# Regression Testing Checklist — Demo Web Shop

**Purpose:** Quick one-page checklist to design and run regression testing for the e‑commerce application (Part 3).

## Risk Assessment Matrix

* **Identify critical user journeys**: Checkout (guest & registered), Login/Registration, Product Search & Filter, Add to Cart / Update Cart, Payment (card/PayPal), Order Confirmation & Email, Account Management (addresses, passwords), Return/Refund flows, Admin product/order management.
* **Classify features by priority**: P0 (business-critical: checkout, payment, login), P1 (high: search, cart, account), P2 (medium: recommendations, wishlist), P3 (low: UI enhancements, non-critical APIs).
* **Impact vs Probability analysis**: For each feature assign Impact (High/Med/Low) and Probability (High/Med/Low); compute risk = Impact × Probability to drive test focus.

## Test Selection Strategy

* **Define smoke (must-run) suite** — run on every build:
  + Application launches & home page load
  + Login/Logout
  + Search returns results
  + Add to cart & view cart
  + Checkout (happy path) & payment success
  + Order confirmation page & confirmation email
* **Full vs Targeted regression criteria**:
  + Run **Full regression** when: major release, payment or checkout changes, critical bugfix in core flows, or pre-prod release.
  + Run **Targeted regression** when: small/isolated change (one microservice/feature), minor UI tweaks, or low-risk bugfix — include related P0/P1 tests plus impacted modules.
* **Automation vs Manual decisions**:
  + Automate smoke suite, checkout/payment workflows, search, add-to-cart, and frequently repeated flows.
  + Manual for exploratory testing, UI polish checks, visual regressions (unless visual regression tooling available), complex multi-step workflows with unstable data.
  + Maintain automation for cross-browser permutations only if ROI justifies cost.

## Execution Framework

* **Test case prioritization approach**:
  + Rank by risk score (Impact×Probability) then by business priority (P0 first).
  + Within same rank, prefer tests that catch regressions fastest (short runtime, high coverage).
* **Regression suite maintenance strategy**:
  + Review suites every sprint; add tests for new features and retire flaky/obsolete tests.
  + Tag tests (smoke, p0, p1, integration, api) and keep test metadata (owner, last-run, last-pass).
  + Quarantine flaky tests, create JIRA for root cause, fix automation or test data.
  + Keep test data versioned and resettable; use test environments with known seeded data.
* **Execution timeline recommendations**:
  + **On every commit / CI**: run smoke suite (fast) — gate for deployment to next stage.
  + **Nightly**: run targeted regression for changed modules.
  + **Weekly or on-demand**: full regression (long run) or before major releases.
  + **Pre-release**: run full regression + performance and security quick checks.

## Reporting & Metrics (keep short)

* Track: smoke pass rate, regression pass rate, test coverage (automation %), mean time to detect/regress (MTTD), flaky test count.
* Post-release: capture defects found-in-production and fold into risk matrix.

**Quick actions to finish setup:** - [ ] Create an automation smoke pipeline in CI (fast feedback). - [ ] Tag existing tests by priority and ownership. - [ ] Schedule nightly targeted runs and weekly full runs. - [ ] Document regression entry/exit criteria in the release checklist.

*End of checklist — adapt cadence to team capacity and release frequency.*