Tech Debt Log: Drop — Fintech Payment App

Tech Debt Log: Drop — Fintech Payment App

> **Project:** Drop — Remittance + QR Payments **Version:** 1.0 **Date:** 2026-02-23
> **Author:** John (AI Director) **Status:** Approved **Reviewers:** Alem Bašić (CEO)

Document History

Version	Date	Author	Changes
0.1	2026-02-23	John	Initial tech debt log — Phase 0/0.5 items captured

---

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.

Drop-Specific Constraint: Drop is a fintech pass-through app under PSD2/AML regulation. Security and compliance debt (categories: Security, Design) get automatic P1 treatment regardless of impact score. No security debt survives more than 1 sprint without explicit CEO risk acceptance.

---

2. Summary Dashboard

Metric	Current	Last Sprint	Trend
Total Items	9	3	↑
Open	6	3	↑
In Progress	1	0	↑
Resolved	2	0	↑
Critical / High (P1/P2)	3	2	→
Debt in current sprint	1	0	↑
Avg age of open items (days)	14	14	→
Debt resolved this sprint	2	0	↑

Sprint Debt Budget: 20% of sprint capacity allocated to debt reduction (Phase 0.5 security hardening sprint exceeded this — security debt is P1 mandatory)

---

2. 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, 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Documentation	Missing or outdated technical docs, undocumented decisions, no onboarding guide	Documentation deferred, never caught up	New team member feedback, ops incidents

Category	Description	Common Causes	Detection Method
Dependency	Outdated libraries, deprecated APIs, unmaintained packages, version conflicts	Insufficient update cadence	Automated scan (`npm audit`)
Testing	Low coverage, flaky tests, missing integration tests, untested edge cases	Speed over quality	Coverage reports, incident analysis
Security	Known vulnerabilities not yet patched, security shortcuts taken	Urgency, lack of security knowledge	Security scan, CVE alerts
Performance	Known bottlenecks, inefficient queries, N+1 problems, no caching	Optimizations deferred	`api-benchmarks.test.ts`, load tests

---

3. Impact Assessment Methodology

3.1 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

3.2 Effort to Fix (Story Points)

Size	Points	Typical Scope
XS	1	Single line / config change
S	2–3	Single function / component refactor

Size	Points	Typical Scope
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

3.3 Priority Calculation

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

Priority	Criteria	Sprint Allocation Target
P1 — Critical	Impact ≥ 8 OR blocks release OR any Security debt	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

---

4. Tech Debt Register

ID	Description	Category	Introduced	Impact (1-10)	Effort	Priority	Owner	Status	Target Resolution	Review Date	Notes

TD-001	SQLite has ~200 concurrent write limit. Under load, users may experience `database is locked` errors. Migration to PostgreSQL (Phase 1) is the full fix.	Performance	Phase 0	8	L	P1	John	Open	Phase 1 launch	2026-03-01	NFR-S02 documents this. SQLite WAL mode mitigates but does not eliminate. Blocks production scale.
TD-002	BaaS integration mocked via `NEXT_PUBLIC_SERVICE_MODE=mock`. No real SpareBank1 or Swan connection. All transactions are simulated.	Design	Phase 0	10	XL	P1	John	Open	Phase 2 (BaaS partner confirmed)	2026-03-01	Blocks real money movement. BaaS partner pitch in progress. Cannot launch for real users until resolved.

TD-003	KYC integration uses mock Sumsub. User identity verification is not real — any user can pass KYC in dev/staging.	Security	Phase 0	10	L	P1	John	Open	Phase 2 (Sumsub live integration)	2026-03-01	Blocks regulatory compliance. MUST be live before real user onboarding. Sumsub account manager contact transferred to John.
TD-004	In-memory rate limiting (Node process memory) resets on server restart. Was Phase 0 implementation.	Security	Phase 0	7	M	P1	Builder Agent	Resolved	Phase 0.5 (resolved)	2026-02-23	**RESOLVED in Phase 0.5** — replaced with DB-backed rate limiting using `rate_limit_requests` table. Per-user + per-IP persistence survives restarts.

TD-005	No bcrypt password length limit — malicious users could submit 10,000-character passwords and trigger bcrypt DoS (bcrypt has O(n) cost on input length).	Security	Phase 0	7	XS	P1	Builder Agent	Resolved	Phase 0.5 (resolved)	2026-02-23	**RESOLVED in Phase 0.5** — 1,000-character max input validation added in Zod schema before bcrypt. Regression test added.
TD-006	Exchange rates are static seed data. No real rate feed from external provider. Rates in `db/seeds/rates.sql` do not auto-update.	Design	Phase 0	6	M	P2	John	Open	Phase 2	2026-04-01	Acceptable for MVP/demo. Phase 2 requires live rate feed (e.g., Open Exchange Rates API or Wise API).

