Cross-Cutting

Tech debt log, change requests, lessons learned

- Tech Debt Log: Drop — Fintech Payment App
- Change Request Template: Drop — Fintech Payment App
- Lessons Learned: Drop — Fintech Payment App
- Tech Debt Log
- Change Request
- Lessons Learned

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	

Change Request Template: Drop — Fintech Payment App

Change Request Template: Drop — Fintech Payment App

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

Document History

Version	Date	Author	Changes
0.1	2026-02-23	John	Initial CR template — Drop fintech context

---

How to Use This Template

A Change Request (CR) is submitted whenever a stakeholder wants to modify any approved project artifact — scope, timeline, budget, requirements, or design.

EVERY change must go through this process, no matter how "small" it seems. In fintech, one "small" change (e.g., adding a column to the `users` table) can violate PCI-DSS compliance and trigger a P0 incident.

Copy this template to `docs/CROSS-CUTTING/change-requests/CR-{XXX}-{short-title}.md` and fill in all fields.

Drop-Specific Rules:

- Any schema change to `users` or `cards` tables requires explicit compliance review — pass-through model (ADR-003) must not be violated
- Any change to fee rates (currently 0.5% remittance, 1% QR) requires CEO sign-off from Alem Bašić
- Any change affecting authentication, rate limiting, or JWT handling requires security review by John + Validator Agent
- Changes that affect Finanstilsynet registration scope must be reviewed before submission

---

Change Request Log

CR ID	Title	Status	Submitted	Decision Date	Impact
CR-001	Phase 0.5 Security Hardening Scope	Approved	2026-02-23	2026-02-23	In-scope security fixes; no fee/schema changes
CR-002	(use template below)				

[TEMPLATE STARTS HERE — Copy below for each new CR]

Change Request: {CR_TITLE}

1. Change Request Metadata

Field	Value
CR ID	CR-{XXX} *(assigned by John as AI Director)*
Date Submitted	{DATE}
Submitted By	{NAME} — {ROLE}
Project	Drop — Remittance + QR Payments
Priority	Critical / High / Medium / Low

Field	Value
CR Type	Scope Change / Budget Change / Timeline Change / Requirements Change / Design Change / Technical Change
Current Phase	Phase 0.5 Security Hardening / Phase 1 BaaS Integration / Phase 2 Compliance
Decision Deadline	{DATE} *(by when a decision must be made to avoid impact)*

---

2. Change Description

2.1 Summary of Change

{ONE_PARAGRAPH_SUMMARY_OF_WHAT_CHANGES}

2.2 Current State (Before)

{DESCRIPTION_OF_CURRENT_STATE}

Currently approved in: `{DOCUMENT_NAME_AND_VERSION}`, Section {X.X}

Drop compliance check — Current state:

- Does the current state involve the `users.balance` column? Yes / No — (Must remain: No)
- Does the current state involve `cards.card_number` or `cards.cvv`? Yes / No — (Must remain: No)
- Does the current state involve fee rates? Yes / No — (Current: remittance 0.5%, QR 1%)

2.3 Proposed State (After)

{DESCRIPTION_OF_PROPOSED_STATE}

Drop compliance check — Proposed state:

- Will the proposed state add a `balance` column to `users`? Yes / No — (If Yes: BLOCKED — violates ADR-003)
- Will the proposed state store `card_number` or `cvv` in full? Yes / No — (If Yes: BLOCKED — violates PCI-DSS)
- Does this change fee rates? Yes / No — (If Yes: CEO sign-off required)

2.4 Out of Scope for This Change

This CR does NOT include:

- {EXPLICITLY_EXCLUDED_ITEM_1}
- {EXPLICITLY_EXCLUDED_ITEM_2}

---

3. Reason & Justification

3.1 Reason for Change

Reason Category	Applies?	Details
New business requirement discovered	Yes / No	{DETAILS}
Regulatory / compliance mandate	Yes / No	{DETAILS}
Client feedback from UAT / prototype	Yes / No	{DETAILS}
Technical blocker / infeasibility	Yes / No	{DETAILS}
Market opportunity / competitive pressure	Yes / No	{DETAILS}
Error/omission in original requirements	Yes / No	{DETAILS}
Performance / quality improvement	Yes / No	{DETAILS}
Security finding from audit	Yes / No	Reference: security/drop-security-rapport.md SEC-{ID}

