TECH DEBT ASSESSMENT CHECKLIST

A Systematic Approach to Identifying and Prioritizing Technical Debt

For Tech Leads and Senior Engineers

ASSESSMENT AREAS

- Code Quality - smells, complexity, and standards
- Architecture & Dependencies - health and risk
- Testing & Documentation - coverage and gaps
- Infrastructure & Process - CI/CD and workflows
- Scoring & Priority Matrix - quantify your debt

TechDebt.works

https://techdebt.works/

Code Quality Assessment

Check each item that applies to your codebase. The more items checked, the higher your technical debt.

Code Smells & Complexity

- [ ] Cyclomatic complexity exceeds team threshold in critical paths
- [ ] Duplicated code blocks across 3+ locations
- [ ] Methods/functions exceeding 50 lines regularly
- [ ] God classes or modules with 10+ responsibilities
- [ ] Dead code not removed after feature flags retired

Code Standards

- [ ] Inconsistent naming conventions across modules
- [ ] Missing or outdated type definitions
- [ ] Suppressed linter warnings without justification
- [ ] Hard-coded configuration values in source

Architecture & Dependencies

Architecture Health

- [ ] Circular dependencies between modules
- [ ] Layer violations (UI calling DB directly, etc.)
- [ ] Inconsistent patterns for same concern (3+ ways to do auth, logging, etc.)
- [ ] Missing or outdated architecture decision records (ADRs)
- [ ] Services with unclear ownership boundaries

Dependency Health

- [ ] Dependencies with known CVEs (check npm audit / Snyk)
- [ ] Major version behind on framework (2+ major versions)
- [ ] Abandoned dependencies (no updates in 12+ months)
- [ ] Pinned versions preventing security patches
- [ ] Missing lock file or inconsistent lock file

Testing & Documentation

Test Coverage

- [ ] Unit test coverage below 60% on critical paths
- [ ] No integration tests for API endpoints
- [ ] No E2E tests for critical user flows
- [ ] Tests that are flaky (fail randomly 5%+ of the time)
- [ ] Tests that mock everything (testing mocks, not code)

Documentation

- [ ] Missing README for onboarding new team members
- [ ] API documentation out of sync with implementation
- [ ] Missing runbooks for production incidents
- [ ] No changelog or release notes process
- [ ] Tribal knowledge not captured anywhere

Infrastructure & Process

CI/CD & DevOps

- [ ] Build takes longer than 10 minutes
- [ ] No automated deployment pipeline
- [ ] Manual steps required for releases
- [ ] No rollback procedure documented/tested
- [ ] Missing health checks or monitoring alerts

Process Debt

- [ ] No code review requirements enforced
- [ ] Technical decisions made without documentation
- [ ] No regular dependency update schedule
- [ ] Incident post-mortems not conducted or not actioned
- [ ] No tech debt tracking system (backlog, labels, etc.)

Scoring & Priority Matrix

Score each item: 0 = Not applicable, 1 = Minor, 2 = Moderate, 3 = Critical

Category	Items	Score
Code Quality	9 items	___ / 27
Architecture	5 items	___ / 15
Dependencies	5 items	___ / 15
Testing	5 items	___ / 15
Documentation	5 items	___ / 15
Infrastructure	5 items	___ / 15
Process	5 items	___ / 15
TOTAL	**39 items**	**___ / 117**

Scoring Guide

0 - 25: Low debt - Maintain current practices

26 - 50: Moderate debt - Allocate 10% sprint capacity

51 - 75: High debt - Allocate 20% sprint capacity, create remediation plan

76+: Critical debt - Executive escalation needed, dedicate team

Full resources at https://techdebt.works/