Query : prepare test cases for user registration and product catalogue for a fashion e commerce orject

**Test Cases Overview**

|  |  |
| --- | --- |
| **Module** | **Test Scenario** |
| User Accounts | Register, login, update profile, reset password |
| Product Catalog | Browse, search, view product details, check recommendations |
| Shopping Cart | Add/remove items, update quantity, verify order summary |
| Checkout | Enter shipping details, select payment method, confirm order |
| Seller Dashboard | Create product listing, manage inventory, process orders |
| Customer Support | Live chat, submit contact form, view FAQ section |
| Cross-Browser | Test site behavior on Chrome & Safari (Mac & Windows) |

**Detailed Test Cases**

**User Registration**

|  |  |  |
| --- | --- | --- |
| **Test Case ID** | **Description** | **Expected Result** |
| UR-01 | Verify that users can access the registration page | Registration page should load properly |
| UR-02 | Verify that mandatory fields (name, email, password) are required | System should prompt user to fill mandatory fields |
| UR-03 | Verify successful registration with valid details | User should receive a confirmation email |
| UR-04 | Verify registration with an already registered email | System should display an error message |
| UR-05 | Verify password complexity requirements | System should enforce password rules |
| UR-06 | Verify that users receive an email verification link after registration | Verification email should be sent |
| UR-07 | Verify that users can log in after successful registration | User should be able to log in successfully |

**Product Catalog**

|  |  |  |
| --- | --- | --- |
| **Test Case ID** | **Description** | **Expected Result** |
| PC-01 | Verify that users can browse products by category | Products should be displayed correctly |
| PC-02 | Verify that users can search for products by keyword | Relevant products should appear in search results |
| PC-03 | Verify that product details page displays correct information | Product page should show name, price, description, and images |
| PC-04 | Verify that users can filter products by price range, brand, and style | Filtered results should match criteria |
| PC-05 | Verify that personalized recommendations appear based on browsing history | Relevant products should be suggested |
| PC-06 | Verify that users can view product reviews and ratings | Reviews should be visible on the product page |
| PC-07 | Verify that unavailable products show "Out of Stock" status | Out-of-stock items should be marked properly |

**great. can we have test data as well for each test cases combinations**

**Detailed Test Cases**

**User Registration**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Description** | **Test Steps** | **Test Data** | **Expected Result** |
| UR-01 | Verify that users can access the registration page | 1. Open the website 2. Click on "Sign Up" | N/A | Registration page should load properly |
| UR-02 | Verify that mandatory fields (name, email, password) are required | 1. Navigate to Sign Up 2. Click Register without entering details | N/A | System should prompt user to fill mandatory fields |
| UR-03 | Verify successful registration with valid details | 1. Enter valid name, email, and password 2. Click Register | Name: John Doe Email: john@example.com Password: Pass@123 | User should receive a confirmation email |
| UR-04 | Verify registration with an already registered email | 1. Enter an already registered email 2. Click Register | Email: existinguser@example.com | System should display an error message |
| UR-05 | Verify password complexity requirements | 1. Enter a weak password 2. Click Register | Password: 12345 | System should enforce password rules |
| UR-06 | Verify that users receive an email verification link after registration | 1. Complete registration 2. Check email inbox | Email: newuser@example.com | Verification email should be sent |
| UR-07 | Verify that users can log in after successful registration | 1. Log in with valid credentials | Email: john@example.com Password: Pass@123 | User should be able to log in successfully |

**Product Catalog**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Description** | **Test Steps** | **Test Data** | **Expected Result** |
| PC-01 | Verify that users can browse products by category | 1. Navigate to Product Categories 2. Click on a category | Category: Dresses | Products should be displayed correctly |
| PC-02 | Verify that users can search for products by keyword | 1. Enter a product name in search 2. Click Search | Keyword: "Leather Jacket" | Relevant products should appear in search results |
| PC-03 | Verify that product details page displays correct information | 1. Click on a product from search results | Product: "Red Dress" | Product page should show name, price, description, and images |
| PC-04 | Verify that users can filter products by price range, brand, and style | 1. Apply price and brand filters | Price: $50-$100 Brand: "Nike" | Filtered results should match criteria |
| PC-05 | Verify that personalized recommendations appear based on browsing history | 1. Browse multiple product pages 2. Go to homepage | Browsing history: "Sneakers, Hoodies" | Relevant products should be suggested |
| PC-06 | Verify that users can view product reviews and ratings | 1. Open a product page 2. Scroll to reviews section | Product: "Running Shoes" | Reviews should be visible on the product page |
| PC-07 | Verify that unavailable products show "Out of Stock" status | 1. Search for an out-of-stock product | Product: "Limited Edition Bag" | Out-of-stock items should be marked properly |

