E2E Test Plan

E2E Test Plan

> **Project:** {{PROJECT_NAME}} **Version:** {{VERSION}} **Date:** {{DATE}}
> **Author:** {{AUTHOR}} **Status:** Draft | In Review | Approved **Reviewers:** {{REVIEWERS}}

Document History

Version	Date	Author	Changes
0.1	{{DATE}}	{{AUTHOR}}	Initial draft

---

1. E2E Testing Objectives

Objectives:

1. Validate that the {{COUNT}} most critical user journeys work end-to-end in a production-like environment
2. Catch integration failures between frontend, backend, and third-party services that unit/integration tests cannot detect
3. Provide confidence before every production deployment
4. Serve as living documentation of critical user flows

What E2E tests are NOT for:

- Complete feature coverage (that's for unit/integration tests)
- Testing every error state (too expensive to maintain)
- Performance benchmarking (use performance tests)
- Visual pixel-perfection (use visual regression tests separately)

---

2. Critical User Journeys

Journey 1: {{JOURNEY_1_NAME}}

Field	Value
Priority	Critical
Business Impact	{{IMPACT}}
Frequency	{{FREQUENCY}}
Test File	`{{TEST_PATH}}`

Steps:

1. {{STEP_1}}
2. {{STEP_2}}
3. {{STEP_3}}
4. {{STEP_4}}
5. {{STEP_5}}
6. {{STEP_6}}

Assertions:

- {{ASSERTION_1}}
- {{ASSERTION_2}}
- {{ASSERTION_3}}

Journey 2: {{JOURNEY_2_NAME}}

Field	Value
Priority	Critical
Business Impact	{{IMPACT}}
Frequency	{{FREQUENCY}}
Test File	`{{TEST_PATH}}`

Steps:

1. {{STEP_1}}
2. {{STEP_2}}
3. {{STEP_3}}

All Journeys Summary

Journey	Priority	Est. Duration	Automated	Last Pass
{{JOURNEY_1_NAME}}	Critical	{{DURATION}}s	Yes	{{DATE}}
{{JOURNEY_2_NAME}}	Critical	{{DURATION}}s	Yes	{{DATE}}
User registration + email verification	Critical	60s	Yes	—
Password reset flow	High	45s	Yes	—
Account settings update	Medium	30s	Yes	—

---

3. Browser & Device Matrix

Desktop Browsers

Browser	Version	OS	Priority	Notes
Chrome	Latest stable	Windows, macOS	Critical	Highest market share
Firefox	Latest stable	Windows, macOS	High	
Safari	Latest stable	macOS	High	WebKit engine
Edge	Latest stable	Windows	Medium	Chromium-based

Mobile Browsers

Browser	Platform	Version	Priority	Notes
Safari	iOS	Latest	Critical	Highest mobile share
Chrome	Android	Latest	Critical	

Screen Resolutions

Category	Resolution	Test Priority
Desktop HD	1920×1080	Critical
Desktop	1366×768	High

Category	Resolution	Test Priority
Laptop	1280×800	Medium
Tablet (landscape)	1024×768	High
Mobile L	428×926	Critical
Mobile M	375×667	Critical

Matrix in CI: Run Critical priority browser/resolution combinations on every deployment. Full matrix on nightly schedule.

---

4. Test Data Setup & Teardown

Setup Strategy

Data Type	Method	Scope
Test user accounts	Pre-seeded before test run	Test suite
{{DATA_TYPE}}	Created via API in `beforeAll`	Test suite
Isolated test data	Created via API in `beforeEach`	Individual test

Seed command: `bash scripts/e2e-seed.sh {{ENVIRONMENT}}` **Seed verification:** `bash scripts/verify-seed.sh {{ENVIRONMENT}}`

Teardown Strategy

Cleanup Item	Method	Trigger
Test-specific records	DELETE via API	`afterEach`
Test session data	Clear browser storage	`afterEach`
Emails in test inbox	API clear	`afterAll`
Payment test data	Stripe test mode auto-cleanup	Daily job

Rule: Tests must not leave state that affects other tests. Tests must be executable in any order.

