E-Commerce User and Admin Guide

This guide provides instructions for users and administrators of the E-Commerce web application, based on the current project status as, it covers the complete end-to-end shopping flow, including user registration, login, product browsing, cart management, checkout, and order confirmation, as well as administrative tasks for managing the platform.

User Guide

1. Getting Started

Accessing the Application

- **URL**: Access the application at http://localhost:5000 (local development environment).
- **Supported Devices**: The application is fully responsive and works on desktops, tablets, and mobile devices.
- **Navigation**: The landing page (Home.html) provides navigation links to "About," "Login," and "Register."

System Requirements

- A modern web browser (e.g., Chrome, Firefox, Safari).
- Internet access for API communication with the backend server.

2. Account Management

Registration

1. Navigate to the registration page via the "Register" link on the landing page.
2. Enter a username (5–30 characters, alphanumeric and underscores only).
3. Enter a password (5–100 characters).
4. Real-time validation provides visual feedback:
    a. Green border for valid input.
    b. Red border with error messages for invalid input (e.g., username already exists, incorrect format).
5. Submit the form to create an account.

6. Upon success, you'll be redirected to the login page.

Login

1. Navigate to the login page via the "Login" link.
2. Enter your username and password (both require a minimum of 5 characters).
3. Real-time validation provides color-coded feedback and dynamic error messages.
4. A loading state appears during authentication.
5. Upon successful login, a JSON Web Token (JWT) is stored in localStorage, and you're redirected to the shop page (shop.html).

Logout

1. On the shop page, click the "Logout" button.
2. Confirm the logout in the prompt.
3. The JWT is removed from localStorage, and you're redirected to the login page.

Access Control

- Attempting to access protected pages (e.g., shop.html, checkout.html, confirmation.html) without a valid JWT redirects you to the login page.
- Toast notifications inform you of unauthorized access attempts.

3. Shopping Experience

Browsing Products

1. After logging in, the shop page (shop.html) displays a product grid loaded from the MongoDB database via the GET /products API.
2. Each product shows:
    a. Name
    b. Price
    c. Description
    d. Category
3. Hover effects enhance the user experience.

Managing the Shopping Cart

1. **Add to Cart**:

    a. Click "Add to Cart" on a product.
    b. The item is added via the POST /cart/add API, and a confirmation message appears.
    c. If the item is already in the cart, the quantity is updated.
2. **View Cart**:
    a. A responsive cart sidebar displays all items, including product name, quantity, unit price, and total price.
    b. The cart counter badge updates in real-time, reflecting the total number of items.
3. **Modify Cart**:
    a. Use quantity controls to increase/decrease items via the PUT /cart/update API.
    b. Remove items using the "Remove" button via the DELETE /cart/remove API.
    c. The cart subtotal is recalculated automatically.
4. **Cart Persistence**:
    a. Cart data is stored in MongoDB, ensuring persistence across browser sessions and devices.
    b. The cart is user-specific, isolated using JWT authentication.
5. **Empty Cart**:
    a. If the cart is empty, guidance is provided to add items.

Checkout Process

1. Click the "Proceed to Checkout" button on the shop page.
2. You're navigated to the checkout page (checkout.html).
3. Complete the shipping information form:
    a. **City**: Select from a dropdown of 20 major US cities.
    b. **Phone Number**: Enter a 10-digit number, automatically formatted as (XXX) XXX-XXXX.
    c. Real-time validation provides color-coded feedback and dynamic error messages.
    d. The submit button is enabled only when all inputs are valid.
4. Review the order summary, including cart items, tax, and shipping costs.
5. Submit the form to process the order via the POST /orders API.
6. Upon success, you're redirected to the order confirmation page (confirmation.html).

Order Confirmation

1. The confirmation page displays:
    a. Order ID
    b. Order details (items, quantities, total)
    c. Success message

2. Options include:
    a. Print receipt
    b. Continue shopping (returns to shop.html)
3. The cart is cleared after a successful order.

4. User Experience Features

- **Real-Time Feedback**: Color-coded inputs, dynamic error messages, and loading states enhance usability.
- **Toast Notifications**: Confirm user actions (e.g., adding to cart, successful checkout).
- **Responsive Design**: The interface adapts to all screen sizes.
- **Error Handling**: User-friendly messages guide you through errors (e.g., invalid input, failed API calls).
- **Seamless Navigation**: Consistent navigation menus ensure easy movement between pages.

5. Troubleshooting

- **Login Issues**: Ensure username and password meet the 5-character minimum and correct format. Check for typos.
- **Cart Not Updating**: Verify your internet connection, as cart operations rely on API calls.
- **Checkout Errors**: Ensure all form fields (city, phone) are valid. Contact the administrator if issues persist.
- **Unauthorized Access**: Log in to access protected pages. If redirected to login, your session may have expired.