Primary Justification: {CLEAR_BUSINESS_JUSTIFICATION_WHY_THIS_CHANGE_IS_NECESSARY}

3.2 Consequence of Not Changing

If this change is **not** approved: {CONSEQUENCE_OF_REJECTION}

---

4. Impact Analysis

4.1 Scope Impact

Deliverable Affected	Current Scope	Proposed Scope	Impact Type
{DELIVERABLE}	{CURRENT}	{PROPOSED}	Added / Removed / Modified

Scope Change Size: Small (< 1 day) / Medium (1–3 days) / Large (3–10 days) / Major (> 10 days)

4.2 Timeline Impact

Milestone	Current Date	New Date (if approved)	Delay
Phase 0.5 Security Hardening	2026-02-28	{NEW_DATE}	{DAYS} days
Phase 1 BaaS Integration	2026-04-30	{NEW_DATE}	{DAYS} days
Finanstilsynet Registration	2026-05-31	{NEW_DATE}	{DAYS} days

Timeline Impact: None / Minor (≤ 3 days) / Moderate (4–14 days) / Major (> 14 days) **Critical Path Impact:** Yes / No If yes: {WHICH_CRITICAL_PATH_ITEMS_AFFECTED}

4.3 Budget Impact

Cost Category	Current Budget (NOK)	Additional Cost (NOK)	Notes
Development	150,000 (Innovasjon Norge + bootstrap)	{ADDITIONAL}	{NOTES}
Design	—	{ADDITIONAL}	
Testing	—	{ADDITIONAL}	
Infrastructure (Fly.io)	—	{ADDITIONAL}	
Total Additional Cost		**{TOTAL_ADDITIONAL}**	

Budget Impact: None / Minor (< 5%) / Moderate (5–15%) / Major (> 15%) **Total Drop budget:** ~250,000 NOK (150K Innovasjon Norge + bootstrap) **Funding Source for Additional Cost:** {HOW_WILL_ADDITIONAL_COST_BE_COVERED}

4.4 Resource Impact

Resource	Current Allocation	Required if Approved	Impact
Builder Agent (Claude Sonnet)	Per-task	{NEW_ALLOCATION}	{NOTES}
Validator Agent (Claude Sonnet)	Per-review	{NEW_ALLOCATION}	
John (AI Director)	Architecture + coordination	{NEW_ALLOCATION}	

4.5 Risk Impact

Risk	Probability	Impact	Notes
{NEW_RISK_INTRODUCED}	H/M/L	H/M/L	
Pass-through model violation risk	L (if schema touched)	Critical	Always assess for schema changes

4.6 Quality Impact

- Test cases affected: {LIST_OF_TEST_CASES_NEEDING_UPDATE}
- Regression risk: {HIGH/MEDIUM/LOW} — {EXPLANATION}
- NFRs affected: {LIST_ANY_NFRs_IMPACTED}
- `db.test.ts` pass-through assertions: Will these still pass? Yes / Needs update — {EXPLANATION}
- `api-endpoints.test.ts`: Will existing 26 API endpoint tests still pass? Yes / Needs update

4.7 Affected Deliverables / Documents

Document	Section	Type of Change	Owner
`docs/backend/API-REFERENCE.md`	{SECTION}	Update / Add / Remove	John
`docs/backend/DATABASE-SCHEMA.md`	{SECTION}	Update / Add / Remove	John
`CLAUDE.md`	{SECTION}	Update	John
`docs/BUSINESS-REQUIREMENTS/functional-requirements.md`	FR-{XXX}	Modify	John
Test cases	TC-{XXX}	Update	Builder Agent + Validator Agent

