**Performance test plan – Login flow**

Objectives

* Validate that the login journey from landing on “/” to reaching inventory.html meets response-time targets under expected and peak load.
* Ensure correctness under load: valid users reach inventory; invalid/locked users get the correct errors; no elevated 5xx/timeout rates.

Test scope

* Pages/APIs:
  + GET / (login page and static assets).
  + Auth submit (login POST/XHR), redirects, inventory.html render readiness.
* Personas and mixes:
  + standard\_user (primary baseline).
  + problem\_user, performance\_glitch\_user (measured separately; excluded from hard SLOs).
  + locked\_out\_user and invalid creds (negative path checks at small traffic percentage).

Workload model

* Concurrency model: open model (arrivals/second) to mimic real users; alternative closed model (fixed VUs) for capacity scans.
* Proposed steps (staging capacity; tune per environment):
  + Warm-up: 2 → 10 logins/sec over 5 min.
  + Peak: 10 → 30 logins/sec over 10 min; hold 10 min.
  + Stress: 30 → 50 logins/sec over 5 min; hold 5 min; ramp down 5 min.
* Traffic composition:
  + 80% standard\_user success path.
  + 10% invalid credentials.
  + 5% locked\_out\_user.
  + 5% performance\_glitch\_user.
* Think time:
  + 200–500 ms between page load and submit to simulate a human pause.
* Data:
  + Reuse persona accounts, use randomized invalid users to avoid caching anomalies.
  + Fresh sessions, disable client cache for “worst-case” static asset timings where needed.

SLIs/SLOs and pass criteria

* Page-load SLI: TTFB for GET / and primary JS/CSS p95 ≤ 400 ms; p99 ≤ 800 ms (staging hardware baseline).
* End-to-end login SLI: from click “Login” to inventory.html with #inventory\_container visible p95 ≤ 2.0 s; p99 ≤ 3.0 s for standard\_user.
* Error rates:
  + 5xx ≤ 0.5% during peak; timeouts ≤ 0.5%.
  + 4xx expected only for negative tests.
* Functional correctness:
  + standard\_user reaches inventory and sees product grid/cart icon.
  + invalid/locked users remain on login with correct error banners.

Instrumentation and measurements

* Capture:
  + HTTP metrics: status codes, latency percentiles per endpoint, redirects.
  + Front-end markers: DOMContentLoaded, load, inventory container visible.
  + Server-Timing header (if available) for backend phase breakdown.
* Artifacts:
  + HARs for sample runs, console/network error summaries, percentile charts (p50/p90/p95/p99).

Scenarios

1. Baseline success (standard\_user)

* Steps: GET / → submit creds → wait for inventory.html + #inventory\_container.
* KPIs: end-to-end latency, errors, throughput.

1. Negative creds

* Steps: GET / → submit invalid creds.
* KPIs: fast rejection latency; no 5xx; error banner presence.

1. Locked user

* Steps: submit locked\_out\_user.
* KPIs: error banner presence; no redirect; latency comparable to invalid path.

1. Persona perf characterization

* Steps: performance\_glitch\_user login.
* KPIs: latency profile recorded; excluded from SLO but trended.

1. Static assets pressure

* Steps: GET / with disabled cache at peak arrivals.
* KPIs: bundle fetch latency and error rate; CDN health.

Environments and constraints

* Staging with production-like TLS/CDN; document browser/OS, bandwidth/latency profiles used.
* No third-party blockers; set constant test window to reduce external variance.

Reporting and gating

* Daily scheduled runs and pre-release runs; publish dashboards with percentiles, error rates, and trend deltas vs last baseline.
* Gate criteria: fail if p95 login > 2.0 s for standard\_user or 5xx > 0.5% at peak; investigate before promoting.

Rollback criteria

* Abort test and escalate if 5xx spikes or timeouts exceed 1% within 5 minutes at any stage.

**Testability improvements**

Selectors and DOM contracts

* Standardize stable data-test attributes for critical controls:
  + Login: [data-test="username"], [data-test="password"], [data-test="login-button"], [data-test="error"].
  + Inventory: [data-test="inventory-container"], [data-test="product-sort-container"], [data-test="inventory-item-name"], [data-test="add-to-cart"], [data-test="remove-from-cart"], [data-test="shopping-cart-link"], [data-test="shopping-cart-badge"].
* Ensure unique hooks per role to avoid strict-mode conflicts where multiple elements share the same accessible name.

Ready-state signals

* Emit a deterministic page-ready signal:
  + Add data-state="ready" on body when inventory list and header are bound.
  + Fire a window event “app:inventory-ready” to simplify waits.

Error messaging determinism

* Centralize validation with codes:
  + Add data-error-code attributes (e.g., ERR\_LOGIN\_LOCKED, ERR\_MISSING\_USERNAME) so tests assert codes instead of brittle full-text.

Performance flags for tests

* Opt-in query flag (e2e=1) to disable heavy animations and defer non-critical third-party scripts during automated runs, preserving correctness while improving stability.

Authentication shortcuts for non-login tests

* Provide a supported pre-auth route or documented storage bootstrap (localStorage/session cookie) so suites can skip UI login for non-auth scenarios, while keeping dedicated tests for the login flow itself.

State propagation and cart consistency

* Publish cart count updates via a shared store event so header badge updates immediately after add/remove on both inventory and cart pages; add a single source of truth to prevent badge staleness.

Sorting determinism

* Use a stable comparator (localeCompare with consistent locale and case policy) and persist the active sort in state so rapid toggles yield deterministic orders.

Visual testing hooks

* Add data-visual-region annotations for header, sort combobox, and cart button to enable targeted snapshot tests, especially for visual\_user persona.

Observability

* Add Server-Timing headers for login and inventory responses (auth, DB, render), expose build hash and persona on a small diagnostics endpoint to stamp test runs.

CI integration

* Store SLOs as code (JSON) and enforce gates in CI; archive traces/HARs on failure.
* Add “selector contract” lint that fails PRs if required data-test hooks are removed or renamed.