Admin Guide

1. Overview

Administrators manage the E-Commerce platform's backend, ensuring smooth operation of user accounts, product data, cart functionality, and order processing. The system is built with Node.js, Express.js, and MongoDB, with APIs tested using Thunder Client.

2. System Setup

- **Environment**:
    - Run the application locally on http://localhost:5000.
    - Ensure MongoDB is running and the connection string is set in the .env file.
- **Dependencies**:
    - Install Node.js dependencies listed in package.json.
    - Use npm install to set up the server.
- **Starting the Server**:
    - Run node server.js to start the backend.

    - Use Live Server for frontend development.
- **API Testing**:
    - Use Thunder Client to test API endpoints (/register, /login, /products, /cart/*, /orders).

3. Backend Management

Project Structure

- **Frontend**:
    - Home.html: Landing page with navigation.
    - Login.html: Enhanced login with validation.
    - Register.html: Comprehensive registration with validation.
    - Shop.html: Product catalog with cart functionality.
    - Checkout.html: Checkout interface with city dropdown and phone validation.
    - Confirmation.html: Order confirmation display.
- **Backend**:
    - Server.js: Main server file with integrated cart and order processing APIs.
    - Routes/auth.js: Handles authentication endpoints.
    - Models/:
        - User.js: User schema for registration and login.
        - Product.js: Product schema for the catalog.
        - CartItem.js: Cart item schema for shopping cart.
        - Order.js: Order schema for order processing.
    - .env: Stores MongoDB connection string.
    - Package.json: Lists dependencies and scripts.

API Endpoints

- **Authentication**:
    - POST /register: Creates a new user with hashed password (bcrypt).
    - POST /login: Authenticates user and returns JWT.
- **Products**:
    - GET /products: Retrieves product catalog from MongoDB.
- **Cart**:
    - POST /cart/add: Adds item to cart.

    - GET /cart: Retrieves user's cart.
    - PUT /cart/update: Updates item quantity.
    - DELETE /cart/remove: Removes item from cart.
- **Orders**:
    - POST /orders: Processes checkout and stores order.
- All endpoints are protected by JWT middleware (Bearer token in Authorization header).

Security

- **Password Hashing**: Passwords are hashed using bcrypt before storage.
- **JWT Authentication**: Tokens are validated for protected endpoints, with proper handling for expired/invalid tokens.
- **Environment Variables**: Store sensitive data (e.g., MongoDB connection string) in .env.
- **Input Validation**: Enhanced validation ensures secure data handling (e.g., username format, phone number).

4. Administrative Tasks

Managing Users

- Monitor the MongoDB User collection for account data.
- Handle duplicate username errors during registration.
- Reset user passwords if needed (requires custom implementation).

Managing Products

- Add or update products in the MongoDB Product collection.
- Ensure product data includes name, price, description, and category.
- Test product display using the GET /products API.

Managing Carts

- Verify cart data in the CartItem collection.
- Check for user-specific cart isolation using JWT.
- Handle errors for failed cart operations (e.g., invalid product ID).

Managing Orders

- Monitor the Order collection for completed orders.
- Verify order details (items, shipping info, total).
- Test order processing using the POST /orders API.

Error Handling

- Review API response logs for errors (e.g., 401 Unauthorized, 400 Bad Request).
- Ensure user-friendly error messages are displayed on the frontend.
- Handle edge cases like empty carts or invalid tokens.

5. Maintenance

- **Port Alignment**: All API calls are configured to http://localhost:5000. Verify this in case of connectivity issues.
- **Database**:
  - Regularly back up the MongoDB database.
  - Monitor for data integrity (e.g., no duplicate usernames).
- **Updates**:
  - Recent enhancements include refined login/registration validation, city dropdown, and phone number formatting.
  - Simplified order processing by removing payment handling and making paymentInfo optional.
- **Testing**:
  - Use Thunder Client to simulate user actions (e.g., adding to cart, checking out).
  - Test responsive design across devices.
  - Validate all pages for consistent navigation and authentication state.

6. Troubleshooting

- **API Failures**: Check server.js logs and ensure MongoDB is running.
- **Port Mismatch**: Confirm all API calls use http://localhost:5000.
- **JWT Issues**: Verify token generation and storage in localStorage. Clear browser storage if tokens are corrupted.
- **Validation Errors**: Ensure frontend and backend validation rules align (e.g., 5-character minimums, phone format).
- **Database Issues**: Check MongoDB connection string in .env and ensure the database is accessible.

Contact

For support, contact the site administrator or development team (me). Refer to my Github Portfolio for source code, project diagrams, and specific code specifications.

Link: https://github.com/Darnae/Senior-Project-