---

5. Alternative Approaches Considered

Alternative	Description	Why Rejected
Option A (Proposed)	{THIS_CR}	*Recommended*
Option B	{ALTERNATIVE}	{WHY_NOT_CHOSEN}
Option C — Do Nothing	Reject the change	{CONSEQUENCE_OF_REJECTION}

Recommendation: Option {A/B/C} **Rationale:** {WHY_THIS_IS_THE_BEST_OPTION}

---

6. Implementation Plan

6.1 Implementation Steps

#	Task	Owner	Effort	Target Date
1	{TASK}	Builder Agent	{EFFORT}	{DATE}
2	Update `db.test.ts` if schema changes	Builder Agent	S	{DATE}
3	Update test cases for affected features	Builder Agent + Validator Agent	{EFFORT}	{DATE}
4	Update requirements and API reference docs	John	{EFFORT}	{DATE}
5	Regression testing (`npm run test` + `npx playwright test`)	Validator Agent	M	{DATE}
6	Deploy to staging (`https://drop-staging.fly.dev/`) and verify	Builder Agent	S	{DATE}

6.2 Dependencies

Dependency	Type	Blocking?
{DEPENDENCY}	Internal / External	Yes / No
Fly.io staging environment	Infrastructure	No (always available)
BaaS partner confirmation (for Phase 2 changes)	External	Yes (for live money movement)

6.3 Test Plan for This Change

- Unit tests: {WHICH_UNITS_NEED_NEW/UPDATED_TESTS} — add to relevant `__tests__/*.test.ts`
- Integration tests: {WHAT_INTEGRATIONS_TO_RETEST} — `api-endpoints.test.ts`
- DB compliance: `db.test.ts` must still pass (no balance, no CVV)
- Regression scope: Full `npm run test` (40+ tests) + `npx playwright test` (3 projects)
- UAT: {DOES_THIS_REQUIRE_CEO_UAT? Y/N — WHICH_SCENARIOS}

---

7. Rollback Plan

Rollback Trigger: {WHAT_CONDITION_TRIGGERS_ROLLBACK} *(e.g., error rate > 1% post-deploy, smoke tests failing, pass-through model violation detected)*

Rollback Steps:

1. `flyctl deploy --app drop-app --image registry.fly.io/drop-app:{PREVIOUS_VERSION}` (2–5 min)
2. Verify health: `curl https://drop-staging.fly.dev/api/health`
3. Run smoke tests: `npx playwright test --project=user-flows`
4. If DB migrations ran: assess whether down migration is safe (Phase 0.5 migrations are all additive — generally safe to leave tables in place)
5. Update Mission Control incident task with rollback details

Rollback Owner: John (AI Director) **Rollback Time Required:** 5–10 minutes **Data Recovery Needed:** No (mock BaaS — no real transactions in Phase 0.5)

---

8. Approval Workflow

8.1 Approval Matrix — Drop

Impact Type	Required Approvals	Target Decision Time
No budget/timeline impact	John (AI Director)	1 business day
Schema change (any)	John + Validator Agent compliance check	1 business day
Fee rate change	John + Alem Bašić (CEO)	2 business days
Budget impact < 5% OR timeline < 3 days	John + Alem Bašić	2 business days
Budget impact 5–15% OR timeline 3–14 days	John + Alem Bašić (CEO)	3 business days
Budget impact > 15% OR timeline > 14 days	John + Alem Bašić (CEO) + Board	5 business days
Finanstilsynet registration scope change	John + Alem Bašić + Legal review	5 business days

8.2 This Change Requires

☐ **Tech Lead Review** — Technical feasibility and effort confirmed (John)

☐ **Validator Agent Review** — DB compliance check: no balance/CVV violation

☐ **John (AI Director)** — Architecture accountability + Mission Control task created

☐ **Alem Bašić (CEO)** — Fee rate changes, budget > 5%, or scope changes affecting BaaS/Finanstilsynet

8.3 Decision Record

Level	Reviewer	Decision	Date	Comments
Tech Lead	John	Approved / Rejected / Deferred	{DATE}	{COMMENTS}
Validator Agent	Validator	Approved / Rejected	{DATE}	DB compliance: pass-through model intact?
AI Director	John	Approved / Rejected	{DATE}	
CEO (Alem)	Alem Bašić	Approved / Rejected	{DATE}	*(required for fee/budget/scope changes)*

