Test Strategy

Test Strategy

> **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. Testing Philosophy & Principles

Core Principles:

1. **Tests are first-class code** — reviewed, maintained, refactored like production code
2. **Test the behavior, not the implementation** — tests enable safe refactoring
3. **Fast feedback** — unit tests run in seconds; blocking tests run in < 5 minutes
4. **Tests document intent** — a failing test explains what broke and why it matters
5. **Shift left** — find bugs as early as possible, before they reach users

Testing philosophy: {{PHILOSOPHY}}

---

2. Test Pyramid

```
pyramid
    title Test Distribution
    accTitle: Test Distribution by Layer
```

```
    "Unit Tests (70%)" : 70
    "Integration Tests (20%)" : 20
    "E2E Tests (10%)" : 10
```

```
graph TB
    subgraph E2E["E2E Tests — 10%"]
        E2E_DESC["Critical user journeys<br/>Browser / API end-to-end<br/>Slow, expensive, high-confidence"]
    end
    subgraph INT["Integration Tests — 20%"]
        INT_DESC["Service-to-service<br/>DB + cache + external APIs<br/>Medium speed, high-value"]
    end
    subgraph UNIT["Unit Tests — 70%"]
        UNIT_DESC["Functions, classes, modules<br/>Pure business logic<br/>Fast, cheap, developer-owned"]
    end
```

2.1 Unit Tests

Attribute	Value
Scope	Individual functions, classes, pure business logic
External dependencies	Mocked (no real DB, network, or filesystem calls)
Framework	{{UNIT_FRAMEWORK}}
Coverage target	≥ {{UNIT_COVERAGE}}% lines, ≥ {{BRANCH_COVERAGE}}% branches
Execution time	Full suite < {{UNIT_TIME}} minutes
Runs on	Every commit, pre-commit hook
Written by	Developer who writes the feature

What to unit test:

- Business logic and algorithms
- Edge cases and error conditions
- Data transformations and validations
- Utility functions

What NOT to unit test:

- Third-party library internals

- Simple property getters/setters with no logic
- Framework boilerplate

2.2 Integration Tests

Attribute	Value
Scope	Service interactions: DB, cache, message queues, internal APIs
External dependencies	Real (test containers or shared test environment)
Framework	{{INT_FRAMEWORK}}
Coverage target	All service boundaries tested; ≥ {{INT_COVERAGE}}% of integration paths
Execution time	Full suite < {{INT_TIME}} minutes
Runs on	Every PR, blocking merge
Written by	Developer who writes the integration

What to integration test:

- Database queries and ORM mappings
- API endpoint request/response contracts
- Message queue publish/consume
- Cache read/write behavior
- Authentication middleware

2.3 E2E Tests

Attribute	Value
Scope	Critical user journeys through the real deployed application
External dependencies	Real (staging environment)
Framework	{{E2E_FRAMEWORK}}
Coverage target	Top {{E2E_JOURNEY_COUNT}} critical user journeys
Execution time	Full suite < {{E2E_TIME}} minutes
Runs on	Post-staging deploy, pre-production gate
Written by	QA + Developer collaboration

Critical journeys to E2E test:

1. {{JOURNEY_1}}

2. {{JOURNEY_2}}
3. {{JOURNEY_3}}

---

3. Testing Tools

Type	Tool	Version	Purpose	Config File
Unit testing	{{UNIT_TOOL}}	{{VERSION}}	Unit tests	`{{CONFIG_PATH}}`
Test runner	{{RUNNER}}	{{VERSION}}	Parallel execution	`{{CONFIG_PATH}}`
Mocking	{{MOCK_TOOL}}	{{VERSION}}	Mock external deps	Built-in
Integration testing	{{INT_TOOL}}	{{VERSION}}	Integration tests	`{{CONFIG_PATH}}`
Test containers	{{CONTAINER_TOOL}}	{{VERSION}}	Real DB/Redis in tests	`{{CONFIG_PATH}}`
E2E testing	{{E2E_TOOL}}	{{VERSION}}	Browser E2E	`{{CONFIG_PATH}}`
Coverage	{{COV_TOOL}}	{{VERSION}}	Coverage reports	`{{CONFIG_PATH}}`
Performance testing	{{PERF_TOOL}}	{{VERSION}}	Load/stress tests	`{{CONFIG_PATH}}`
Visual regression	{{VIS_TOOL}}	{{VERSION}}	UI screenshot diff	`{{CONFIG_PATH}}`

---

4. Test Environments & Data Management

Test Environments

Test Type	Environment	Database	External Services
Unit	Local (no env)	None / in-memory	Mocked
Integration	Local + Docker	TestContainers	Stubbed
E2E	Staging	Dedicated test DB	Sandbox/test mode
Performance	Staging (production-sized)	Production-scale data	Sandbox

Test Data Management

Approach	Used For	Tool	Notes

Fixtures / factories	Unit + integration tests	{{FACTORY_TOOL}}	Reset per test
Database seeding	E2E tests	`scripts/seed.sh`	Reset per test run
Anonymized production data	Performance tests	`scripts/anonymize.sh`	Weekly refresh
Shared test accounts	E2E (cross-service)	Test account vault	Never modified by tests

