Tech Debt Log

Tech Debt Log: {{PROJECT_NAME}}

> **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. Tech Debt Philosophy

Technical debt is not inherently bad. Sometimes taking on debt is the right business decision (shipping faster to meet a deadline). The problem is **unacknowledged and unmanaged** debt.

This log exists to:

1. Make invisible debt visible
2. Enable informed decisions about when to repay debt
3. Track the "interest" — how debt is slowing the team down
4. Demonstrate to stakeholders that quality is actively managed

Debt Entry Trigger: Any time you make a technical shortcut, write a workaround, defer a refactor, skip a test, or make a "we'll fix this later" comment — add an entry here.

---

2. Summary Dashboard

Metric	Current	Last Sprint	Trend
Total Items	{{COUNT}}	{{COUNT}}	{{↑/↓/→}}
Open	{{COUNT}}	{{COUNT}}	{{↑/↓/→}}
In Progress	{{COUNT}}	{{COUNT}}	{{↑/↓/→}}
Resolved	{{COUNT}}	{{COUNT}}	{{↑/↓/→}}
Critical / High (P1/P2)	{{COUNT}}	{{COUNT}}	{{↑/↓/→}}
Debt in current sprint	{{COUNT}}	{{COUNT}}	{{↑/↓/→}}
Avg age of open items (days)	{{DAYS}}	{{DAYS}}	{{↑/↓/→}}
Debt resolved this sprint	{{COUNT}}	{{COUNT}}	{{↑/↓/→}}

Sprint Debt Budget: {{X}}% of sprint capacity allocated to debt reduction (recommended: 15-20%)

---

3. Tech Debt Categories

Category	Description	Common Causes	Detection Method
Design	Architectural shortcuts, wrong abstractions, poor separation of concerns	Time pressure, evolving requirements	Code review, architecture review
Code Quality	Duplication, high complexity, poor naming, missing documentation	Speed over quality, lack of standards	Static analysis (SonarQube), code review
Infrastructure	Manual processes, missing monitoring, config drift, single points of failure	Growth outpacing ops maturity	Ops review, incident postmortems
Documentation	Missing or outdated technical docs, undocumented decisions, no onboarding guide	Documentation deferred, never caught up	New team member feedback, ops incidents
Dependency	Outdated libraries, deprecated APIs, unmaintained packages, version conflicts	Insufficient update cadence	Automated scan (Snyk/Dependabot)
Testing	Low coverage, flaky tests, missing integration tests, untested edge cases	Speed over quality	Coverage reports, incident analysis

Category	Description	Common Causes	Detection Method
Security	Known vulnerabilities not yet patched, security shortcuts taken	Urgency, lack of security knowledge	Security scans, CVE alerts
Performance	Known bottlenecks, inefficient queries, N+1 problems, no caching	Optimizations deferred	Performance monitoring, load tests

---

4. Impact Assessment Methodology

4.1 Impact Score (1–10)

Score	Impact Level	Effect on Team/Business
1–2	Negligible	Cosmetic; no measurable effect on velocity or product
3–4	Minor	Slightly slows specific tasks; <5% velocity impact
5–6	Moderate	Slows multiple tasks; 5–15% velocity impact; occasional bugs
7–8	High	Significantly slows development; 15–30% velocity impact; recurring bugs
9–10	Critical	Blocks progress; >30% velocity impact; high risk of production incidents

4.2 Effort to Fix (Story Points)

Size	Points	Typical Scope
XS	1	Single line / config change
S	2–3	Single function / component refactor
M	5–8	Module-level changes; cross-cutting update
L	13	Multi-module refactoring; architecture change
XL	21+	Major refactor; break into sub-tasks first

4.3 Priority Calculation

Priority = Impact Score × (1 / Effort) — High impact, low effort = P1

Priority	Criteria	Sprint Allocation Target
P1 — Critical	Impact ≥ 8 OR blocks release	Schedule within 1 sprint
P2 — High	Impact 6–7	Schedule within 2 sprints
P3 — Medium	Impact 4–5	Schedule within 4 sprints
P4 — Low	Impact ≤ 3	Schedule when convenient

---

5. Tech Debt Register

ID	Description	Category	Introduced	Impact (1-10)	Effort	Priority	Owner	Status	Target Resolution	Review Date	Notes
TD-001	{{DESCRIPTION}}	{{CATEGORY}}	{{DATE_OR_SPRINT}}	{{1-10}}	{{XS/S/M/L/XL}}	P{{1-4}}	{{@OWNER}}	Open	{{DATE}}	{{DATE}}	{{CONTEXT}}
TD-002		Code Quality						Open			
TD-003		Testing						In Progress			
TD-004		Architecture						Open			
TD-005		Dependencies						Open			
TD-006		Infrastructure						Open			
TD-007		Documentation						Open			
TD-008		Security						Open			
TD-009		Performance						Open			

Status Values: Open | In Progress | Resolved | Accepted (deferred) | Won't Fix (justified)

6. Prioritization Framework

6.1 Cost of Delay Analysis

ID	Description	Cost Per Sprint (velocity loss %)	Number of Sprints Open	Accumulated Cost
TD-001	{{DESCRIPTION}}	{{X}}%	{{COUNT}}	{{ACCUMULATED_COST}}

6.2 Debt Payoff Prioritization

When multiple P3 items compete for sprint capacity, use:

1. **Quick wins first** — Impact ≥ 5 AND Effort ≤ S (high ROI)
2. **Blocker debt** — Any item that blocks a planned feature
3. **Aging debt** — Items open > 3 sprints get auto-elevated one priority
4. **Security debt** — Always scheduled within 1 sprint regardless of impact score

---

7. Sprint Debt Allocation Guidelines

Team Velocity	Debt Budget (15%)	Recommended Items to Schedule
40 points/sprint	6 points	2–3 S items or 1 M item
60 points/sprint	9 points	3–4 S items or 1 M + 1 S
80 points/sprint	12 points	1 L item or 2 M items

Sprint Debt Review (each sprint planning):

1. PM + Tech Lead review open debt register (15 min)
2. Select debt items for sprint based on budget and priority
3. Add selected items to sprint backlog as technical tasks
4. Update status of in-progress items

---

8. Debt Reduction Tracking

Sprint	Debt Items Added	Debt Items Resolved	Net Change	Running Total Open
Sprint 1	{{COUNT}}	{{COUNT}}	{{+/-}}	{{COUNT}}
Sprint 2				
Sprint 3				

Debt Trend: {{INCREASING / STABLE / DECREASING}}

---

9. Trend Analysis

9.1 Debt by Category (this quarter)

Category	Items Added	Items Resolved	Net
Code Quality	{{COUNT}}	{{COUNT}}	{{+/-}}
Testing			
Architecture			
Dependencies			
Infrastructure			
Documentation			
Security			
Performance			

9.2 Root Cause Analysis

Dominant category this quarter: {{CATEGORY}} **Suspected root cause:** {{ROOT_CAUSE}}
Proposed systemic fix: {{FIX — hook/tool/process/rule}}

---

10. Resolved / Accepted Debt Archive

ID	Description	Resolution	Resolution Date	Sprint	Resolved By
TD-{{RESOLVED}}	{{DESCRIPTION}}	{{HOW_FIXED}}	{{DATE}}	Sprint {{X}}	{{@OWNER}}

Approval

Role	Name	Date	Signature
Author			
Reviewer			
Tech Lead			
AI Director (John)			

Revision #3
Created 2026-02-24 14:54:36 UTC by John
Updated 2026-05-25 07:35:02 UTC by John