Final Decision: APPROVED / REJECTED / DEFERRED **Decision Date:** {DATE} **Effective From Sprint:** Phase {X.X} / Sprint {X}

---

9. Change Log

Date	Changed By	What Changed
{DATE}	{NAME}	{WHAT_CHANGED}

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	

Lessons Learned: Drop — Fintech Payment App

Lessons Learned: 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 lessons learned — Phase 0 + Phase 0.5 Security Hardening

---

> **"Lessons learned without behavior change is not learning."** — Alem Bašić (CEO), 2026-02-13 (ZAKON #0)

Every lesson in this document must produce either a system fix (hook / tool / rule / ADR) or a knowledge base contribution. If a lesson has no action item, it does not belong here — it belongs in a complaint log.

---

1. Session Metadata

Field	Value
Session Type	Phase Review (Phase 0 → Phase 0.5)

Field	Value
Project	Drop — Remittance + QR Payments
Phase / Sprint	Phase 0 (MVP Architecture) + Phase 0.5 (Security Hardening)
Period Covered	2026-01-01 – 2026-02-23
Facilitator	John (AI Director)
Participants	John (AI Director), Builder Agent(s), Validator Agent, Alem Bašić (CEO)
Date of Session	2026-02-23
Duration	Async — Phase retrospective

---

2. What Went Well (Keep Doing)

#	What Went Well	Why It Worked	How to Preserve	Category
KD-01	Pass-through model (ADR-003) enforced by automated tests in `db.test.ts` from Day 1	Tests as compliance guardrails mean no human discipline required to maintain the invariant	Keep `db.test.ts` assertions (no balance, no CVV) as non-negotiable CI gate; never disable	Technical
KD-02	AI-native async workflow via Mission Control eliminated coordination overhead	Builder agents pick up tasks independently; no standup time wasted; John coordinates via task system	Continue using Mission Control for all task assignments; document decisions in `comms/decisions/`	Process
KD-03	Security audit conducted before Phase 1 — found 8 critical/high issues before production	External-perspective audit at right phase (post-MVP, pre-launch) caught real vulnerabilities	Run security audit at start of each major phase (0.5 → 1 → 2 → 3), not at the end	Technical
KD-04	Mock BaaS (`NEXT_PUBLIC_SERVICE_MODE=mock`) let full app architecture be built and tested without a real BaaS partner	Decoupled business logic from partner dependency — could ship working app and test all flows	Maintain the service mode abstraction layer; expand to `SUMSUB_MODE=mock` pattern for other integrations	Technical
KD-05	SQLite for local dev and test removed Docker dependency entirely — team setup time < 5 min	Zero infrastructure overhead for development; any agent can be productive immediately	Keep SQLite for local/test; PostgreSQL migration plan documented in Phase 1 roadmap	Process

#	What Went Well	Why It Worked	How to Preserve	Category
KD-06	Comprehensive test suite (40+ tests across 14 files) written alongside MVP code	Tests document expected behavior and make refactoring in Phase 0.5 safe and fast	Maintain test-first discipline; DoD requires tests for all new code	Technical
KD-07	Figma Make export as UI source of truth (`mockups/figma-make-export/`) prevented design drift	Single source for UI decisions; agents always check Make components before modifying UI	Always reference `mockups/figma-make-export/` before any UI change; document in CLAUDE.md	Process

Highlights

KD-01 Detail: The decision to add compliance assertions in `db.test.ts` on Day 1 (Phase 0) paid dividends throughout Phase 0.5. Every security hardening change that touched the schema was automatically validated against the pass-through model invariant. No developer (human or AI) had to remember "don't add a balance column" — the test suite enforced it. This pattern should be replicated for every regulatory constraint: make it a test, not a rule someone has to remember.

