System Test Report

Generated: 09 Oct 2025 07:45 AM (Australia/Sydney)

Subject: 32601 Advanced Project Management – Assignment 3 (System Testing)

Group ID: [Enter your Group ID]

Team Members: [List all member names and UTS emails]

1. Overview

Target System: https://ecommerce-playground.lambdatest.io/index.php?route=common/home
This report outlines our group's system testing plan and execution for the LambdaTest e-commerce playground site. We focus on end-to-end functional flows typical of an online retail platform.

High-level Business Goals:
• Enable customers to discover products, register/log in, place orders, and manage accounts.
• Support secure checkout, payment, and order tracking.
• Provide administrators with catalog and order management tools (beyond current scope).

Key User Roles:
• Visitor (browse, search, add-to-cart)
• Registered Customer (account, checkout, order history)
• Admin/Staff (catalog & order management – out of test scope)

Primary Features Under Test:
• Registration & Login • Product Search & Filters • Product Page • Cart • Checkout • Wishlist • Account Orders

2. System Functions

The following table lists the core functions selected for testing. Update or extend as needed.

Function ID	Function Name	Description and Location (URL)
F1	User Registration	Register a new customer account with a unique email. https://ecommerce-playground.lambdatest.io/index.php?route=account/register
F2	User Login	Log in with registered email and password. https://ecommerce-playground.lambdatest.io/index.php?route=account/login

F3	Product Search	Search for products using the site search. [Use top search bar on home page]
F4	Product Filters	Filter product listings (e.g., by category/price). [Navigate to a category and apply filters]
F5	Product Details Page	View a product page and verify details (price, images, description).
F6	Add to Cart	Add an item to the cart from product listing or product page.
F7	Cart Update	View cart, change quantity, remove items. https://ecommerce-playground.lambdatest.io/index.php?route=checkout/cart
F8	Checkout	Proceed through checkout steps (address, shipping, payment simulation). https://ecommerce-playground.lambdatest.io/index.php?route=checkout/checkout
F9	Wishlist	Add/Remove items in wishlist (requires login).
F10	Order History	View past orders after successful checkout (requires login).

Dependencies & Flow: Registration (F1) → Login (F2) → Browse/Search (F3/F4) → PDP (F5) → Add to Cart (F6) → Cart Update (F7) → Checkout (F8) → Order History (F10). Wishlist (F9) depends on F2.

3. Test Team

Provide full names, UTS emails, subject & tutorial codes, roles, and responsibilities.

Student ID	Member Name	Role	Responsibilities (Function IDs)
[UTS ID]	[Full Name] <first.last@student.uts.edu.au>	[e.g., Team Lead / Tester / Scribe / DevOps]	[e.g., F1, F2, F8]
[UTS ID]	[Full Name] <first.last@student.uts.edu.au>	[e.g., Team Lead / Tester / Scribe / DevOps]	[e.g., F1, F2, F8]
[UTS ID]	[Full Name]	[e.g., Team Lead / Tester / Scribe	[e.g., F1, F2, F8]

	<first.last@student.uts.edu.au>	/ DevOps]	
[UTS ID]	[Full Name] <first.last@student.uts.edu.au>	[e.g., Team Lead / Tester / Scribe / DevOps]	[e.g., F1, F2, F8]

Presentation: [In-class Week 12 / Recorded – link here]

4. Task Schedule

We group tasks to maximise parallel testing while respecting functional dependencies. Example high-level Gantt-style plan:

Task	Owner	Start	End	Notes
F1 Registration	[Name A]	2025-10-10	2025-10-10	Must complete before F2/F9/F10
F2 Login	[Name B]	2025-10-10	2025-10-10	Depends on F1
F3 Search & F4 Filters	[Name C]	2025-10-10	2025-10-11	Can run in parallel after F2 ready
F5 PDP + F6 Add to Cart	[Name D]	2025-10-11	2025-10-12	Parallel with F3/F4
F7 Cart Update	[Name D]	2025-10-12	2025-10-12	After F6
F8 Checkout	[Name A]	2025-10-12	2025-10-13	After F7
F9 Wishlist	[Name B]	2025-10-12	2025-10-13	Requires login (F2)
F10 Order History	[Name C]	2025-10-13	2025-10-13	After successful F8

Parallelism: F3–F7 can overlap across members once registration/login complete. Blocking items are F1→F2 and F6→F7→F8→F10.