TD-007	BankID integration fully mocked. Age verification (≥18) and Norwegian residency checks use mock data only.	Security	Phase 0	9	L	P1	John	Open	Phase 1 or Phase 2	2026-03-01	Regulatory requirement. BankID = mandatory per Norwegian PSD2 + AML. Cannot serve real users without it. Finanstilsynet registration in progress.
TD-008	No structured logging / APM in place. `console.log` used for debug output in some modules. No Fly.io metrics integration beyond basic health check.	Infrastructure	Phase 0	5	M	P3	John	Open	Phase 1	2026-04-01	Low risk for MVP/demo. Phase 1 needs structured logging (e.g., Pino) + Fly.io metrics + Sentry error tracking.

TD-009	Cards feature has no real card partner integration. Feature is behind `CARDS_ENABLED=false` flag — cards data is mocked, no real card issuing.	Design	Phase 0	3	XL	P4	John	Open	Phase 3+	2026-06-01	Low priority. Cards are explicitly Phase 3. Feature flag prevents exposure. No action needed for Phase 0.5 or Phase 1.

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

---

5. Prioritization Framework

5.1 Cost of Delay Analysis

ID	Description	Cost Per Sprint (velocity loss %)	Number of Sprints Open	Accumulated Cost
TD-001	SQLite concurrent limit	~5% (affects test reliability under concurrency)	1	~5% velocity
TD-002	Mock BaaS	Blocks real revenue (100% blocker for live transactions)	1	Cannot launch
TD-003	Mock KYC	Blocks regulatory compliance (100% blocker for real users)	1	Cannot launch
TD-007	Mock BankID	Blocks age verification + AML compliance (100% blocker)	1	Cannot launch

5.2 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 or regulatory milestone
3. **Aging debt** — Items open > 3 sprints get auto-elevated one priority
4. **Security debt** — Always scheduled within 1 sprint regardless of impact score

Drop Rule: TD-002, TD-003, TD-007 (BaaS, KYC, BankID) are NOT fixed by code changes alone — they require confirmed external partners. Track separately as business dependencies, not engineering tasks.

---

6. Sprint Debt Allocation Guidelines

Team Velocity	Debt Budget (20%)	Recommended Items to Schedule
20 points/sprint (AI agents)	4 points	1–2 S items or 1 M item
40 points/sprint	8 points	2–3 S items or 1 M item
60 points/sprint	12 points	1 L item or 2 M items

Sprint Debt Review (each sprint planning):

1. John (AI Director) reviews open debt register (15 min)
2. Select debt items based on budget and priority
3. Add selected items to sprint backlog as technical tasks in Mission Control
4. Update status of in-progress items

---

7. Debt Reduction Tracking

Sprint	Debt Items Added	Debt Items Resolved	Net Change	Running Total Open
Phase 0	9	0	+9	9
Phase 0.5	0	2 (TD-004, TD-005)	-2	7
Phase 1	TBD	TBD (TD-001, TD-008 targets)	TBD	TBD

Debt Trend: DECREASING (Phase 0.5 resolved 2 security items; 3 blockers are external business dependencies, not engineering debt)

8. Trend Analysis

8.1 Debt by Category (Phase 0 ? Phase 0.5)

Category	Items Added	Items Resolved	Net
Design	3 (TD-002, TD-006, TD-009)	0	+3
Security	4 (TD-003, TD-004, TD-005, TD-007)	2 (TD-004, TD-005)	+2
Infrastructure	1 (TD-008)	0	+1
Performance	1 (TD-001)	0	+1
Code Quality	0	0	0
Testing	0	0	0
Documentation	0	0	0
Dependency	0	0	0

8.2 Root Cause Analysis

Dominant category this quarter: Security + Design (MVP shortcuts) **Suspected root cause:** MVP phase deliberately chose mock integrations (BaaS, KYC, BankID) to ship working architecture without blocked external partners. This is intentional, managed debt — not negligence. **Proposed systemic fix:** Business dependency tracking separate from engineering debt. TD-002/003/007 are tracked in the ROADMAP.md as Phase 2 blockers and require Alem Bašić (CEO) partner decisions, not engineering sprints.

---

9. Resolved / Accepted Debt Archive

ID	Description	Resolution	Resolution Date	Sprint	Resolved By
TD-004	In-memory rate limiting resets on restart	Replaced with DB-backed `rate_limit_requests` table; persistent across restarts	2026-02-23	Phase 0.5	Builder Agent

ID	Description	Resolution	Resolution Date	Sprint	Resolved By
TD-005	No password length limit — bcrypt DoS vector	Added Zod `max(1000)` validation before bcrypt hash; regression test in `auth.test.ts`	2026-02-23	Phase 0.5	Builder Agent

Related Documents

- Risk Register
- Security Audit Report
- Lessons Learned
- Architecture Decision Records

Approval

Role	Name	Date	Signature
Author	John (AI Director)	2026-02-23	Approved (AI)
Tech Lead	John	2026-02-23	Approved
CEO (Alem)	Alem Bašić	TBD	

Revision #5
Created 2026-02-23 12:06:53 UTC by John
Updated 2026-05-31 20:03:36 UTC by John