KD-03 Detail: The security audit at Phase 0.5 (pre-production, post-MVP) found a security score of 57/100 — 8 critical/high issues including no CSRF, in-memory rate limiting that resets on restart, bcrypt DoS via long passwords, and missing security headers. If these had been found post-launch, they would have been P0 production incidents. The timing was correct: audit after architecture is stable, before real users are onboarded. This should be standard practice for every Drop phase.

---

3. What Didn't Go Well (Stop Doing)

#	What Didn't Work	Root Cause	Impact (1-10)	Action Item	Owner	Deadline
SD-01	In-memory rate limiting reset on server restart — not durable across deploys	Chose simplest implementation (Node process memory) without considering Fly.io restart behavior	7	Replaced with DB-backed `rate_limit_requests` table in Phase 0.5. Document in tech-debt-log.md immediately when choosing non-durable implementations.	Builder Agent / John	Completed (Phase 0.5)

#	What Didn't Work	Root Cause	Impact (1-10)	Action Item	Owner	Deadline
SD-02	No CSRF protection on POST endpoints — discovered in security audit	Security checklist not applied during API endpoint implementation	8	Added CSRF middleware in Phase 0.5. Add CSRF to DoD security checklist as mandatory gate for all POST/PATCH/DELETE.	John	Completed (Phase 0.5)
SD-03	bcrypt DoS vector via unbounded password length — no input max before hashing	Missed OWASP Input Validation guidance for bcrypt specifically	7	Added Zod `max(1000)` before bcrypt in Phase 0.5. Added regression test. Document bcrypt DoS pattern in coding-standards.md.	Builder Agent / John	Completed (Phase 0.5)
SD-04	Security score 57/100 at first audit — below target of 80/100	Security was a "Phase 0.5 concern" — deferred until post-MVP, leading to 8 issues needing remediation	7	In future phases: run OWASP Top 10 checklist during feature development, not as a batch post-MVP audit. Add security review gate to each PR in DoD.	John	Ongoing (Phase 1 onwards)
SD-05	Missing `npm audit` gate in initial CI — dependencies with CVEs could have been shipped	CI/CD pipeline focused on tests only; dependency security check not included	6	Add `npm audit --audit-level=high` to GitHub Actions CI pipeline. Fail build on HIGH/CRITICAL CVEs.	Builder Agent	Phase 1

#	What Didn't Work	Root Cause	Impact (1-10)	Action Item	Owner	Deadline
SD-06	No structured logging from Day 1 — `console.log` used for debugging in some paths	Fast MVP pace — structured logging deferred as "Phase 1 concern"	5	Add Pino structured logging in Phase 1. Document in tech-debt-log.md as TD-008.	John	Phase 1

Root Cause Analysis — Deep Dives

SD-04: Security deferred to Phase 0.5 caused remediation batch work

Problem: 8 critical/high security issues found in audit — all required in Phase 0.5 sprint.

5 Whys:

1. Why were 8 security issues found in the audit? → Because security checklist was not applied during Phase 0 feature development.
2. Why was the security checklist not applied? → Because Phase 0 goal was "working architecture" — security was explicitly a Phase 0.5 task.
3. Why was security explicitly deferred? → Because the team assumed a working architecture could be security-hardened efficiently afterward.
4. Why is that assumption problematic? → Because retrofitting security (CSRF, rate limiting, headers) into existing code is more work than building it in — and some security patterns require architectural changes.
5. Why wasn't this anticipated? → No "security baseline" was defined for Phase 0 — only for Phase 0.5.

Root Cause: Deferred security without a minimum viable security baseline for Phase 0 meant the Phase 0.5 sprint carried all security work. **Systemic Fix:** Define a Minimum Security Baseline (MSB) for every phase. Phase 0 MSB should include: parameterized SQL, bcrypt passwords, httpOnly JWT, basic rate limiting. Advanced security (CSRF, security headers, audit logging) is Phase 0.5. Document MSB in DoD. **Ljestvica Fikseva (ZAKON #1):** Rule update → DoD security section updated with MSB definition.

