# **Regression Testing Strategy — DemoWebShop (e-commerce)**

**Objective** Design a pragmatic, risk-focused regression testing approach for DemoWebShop that minimizes release risk while keeping execution cost and maintenance manageable.

## **1) Risk Assessment Matrix**

### **1.1 Critical user journeys (top-level)**

1. Account management — Register / Login / Logout / Password reset
2. Product discovery — Browse categories, Search, Filters, Sorting
3. Product details — PDP shows name, price, description, attributes, availability
4. Cart operations — Add/Update/Remove items, quantity limits
5. Checkout & payments — Shipping, Billing, Payment selection, Order placement
6. Order management — Order history, order details, order status
7. Promotions & pricing — Coupons, discounts, price calculations, tax
8. Content & navigation — Header/footer, menus, responsive layout
9. Accessibility & keyboard navigation — basic ARIA, focus order
10. Third-party integrations — Payment gateway, email notifications, analytics

### **1.2 Feature priority classification**

Use three priority buckets (P0 / P1 / P2) for regression focus.

* **P0 (Critical — Must pass every release)**
  + Checkout & payments
  + Cart operations
  + Product detail viewing
  + User Login / Registration / Password reset
  + Order placement & confirmation
* **P1 (High — Should pass in release, run regularly)**
  + Product discovery/search/filter/sort
  + Promotions & coupon application
  + Order history/reports
  + Basic responsiveness (mobile/desktop)
* **P2 (Medium — Run periodically or after relevant changes)**
  + Newsletter subscription
  + Static content, help pages
  + Minor UI cosmetic features
  + Non-critical integrations (analytics tagging, optional widgets)

### **1.3 Impact vs Probability analysis (sample matrix)**

Provide a 3×3 table mapping Impact (High/Medium/Low) × Probability (High/Medium/Low) to decide frequency.

* **High Impact & High Probability:** Checkout, Cart, Login — run every cycle (Daily/Per-build).
* **High Impact & Medium/Low Probability:** Payment gateway changes, pricing logic — run in full regression for releases.
* **Medium Impact & High Probability:** Search quality, filters — run in targeted regression on UI/logic changes.
* **Low Impact & Low Probability:** Content pages — run weekly or on-demand.

## **2) Test Selection Strategy**

### **2.1 Smoke test suite (must-run tests)**

A compact suite (10–15 tests) that verifies application is stable enough for deeper testing. Example smoke cases:

* SMK\_001: Home page loads (title, main navigation visible)
* SMK\_002: User login (valid credentials)
* SMK\_003: Product search returns results
* SMK\_004: Open product detail page (name, price visible)
* SMK\_005: Add product to cart
* SMK\_006: Update product quantity in cart
* SMK\_007: Proceed to checkout (shipping address page reachable)
* SMK\_008: Place order using test payment (sandbox)
* SMK\_009: Order confirmation page shows order id
* SMK\_010: Logout

Run smoke after every deployment to test environment and on each nightly build.

### 2.2 Full regression vs targeted regression criteria

**Full regression** (run entire suite) when: - Major release or release candidate (GA) is prepared. - Significant changes across multiple modules (checkout, catalog, pricing). - Upgrades to shared libraries, framework, or critical infra (DB, auth).

**Targeted regression** (subset) when: - Minor feature release or bug-fix localized to a single module (e.g., search, newsletter). - Hotfixes that affect a specific area. - Time-limited test window — run smoke + impacted-area suite.

Selection rules: - Always include all **P0** tests. - Include P1 tests if the change touches search, cart, checkout, or promotions. - Use dependency mapping (impact analysis) to pick relevant tests automatically where possible.

### 2.3 Automation vs Manual testing decisions

**Automate when:** - Tests are deterministic and stable (login, add to cart, checkout flow). - Tests are run frequently (smoke, P0 regression) — automation yields ROI. - Tests are data-driven or repetitive (price calculations, promo combinations).

**Keep manual when:** - Tests require human judgment (visual polish, UX, subjective layout checks). - Exploratory testing for new features. - Accessibility audits requiring screen reader checks and manual verification (though many accessibility checks can be automated for basics).

Automation target: - Aim to automate 70–85% of P0 and P1 tests. - Manual testing for visual/UX, exploratory, and complex accessibility scenarios.

## **3) Execution Framework**

### **3.1 Test case prioritization approach**

Score each test case with a **Priority Score = ImpactWeight × FrequencyWeight × ComplexityFactor** (simplified example): - ImpactWeight: P0=5, P1=3, P2=1 - FrequencyWeight: Daily=5, Weekly=3, Release=2 - ComplexityFactor: Easy=1, Medium=1.5, Hard=2

Sort tests descending by the score; run highest-scoring tests first in constrained windows.

### 3.2 Regression suite maintenance strategy

* **Version the test suite**: Keep suites per release line (e.g., regression\_v1.2).
* **Review cycle**: Every sprint, review failing/obsolete tests — retire or rewrite flakey tests.
* **Flakiness budget**: If a test flakes more than N times in 10 runs, quarantine it for investigation.
* **Ownership**: Assign test owners by module — they are responsible for maintenance and updates.
* **Test data management**: Use stable test accounts and sandbox payment credentials; ensure data reset between runs.
* **CI Integration**: Run smoke on every push/PR; run targeted regression on merge to develop; run full regression on nightly or pre-release pipeline.

### 3.3 Execution timeline recommendations

A sample cadence for a typical sprint/release flow (2-week sprint):

* **On every PR / Build:** Run smoke suite (SMK\_001..SMK\_010) → quick pass/fail (< 15 minutes if automated in parallel).
* **Nightly:** Run prioritized regression subset (P0 + top P1 tests) — results stored to dashboard.
* **Weekly:** Run a larger regression (most automated tests) to catch integration regressions.
* **Pre-release (2–3 days before release):** Run full regression (all automated tests) + manual exploratory and accessibility audits.
* **Post-deployment:** Run smoke on production; sanity checks for critical journeys.

Sample time budget (for automation-friendly environment using parallel CI): - Smoke: 10–15 minutes - Prioritized regression subset: 30–60 minutes - Large regression: 2–4 hours (parallelized) - Full regression: 4–12 hours (depending on breadth and parallelization)

Adjust timings to fit CI capacity and SLAs.

## **4) Practical Recommendations & Next Steps**

1. **Start with a small, reliable smoke suite** and automate it first.
2. **Map features-to-tests** to enable targeted regression selection automatically (traceability matrix).
3. **Introduce test data and environment provisioning** (infrastructure-as-code) to ensure reproducible test runs.
4. **Add observability**: failing tests should capture screenshots, DOM snapshots, and logs.
5. **Schedule regular reviews** of the regression inventory to remove outdated tests and add new ones.
6. **Measure ROI**: Track automation pass rates, flakiness, and mean time to repair test failures.

## **5) Appendix — Suggested Smoke & Regression Test List (quick reference)**

**Smoke (must-run)**: SMK\_001..SMK\_010 (see above)

**Top P0 Regression Candidates (automate priority)**: - Login/Registration - Product search & PDP - Add to cart / update qty - Checkout (happy path) - Order confirmation & email

**Accessibility quick checks (automated + manual)**: - Page titles and headings - Image alt attributes - Form labels present - Keyboard navigation for critical flows - Color contrast automated scans