmation displayed on confirmation.html, including order ID and cart clearing. However, the planned payment processing simulation was removed due to time constraints and implementation complexity, limiting the checkout to shipping information only.

FR-010: Order History

- **Description**: Authenticated users can view past orders with order ID, date, items, and totals, with detailed order views.
- **Status**: Not met
- **Details**: The order history page was planned but not implemented due to development focus on core checkout and cart functionality. This feature remains a future enhancement.

FR-011: Navigation System

- **Description**: Users experience consistent navigation across pages, maintaining authentication state.
- **Status**: Adequately met
- **Details**: All pages (Home.html, login.html, register.html, shop.html, checkout.html, confirmation.html) feature a consistent navigation menu, with proper routing and authentication state preservation.

FR-012: Data Persistence

- **Description**: The system stores user accounts, cart data, and order history in MongoDB, ensuring data integrity and security.
- **Status**: Partially met
- **Details**: User accounts and cart data are persistently stored in MongoDB with secure handling (bcrypt for passwords, JWT for authentication). Order data is stored via the POST /orders API. However, order history persistence is incomplete due to the unimplemented order history page (FR-010).

Planned Improvements

To address unmet or partially met requirements and enhance the application, the following improvements are proposed:

- **Implement Order History Page (FR-010)**: Develop an order history page allowing users to view past orders with details (order ID, date, items, totals). This will involve creating a new page (orders.html) and a GET /orders/user API endpoint to retrieve user-specific order data from MongoDB.
- **Simulated Payment Processing (FR-009)**: Reintroduce a simplified payment simulation for the checkout process, such as a mock payment form with basic validation (e.g., card number format), to complete the checkout experience without requiring real payment integration.
- **Expand City Dropdown Options**: Enhance the checkout form’s city dropdown to include more US cities or allow freeform input with validation to accommodate a broader user base.
- **Add Product Search and Filtering**: Implement search and category filter functionality on shop.html to improve product discovery, using GET /products API with query parameters.
- **Introduce User Profile Management**: Create a profile page where users can update account details (e.g., email, password), enhancing user control and engagement.
- **Email Notifications**: Integrate email confirmations for registration and order placement using a service like Nodemailer, improving user communication.
- **Improve Scalability**: Optimize MongoDB queries and consider server-side session management to replace localStorage for JWT tokens, addressing potential performance issues with increased traffic.
- **Multi-Language Support**: Add localization for non-English-speaking users, starting with Spanish, by implementing language toggle and translated validation messages.