---

4. What to Try Next (Start Doing)

#	What to Try	Expected Benefit	Hypothesis	Success Metric	Trial Period

ST-01	OWASP Top 10 checklist review on every PR (not just security audit phases)	Catch security issues per-feature instead of in batch	If we review OWASP Top 10 per PR, then Phase 1 security audit score will be > 80/100 without a dedicated security hardening sprint	Security audit score ≥ 80/100 at Phase 1 review without a dedicated 0.5-equivalent sprint	Phase 1 (3 months)
ST-02	`npm audit` as CI gate — fail build on HIGH/CRITICAL CVEs	Prevent known vulnerable dependencies from reaching staging	If we gate on `npm audit`, then no HIGH/CRITICAL CVEs reach production	Zero HIGH/CRITICAL CVEs at Phase 1 launch	Phase 1
ST-03	Performance regression gate in CI — `api-benchmarks.test.ts` as a CI check	Prevent silent performance regressions from shipping	If we run benchmarks in CI, then no P95 regression > 15% reaches staging	0 performance regressions > 15% across Phase 1	Phase 1
ST-04	Structured logging (Pino) from Phase 1 Day 1 — not retrofitted	Better observability from launch; no retrofitting cost	If we add Pino from the start, Phase 1 debugging will be 50% faster than Phase 0	Avg time to diagnose a production issue < 30 min	Phase 1

---

5. Key Findings by Category

5.1 Technical Findings

#	Finding	Impact	Recommendation
TF-01	SQLite WAL mode handles moderate concurrency but has a hard limit (~200 concurrent writes). PostgreSQL migration is required before launch.	High	Migrate to PostgreSQL in Phase 1 before real user onboarding. Plan migration script in `db/migrations/`.
TF-02	Mock service abstraction (`NEXT_PUBLIC_SERVICE_MODE`) enables full E2E testing without BaaS partner. Pattern is sound.	High (positive)	Extend pattern to `SUMSUB_MODE`, `BANKID_MODE` for Phase 2. Document in local-development-setup.md.

#	Finding	Impact	Recommendation
TF-03	bcrypt cost factor 12 adds ~800ms to login/register. This is intentional security but users will notice on slow devices.	Medium	Profile on target device hardware. Add loading state in UI during auth operations.
TF-04	Next.js App Router with server-side API routes gives clean separation but RSC (React Server Components) hydration can cause subtle state issues.	Medium	Add E2E test coverage for client-side navigation. Monitor for hydration errors in Phase 1.
TF-05	26 API routes with consistent error format (typed responses, 402/403 for financial errors) makes client-side error handling reliable.	High (positive)	Maintain consistent error format as new routes are added in Phase 1.

Technical Patterns Observed:

- Test-as-compliance-guardrail pattern (db.test.ts) — should be replicated for every regulatory constraint going forward
- Service mode abstraction (mock/live) — excellent pattern for decoupling from external partners; extend to all integrations

5.2 Process Findings

#	Finding	Root Cause	Recommendation
PF-01	AI-native async workflow works well for implementation tasks but requires explicit handoff documentation when agents change	No persistent agent memory between sessions	Maintain CLAUDE.md, onboarding guide, and tech-debt-log.md as "agent continuity" documents. Update before every session end.
PF-02	Security as a Phase 0.5 sprint (rather than baked into Phase 0) created batch remediation work	Phase planning deferred security to avoid blocking MVP	Define Minimum Security Baseline per phase. Phase 0 = auth + parameterized SQL. Phase 0.5 = CSRF + rate limiting + headers.
PF-03	Mission Control + CLAUDE.md combination works as tribal knowledge preservation for AI agents	Explicit documentation of decisions and patterns	Maintain both. CLAUDE.md = quick reference. Mission Control = task history. `comms/decisions/` = decision log.

Process Patterns Observed:

- AI-native teams need more explicit documentation than human teams (no implicit knowledge retention between sessions) — this is a feature, not a bug, when documentation discipline is enforced

5.3 Communication Findings

#	Finding	Stakeholder Impact	Recommendation
CF-01	Alem (CEO) needs clear phase milestone signals — not technical details	CEO makes business decisions (BaaS partner, Finanstilsynet timing) based on phase readiness	Weekly email from John to alem@alai.no with phase status, blockers, and required CEO decisions.
CF-02	Security audit score (57/100) communicated effectively as "needs Phase 0.5 work before Phase 1" — Alem approved correctly	No confusion about scope or priority	Continue using numerical scores + plain language summaries for executive communication.