Data cleanup policy: All test data cleaned up after test run (via teardown hooks or transaction rollback)

---

5. Test Automation Strategy

Test Type	Automation	Trigger	Tooling
Unit tests	100% automated	Pre-commit + CI	{{UNIT_TOOL}}
Integration tests	100% automated	CI on PR	{{INT_TOOL}}
E2E (critical paths)	100% automated	Post-staging deploy	{{E2E_TOOL}}
E2E (edge cases)	Partially automated	Weekly scheduled run	{{E2E_TOOL}}
Performance tests	Automated baseline	Weekly + pre-release	{{PERF_TOOL}}
Security (SAST/SCA)	100% automated	Every PR	{{SAST_TOOL}}
Visual regression	Automated	On UI changes	{{VIS_TOOL}}
Accessibility	Automated + manual	On UI changes	{{A11Y_TOOL}}
Manual exploratory	Manual	Sprint end / pre-release	—

---

6. Manual Testing Scope

Manual testing is required for:

- New feature exploratory testing (first sprint of a feature)
- Usability and UX review
- Accessibility testing (beyond automated checks)
- Complex multi-step business scenarios not yet automated
- Third-party integrations with limited test environments
- Devices/browsers outside automated matrix

Manual testing is NOT required for:

- Regression of previously tested features (automate these)

- CRUD operations on standard forms
- Unit-level logic covered by automated tests

---

7. Code Coverage Requirements

Layer	Lines	Branches	Functions	Notes
Business logic	≥ {{BIZ_COV}}%	≥ {{BIZ_BRANCH}}%	≥ {{BIZ_FN}}%	Strictly enforced
API handlers	≥ {{API_COV}}%	≥ {{API_BRANCH}}%	≥ {{API_FN}}%	
Utilities	≥ {{UTIL_COV}}%	≥ {{UTIL_BRANCH}}%	≥ {{UTIL_FN}}%	
UI components	≥ {{UI_COV}}%	—	≥ {{UI_FN}}%	Render + interaction
Overall minimum	**≥ {{TOTAL_COV}}%**	—	—	CI gate

Coverage enforcement: CI pipeline fails if coverage drops below minimum **Coverage report:** Published to {{COV_REPORT_URL}} on every PR

---

8. Quality Gates

PR Merge Gate (must pass before merge)

- ☐ All unit tests pass
- ☐ All integration tests pass
- ☐ Coverage ≥ minimum thresholds
- ☐ No HIGH/CRITICAL security findings (SAST/SCA)
- ☐ No secrets detected
- ☐ Linting passes
- ☐ Type checking passes (if typed language)

Staging Deploy Gate (must pass before staging deploy)

- ☐ All PR gates passed

- [ ] Build artifact created and signed

Production Deploy Gate (must pass before production deploy)

- [ ] All E2E tests pass on staging
- [ ] Performance baseline not degraded > {{PERF_REGRESSION}}%
- [ ] Manual QA sign-off (if sprint includes new features)
- [ ] Manual approval in CI pipeline

---

9. Responsibility Matrix

Test Type	Writes Tests	Reviews Tests	Maintains Tests	Signs Off
Unit tests	Developer	PR reviewer	Developer	Tech lead
Integration tests	Developer	QA engineer	Developer	Tech lead
E2E tests	QA + Developer	Tech lead	QA engineer	QA lead
Performance tests	DevOps + Developer	Tech lead	DevOps	Eng manager
Security tests	DevOps (automated)	Security	DevOps	Security lead

---

10. Test Reporting & Metrics

Metric	Target	Reporting
Test pass rate	≥ 99% (unit), ≥ 95% (E2E)	CI dashboard
Flaky test rate	< {{FLAKY_RATE}}%	Weekly report
Test execution time	< {{TOTAL_TEST_TIME}} min (full suite)	CI dashboard
Coverage trend	Stable or improving	PR comments
Tests added per sprint	≥ {{TESTS_PER_SPRINT}}	Sprint metrics

---

11. Continuous Testing in CI/CD

See CI/CD Pipeline for full pipeline details.

Stage	Tests Run	Blocking
Pre-commit (local)	Unit tests, linting	Yes (can be bypassed with `--no-verify` with justification)
PR open / update	Unit + integration + SAST + SCA	Yes — blocks merge
Staging deploy	E2E, visual regression	Yes — blocks production
Production deploy	Smoke tests, monitoring check	Yes — auto-rollback on failure
Scheduled (nightly)	Full E2E suite, performance baseline	No — alerts only

Related Documents

- Test Plan
- E2E Test Plan
- Performance Test Plan
- Definition of Done
- CI/CD Pipeline

Approval

Role	Name	Date	Signature
Author			
Reviewer			
Approver			

Revision #3
Created 2026-02-24 14:53:45 UTC by John
Updated 2026-05-25 07:34:04 UTC by John