Test Accounts

Account	Email	Role	Purpose
Standard user	`e2e-user@test.{{DOMAIN}}`	User	Standard journeys

Account	Email	Role	Purpose
Admin user	`e2e-admin@test.{{DOMAIN}}`	Admin	Admin panel journeys
{{ROLE}} user	`e2e-{{ROLE}}@test.{{DOMAIN}}`	{{ROLE}}	{{PURPOSE}}

Credentials stored in: {{CREDENTIAL_STORE}} (never hardcoded in tests)

---

5. Authentication Handling in Tests

Strategy: {{AUTH_STRATEGY}}

Recommended approach (Playwright):

```
// tests/fixtures/auth.ts
// Authenticate via API, save storage state
// Reuse state across tests in the same suite
// Only use UI login when testing the login flow itself
```

Session reuse:

- Session state saved per role after first auth
- Reused for all tests requiring that role
- Invalidated and refreshed if expired

---

6. Test Environment Requirements

Requirement	Specification
Environment	Staging (`staging.{{DOMAIN}}`)
Database	Dedicated E2E database, seeded before run
External services	Sandbox / test mode (Stripe test, email sandbox)
Stability	No active deployments during E2E run
Data persistence	Isolated — not shared with manual testing
Response times	< {{TIMEOUT}}s for all endpoints (E2E assertions time out at 2× this)

Environment locking: CI acquires a lock before E2E run to prevent concurrent deployments **Lock mechanism:** {{LOCK_MECHANISM}}

7. Flaky Test Management Strategy

Definition: A test that fails inconsistently for the same code

Prevention:

- Use explicit waits (not `sleep`) — wait for elements/network, not time
- Isolate test data (no shared state between tests)
- Use retry-on-assertion, not retry-on-run
- Avoid animation timing issues with `prefers-reduced-motion` in test env

Detection:

- Track flakiness rate per test in CI
- Any test failing > {{FLAKY_THRESHOLD}}% of runs without code changes = flaky

Response:

- Immediately tag flaky test with `@flaky` annotation
- Create bug ticket with priority based on journey criticality
- Quarantine (run separately, don't block CI) while fixing
- Fix within {{FLAKY_FIX_SLA}} days or remove the test
- Monthly flaky test review

---

8. Visual Regression Testing

Tool: {{VIS_TOOL}} **Baseline:** Stored in {{BASELINE_STORAGE}} **Threshold:** Allow up to {{VIS_THRESHOLD}}% pixel difference for font rendering

Check	Scope	Trigger
Full page screenshots	Critical pages	UI-touching PRs
Component snapshots	UI components	Component changes
Responsive layout	Mobile + desktop	Layout changes

Review process: Visual diffs in PR requiring explicit approval before merge

---

9. Performance Assertions Within E2E

Assertion	Metric	Threshold	Journey
Page load	Time to Interactive	< {{TTI}}ms	All critical pages
Core Web Vitals — LCP	Largest Contentful Paint	< {{LCP}}ms	Homepage, landing pages
Core Web Vitals — CLS	Cumulative Layout Shift	< {{CLS}}	All pages
API response	Time in test assertions	< {{API_TIMEOUT}}ms	All API calls

---

10. CI Integration

Trigger: Post-staging deployment

Parallelization:

- Test sharding: {{SHARD_COUNT}} shards
- Browser parallelism: {{BROWSER_WORKERS}} workers per shard
- Estimated total time: {{E2E_TOTAL_TIME}} minutes

CI configuration:

```
# Reference: {{CI_CONFIG_PATH}}
# Key settings:
# - Shards: {{SHARD_COUNT}}
# - Retries: {{RETRY_COUNT}} (for non-flaky tests only)
# - Timeout per test: {{TEST_TIMEOUT}}ms
# - Timeout total: {{SUITE_TIMEOUT}}min
```

On failure:

- Collect screenshots, videos, traces as artifacts
- Retain artifacts for {{ARTIFACT_RETENTION}} days
- Alert Slack channel `#e2e-failures`

---

11. Test Report Format & Artifacts

Report format: {{REPORT_FORMAT}} **Report location:** {{REPORT_URL}} or CI artifacts

Artifacts collected on failure:

Artifact	Format	When Collected

Screenshot	PNG	On step failure
Screen recording	WebM video	On test failure
Network trace	HAR file	On test failure
Browser console log	JSON	On test failure
Playwright trace	`.zip` (trace viewer)	On test failure

---

12. Maintenance Strategy

Page Object Pattern:

- All page interactions wrapped in Page Object classes
- Selectors centralized — change in one place
- Location: `tests/e2e/pages/`

Selector strategy:

1. `data-testid` attributes (preferred — stable, intent-clear)
2. ARIA roles + accessible name (good for accessibility alignment)
3. Text content (acceptable for static text)
4. CSS class names (avoid — coupled to styling)
5. XPath (never — fragile)

Review cadence:

- Test suite reviewed monthly for relevance
- Remove tests for deprecated features immediately
- Update tests before merging feature changes (tests are part of the PR)

Related Documents

- Test Strategy
- Test Plan
- Test Case Template
- CI/CD Pipeline

Approval

Role	Name	Date	Signature
Author			
Reviewer			
Approver			

Revision #6
Created 2026-02-23 12:06:05 UTC by John
Updated 2026-05-25 07:34:13 UTC by John