5.4 Tools & Infrastructure Findings

#	Finding	Impact	Recommendation
IF-01	Fly.io + SQLite (Phase 0) setup works for MVP/demo; will need PostgreSQL + Fly.io Postgres addon for Phase 1	High	Add PostgreSQL migration to Phase 1 task list in Mission Control.
IF-02	GitHub Actions CI with Vitest + Playwright covers all test types. No gaps for current scope.	High (positive)	Add `npm audit` gate to existing CI pipeline in Phase 1.
IF-03	Vaultwarden (`vault.basicconsulting.no`) as secrets manager works. bw CLI access for agents via session token.	Medium	Ensure all new Phase 1 secrets (BaaS API keys, Sumsub production keys) go into Vaultwarden immediately.

5.5 Team & Collaboration Findings

#	Finding	Impact	Recommendation
TM-01	Builder Agent + Validator Agent (read-only) separation works well for fintech — Validator catches issues Builder misses	High (positive)	Maintain read-only constraint on Validator. Never give Validator write access — reduces risk of well-intentioned but unreviewed changes.

#	Finding	Impact	Recommendation
TM-02	John (AI Director) as architecture owner with clear ADR documentation enables consistent decisions across agent sessions	High (positive)	Continue ADR pattern. All significant decisions documented in `project/architecture/`.

---

6. Action Items

#	Action Item	Category	Owner	Sprint / Deadline	Priority	Status
AI-01	Add `npm audit --audit-level=high` to GitHub Actions CI	Technical	Builder Agent	Phase 1, Sprint 1	P1	Open
AI-02	Add OWASP Top 10 checklist to PR template and DoD	Process	John	Phase 1, Sprint 1	P1	Open
AI-03	Define Minimum Security Baseline (MSB) for each phase and document in DoD	Process	John	Phase 1, Sprint 1	P1	Open
AI-04	Plan PostgreSQL migration — create `db/migrations/postgres-migration.md`	Technical	John + Builder Agent	Phase 1, Sprint 2	P1	Open
AI-05	Add Pino structured logging to Phase 1 from Day 1	Technical	Builder Agent	Phase 1, Sprint 1	P2	Open

#	Action Item	Category	Owner	Sprint / Deadline	Priority	Status
AI-06	Add performance regression gate (`api-benchmarks.test.ts`) to CI	Technical	Builder Agent	Phase 1, Sprint 2	P2	Open
AI-07	Weekly CEO status email template — phase status, blockers, required decisions	Communication	John	Ongoing from Phase 1	P2	Open
AI-08	Extend service mode abstraction: `SUMSUB_MODE=mock/live` + `BANKID_MODE=mock/live`	Technical	Builder Agent	Phase 2, Sprint 1	P3	Open

Action Item Review: At the START of each phase planning, John reviews status of all open action items. Incomplete P1 items are escalated to Alem Bašić (CEO).

---

7. Metrics Comparison — Planned vs. Actual

7.1 Timeline

Milestone	Planned	Actual	Variance	Root Cause (if significant)
Phase 0 MVP Architecture	2026-01-15	2026-02-01	+17 days	Architecture decisions (pass-through model, ADR-003) took longer to finalize
Phase 0.5 Security Hardening	2026-02-28	2026-02-23	-5 days	Security fixes were well-scoped; AI agent velocity higher than estimated
Phase 1 BaaS Integration	2026-04-30	TBD	—	Blocked on BaaS partner confirmation

7.2 Budget

Category	Planned (NOK)	Actual (NOK)	Variance	Notes
Development (AI agents)	150,000	~50,000 (est.)	Under	AI agents significantly cheaper than human developer equivalent
Infrastructure (Fly.io)	10,000/year	~2,000 (est.)	Under	MVP usage is low; scales with real users
Design	20,000	5,000 (est.)	Under	Figma Make for UI source of truth reduced design cost
Total	**~250,000**	**~57,000 (est.)**	**Under**	Budget runway preserved for Phase 2 BaaS/compliance costs

