Definition of Done

Definition of Done

> **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. Purpose

The Definition of Done (DoD) is a shared agreement across the team on what must be true before any work item can be considered complete. It exists to:

- Prevent technical debt accumulation through rushed or incomplete work
- Ensure consistent quality across all team members and work types
- Create a shared language for "done" that developers, QA, and product all agree on
- Prevent issues from reaching production that could have been caught earlier

Enforcement: The DoD is non-negotiable. Exceptions require explicit sign-off from the Tech Lead and must be tracked as technical debt tickets. Undocumented shortcuts are not acceptable.

---

2. Feature-Level DoD

When is a **feature** done?

Code Quality

- [ ] Code is complete and implements all acceptance criteria from the ticket
- [ ] Code reviewed and approved by at least {{REVIEW_COUNT}} reviewer(s)
- [ ] All review comments resolved or explicitly deferred with documentation
- [ ] No TODO/FIXME comments left without corresponding tickets
- [ ] Code follows coding standards — naming, formatting, patterns
- [ ] No unnecessary dead code added
- [ ] No console.log / debug statements left in production code

Testing

- [ ] Unit tests written for all new business logic (≥ {{UNIT_COV}}% coverage on new code)
- [ ] Integration tests written for all new service boundaries (API endpoints, DB queries, external calls)
- [ ] E2E tests written for any new critical user journeys
- [ ] All existing tests still pass (no regressions introduced)
- [ ] Test coverage has not decreased below project minimum ({{MIN_COV}}%)
- [ ] Edge cases and error conditions tested

Accessibility

- [ ] Accessibility verified — all interactive elements keyboard navigable
- [ ] ARIA attributes correct where needed
- [ ] Color contrast meets WCAG AA (4.5:1 for text, 3:1 for large text)
- [ ] Screen reader compatible (tested with {{SCREEN_READER}})
- [ ] No accessibility regressions (automated scan with {{A11Y_TOOL}} passes)

Security

- [ ] OWASP Top 10 checklist reviewed for this feature
- [ ] Input validation in place for all user inputs
- [ ] Authorization checks correct (user can only access what they're permitted to)
- [ ] No sensitive data logged or exposed in API responses
- [ ] Dependency scan passes (no new HIGH/CRITICAL CVEs introduced)

- [ ] Security code review completed for authentication/authorization changes

Performance

- [ ] Performance budget met — no P95 regression > {{PERF_REGRESSION}}% vs baseline
- [ ] No N+1 query problems introduced
- [ ] Large datasets paginated or streamed (not loaded into memory)
- [ ] Client-side bundle size impact reviewed (if frontend change)

Documentation

- [ ] Code is self-documenting or has inline comments for non-obvious logic
- [ ] README updated if developer setup changed
- [ ] Architecture Decision Record created for significant architectural choices
- [ ] User-facing documentation updated (if applicable)

API (if applicable)

- [ ] API documentation updated (OpenAPI spec / Postman collection)
- [ ] Breaking changes explicitly documented and versioning strategy applied
- [ ] Backwards compatibility maintained or migration guide provided

Error Handling

- [ ] All error conditions handled gracefully
- [ ] Error messages are user-friendly (no stack traces in production)
- [ ] Errors logged with appropriate severity and context
- [ ] External service failures handled with circuit breakers / fallbacks

Observability

- [ ] Logging in place for significant events (user actions, errors, integrations)
- [ ] Metrics emitted for new features (request rate, error rate, business events)
- [ ] Alerts configured for new failure modes (if applicable)
- [ ] Traces instrumented for new service calls

Feature Flags

- [ ] Feature flag configured for the feature (if it's a significant or risky change)
- [ ] Flag documented in feature flag register
- [ ] Rollback plan documented (flag can be disabled to revert)

Deployment

- [ ] Deployed to staging and manually verified
- [ ] Database migrations tested on staging (up and down)
- [ ] Environment variables documented for any new configuration
- [ ] Product Owner has reviewed and accepted the feature in staging

---

3. Sprint-Level DoD

When is a **sprint** done?

- [ ] All committed tickets meet their individual Definition of Done
- [ ] All automated tests passing in CI (unit, integration, E2E)
- [ ] Staging environment is stable (no broken features)
- [ ] Technical debt created during the sprint is tracked in the backlog
- [ ] Sprint review conducted — stakeholders have seen and accepted deliverables
- [ ] Retrospective conducted — process improvements identified and documented
- [ ] Next sprint's tickets are refined and estimated
- [ ] Performance baseline checked — no significant regressions vs last sprint

---

4. Release-Level DoD

When is a **release** done?

- [ ] All features in scope meet the Feature-Level DoD
- [ ] Full regression test suite passing on staging
- [ ] E2E tests passing for all critical user journeys
- [ ] Performance test passing — all SLAs met under load

- [ ] Security scan complete — no unresolved HIGH/CRITICAL findings
- [ ] Accessibility audit complete for new/changed UI
- [ ] UAT conducted and sign-off obtained from Product Owner
- [ ] Release notes written and reviewed
- [ ] Deployment checklist complete (see deployment-checklist.md)
- [ ] Rollback plan prepared and tested
- [ ] On-call engineer briefed on changes and potential failure modes
- [ ] Monitoring dashboards updated for new features
- [ ] All critical and high defects resolved or explicitly accepted with justification

---

5. Bug Fix DoD

When is a **bug fix** done?

- [ ] Root cause understood and documented in the ticket
- [ ] Fix addresses root cause (not just symptoms)
- [ ] Unit/integration test written that would have caught this bug (regression test)
- [ ] Fix does not introduce new failures (all tests passing)
- [ ] Fix verified in the environment where the bug was reported
- [ ] Related areas spot-checked for similar issues
- [ ] Bug ticket updated with root cause and resolution notes

---

6. Hotfix DoD

When is a **hotfix** done?

- [ ] Fix verified locally and on staging
- [ ] At least 1 reviewer has approved (no solo hotfixes)
- [ ] Smoke tests passing post-deployment
- [ ] Incident ticket updated with the fix details
- [ ] Full regression test run scheduled within {{REGRESSION_SLA}} hours
- [ ] Post-mortem scheduled if P1/P2 severity

---

7. Customization Guide

How to adapt this DoD:

1. Review each checklist item — mark N/A if genuinely not applicable to your project type
2. Add project-specific items in the appropriate section
3. Adjust thresholds (coverage %, performance budget) to match your NFRs
4. Get explicit agreement from all team members before finalizing
5. Review and update the DoD at each sprint retrospective

Common adaptations:

- Mobile apps: Add device testing matrix, app store submission requirements
- Backend-only services: Remove/simplify accessibility and UI performance items
- Highly regulated environments: Add compliance checkboxes (GDPR, HIPAA, etc.)
- Startups / early stage: Simplify coverage requirements while maintaining security basics

---

Related Documents

- Test Strategy
- Coding Standards
- Deployment Checklist
- UAT Sign-off

Approval

Role	Name	Date	Signature
Author			
Reviewer			
Approver			

Revision #3
Created 2026-02-24 14:53:55 UTC by John
Updated 2026-05-25 07:34:19 UTC by John