**can you please provide End to End test case for user flow**

| **est Case ID** | **Description** | **Test Steps** | **Test Data** | **Expected Result** |
| --- | --- | --- | --- | --- |
| E2E-01 | **User Registration & Login** | 1. Navigate to the "Sign Up" page.  2. Enter valid user details (name, email, password).  3. Submit the form.  4. Verify email confirmation.  5. Log in with registered credentials. | Name: John Doe  Email: john@example.com  Password: Pass@123 | User successfully registers, receives a verification email, and logs in successfully. |
| E2E-02 | **Browsing & Searching Products** | 1. Navigate to the product catalog.  2. Search for a product using keywords.  3. Apply category and price filters.  4. Click on a product to view details. | Keyword: "Leather Jacket"  Category: "Jackets"  Price: $50-$100 | Product search and filtering work correctly, displaying accurate results. |
| E2E-03 | **Adding Products to Cart** | 1. On the product page, select size/color (if applicable).  2. Click "Add to Cart."  3. Verify the cart updates correctly. | Product: "Black Leather Jacket"  Size: M  Quantity: 1 | Product is added to the cart, and cart updates accurately. |
| E2E-04 | **Proceed to Checkout** | 1. Navigate to the shopping cart.  2. Review the order summary.  3. Click "Proceed to Checkout." | Cart contains 1 "Black Leather Jacket" | User successfully moves to the checkout process. |
| E2E-05 | **Entering Shipping & Payment Details** | 1. Enter valid shipping address.  2. Select shipping method.  3. Enter valid payment details.  4. Submit the order. | Address: "123 Fashion St, NY"  Payment: Visa Card (\*\*\*\*1234) | Order is successfully placed, and confirmation is displayed. |
| E2E-06 | **Order Confirmation & Tracking** | 1. Check email for order confirmation.  2. Log in and navigate to "Order History."  3. Verify order status updates. | Order ID: #SH12345 | Order details are correct, and tracking status updates as expected. |

* **As a part of shift left testing, can you tell me which tests will go in which layer of test pyramid?**

**Mapping Tests to the Test Pyramid for Shift-Left Testing**

The **Test Pyramid** consists of three primary layers:

1. **Unit Tests (Base Layer - Fast & Automated)**
2. **API/Integration Tests (Middle Layer - Service-Level)**
3. **UI/End-to-End Tests (Top Layer - Slow & Comprehensive)**

Here's how different tests from the "Style Haven" test plan fit into the Test Pyramid:

**1. Unit Tests (Automated)**

**Goal:** Validate individual functions/components at the code level before integration.  
✅ **Tests Included:**

* User Registration: Validate email format, password complexity, and required fields.
* Product Search: Test search logic, filtering, and sorting.
* Shopping Cart: Ensure add/remove/update cart items work correctly.
* Checkout: Validate input fields, calculations (subtotal, tax, total).
* Payment Processing: Mock API calls for payment validation.

🔧 **Tools:** Jest, Mocha, JUnit, Selenium (for UI component testing).

**2. API/Integration Tests (Automated)**

**Goal:** Validate the communication between different system components and external services.  
✅ **Tests Included:**

* **User Authentication API** (Login, Signup, Token Expiry, Password Reset).
* **Product Catalog API** (Fetch product details, search, filter).
* **Shopping Cart API** (Add, update, delete items, validate stock availability).
* **Checkout API** (Address validation, payment gateway integration).
* **Order Management API** (Order placement, status updates, cancellation).

🔧 **Tools:** Postman, Newman, REST Assured, Cypress (API testing).

**3. UI/End-to-End Tests (Manual & Automated)**

**Goal:** Ensure the entire user journey works seamlessly across different browsers and devices.  
✅ **Tests Included:**

* **End-to-End User Journey:** Registration → Search → Add to Cart → Checkout → Order Confirmation.
* **Cross-Browser Testing:** Chrome & Safari (on Mac & Windows).
* **UI Validation:** Button clicks, form submissions, responsive design, page load times.
* **Error Handling:** Invalid login attempts, incorrect payment details, handling out-of-stock items.
* **Performance Testing:** Load times for search, cart, checkout, and order confirmation pages.

🔧 **Tools:** Selenium, Cypress, Playwright, BrowserStack (for cross-browser testing).

**How Shift-Left Helps?**

* **Early detection of defects in Unit/API tests** → Reduces cost of fixing defects in later stages.
* **Automation at lower levels** → Faster feedback loops, reducing UI test dependency.
* **Parallel execution of API/UI tests** → Ensures smoother test coverage before deployment.