5. Test Environments and Tools

Platforms & OS:
• x86 laptops/desktops; Windows 11, macOS 14.
Browsers:

• Chrome (latest), Edge (latest), Safari (latest on macOS).

Automation Options (choose one or mix):
• Selenium WebDriver (Python/Java) – UI flow automation.
• Playwright (TypeScript/Python) – reliable cross-browser testing.
• Cypress (JavaScript) – component/UI tests (Chromium based).

Setup (example – Playwright Python):
1) Create venv and install Playwright and pytest.
2) Run 'playwright install' for browsers.
3) Structure tests with page objects; add fixtures for login/session.
4) Use CI (e.g., GitHub Actions) for regression runs; export HTML reports/screenshots.

Accessibility & Performance (optional):
• Lighthouse for basic performance audits.
• axe-core for accessibility checks.

6. Test Cases (Individual – 3 cases per member)

Below are three fully defined examples. Each member should adapt data and add screenshots during execution.

Test Case ID: T001

Function IDs/Names: F1 User Registration; F2 User Login

Tester: [Your Name]

Step #	Test Action	Expected Result
1	Navigate to Registration page URL.	Registration form loads.
2	Enter unique email, valid password, and required personal info.	Form accepts inputs; password policy satisfied.
3	Agree to privacy/terms if required.	Checkbox validated.
4	Submit the form.	Account created; success message shown; redirected to account page or login.
5	Attempt login using the new credentials.	Login succeeds (verifies account creation).

Test Case ID: T002

Function IDs/Names: F5 Product Details; F6 Add to Cart; F7 Cart Update

Tester: [Your Name]

Step #	Test Action	Expected Result
1	From home, search for a product (e.g., 'iPhone') or browse a category.	Results list appears.
2	Open a product page.	Product details visible (title, price, images).
3	Click 'Add to Cart' on PDP.	Success alert or cart notification appears.
4	Open cart page.	Item appears with correct title/price.
5	Increase quantity by 1 and update.	Cart total reflects new quantity.
6	Remove item from cart.	Cart updates and item is removed.

Test Case ID: T003

Function IDs/Names: F8 Checkout; F10 Order History

Tester: [Your Name]

Step #	Test Action	Expected Result
1	Ensure user is logged in (use account from T001).	User session is active; account menu visible.
2	Add any product to cart; proceed to checkout.	Checkout page loads with the cart item.
3	Enter shipping address; choose shipping method.	Validation passes; next step enabled.
4	Select payment (simulation) and place order.	Order confirmation page appears with an order number.
5	Navigate to Account →	New order appears with

	Order History.	correct details.

Test Results (per Test Case)

Attach annotated screenshots and mark each step Pass/Fail/Error. Capture defects with URLs, DOM locators, and browser/OS details.

7. Analysis and Reflections

Summary Statistics: [e.g., 18 steps passed, 2 failed, 0 error across 3 cases].

- Defect Log: [ID, title, severity, repro steps, expected vs actual, environment]
- Root Causes: [e.g., validation bug, flaky selector, environment config]
- Improvements: [test data management, CI schedule, cross-browser matrix]
- Next Iteration Plan: [prioritise checkout edge cases, expand negative tests]

Appendix A – Tooling Commands (Example: Playwright + Pytest)

```
python -m venv .venv
source .venv/bin/activate  # or .venv\Scripts\activate on Windows
pip install playwright pytest pytest-html
playwright install
pytest -q --html=report.html --self-contained-html
```

Appendix B – Negative Test Ideas

- Registration with existing email; weak passwords; missing mandatory fields
- Login with wrong credentials; SQLi strings in fields (should be safely handled)
- Checkout with empty cart; invalid addresses; network throttling