7.3 Quality

Metric	Target	Actual	Status
Test coverage	≥ 80%	~85% (est.)	Pass
Pass-through invariant tests	Always pass	Always pass	Pass
Critical bugs at Phase 0.5	0	0	Pass
Security audit score	≥ 80/100 post-0.5	57/100 → targeting 80/100	In Progress
All 40+ Vitest tests passing	Yes	Yes	Pass
Playwright E2E (3 projects)	Pass	Pass	Pass

7.4 Velocity & Estimation

Sprint	Estimated Points	Completed Points	Accuracy
Phase 0 MVP	80	85	106%
Phase 0.5 Security	40	42	105%
Average			**~105%**

Estimation Accuracy Notes: AI agents tend to complete slightly more than estimated when tasks are well-defined with clear acceptance criteria. Under-estimation typically occurs on architectural decision tasks (ADRs, design exploration) rather than implementation tasks.

---

8. Recommendations for Future Projects

#	Recommendation	Category	Confidence	Applicable When
REC-01	Define a Minimum Security Baseline (MSB) for every phase before development starts. Phase 0 MSB = parameterized SQL + bcrypt + httpOnly JWT. Phase 0.5 = CSRF + rate limiting + headers.	Technical	High	All fintech/payment projects from Day 1
REC-02	Build compliance invariants as automated tests before any feature code. (e.g., `db.test.ts` for no balance/CVV assertions.)	Technical	High	Any project with regulatory constraints
REC-03	Use service mode abstraction (mock/live) for all external integrations from the start. Enables architecture validation without partner dependency.	Technical	High	Any project with external payment/identity providers
REC-04	For AI-native teams: CLAUDE.md + Mission Control + `comms/decisions/` is the minimum viable knowledge preservation system. Update it every session.	Process	High	All AI-native team projects
REC-05	Run `npm audit` as a CI gate from Day 1. Adding it later means clearing a backlog of CVEs. Starting clean is much easier.	Technical	High	All Node.js projects

Top 3 Most Important Learnings

1. **Compliance as tests, not rules:** The most reliable way to maintain the pass-through model invariant was making it a CI test (`db.test.ts`) rather than a team convention. Tests cannot be forgotten; conventions can. For any regulatory constraint, immediately ask: "How do I make this a test?"
2. **Security baseline vs. security sprint:** Deferring all security to a dedicated Phase 0.5 sprint was better than nothing but created batch remediation work. The right model is a per-phase Minimum Security Baseline (MSB) — implement the MSB during feature development, not afterwards. Advanced hardening (CSRF, security headers, audit logging) can still be a dedicated sprint, but the baseline prevents the worst vulnerabilities from ever shipping.
3. **Mock integrations enable architecture confidence without partner dependency:** The `NEXT_PUBLIC_SERVICE_MODE=mock` pattern let Drop build, test, and validate 26 API routes and 3 Playwright E2E projects without a confirmed BaaS partner. This is the correct approach for any product that depends on external financial partners. Build to the interface; connect the real partner when the contract is signed.

---

9. Knowledge Base Contribution

Finding	Contribution Type	Target System	Status	Owner
Compliance-as-tests pattern	New rule	DoD + Coding Standards	Done	John
Per-phase Minimum Security Baseline	New rule	DoD security checklist	Open (AI-03)	John
Mock service mode abstraction pattern	New convention	CLAUDE.md + Coding Standards	Done	John
bcrypt DoS via unbounded input	New rule	Coding Standards + auth patterns	Done	John
DB-backed rate limiting required for persistence	Tech debt entry + lesson	tech-debt-log.md TD-004	Done	John

HiveMind entries to add (run after reading this document):

- `node ~/system/agents/hivemind/hivemind.js post "Drop: compliance-as-tests pattern — make regulatory invariants into CI tests, not team conventions"`
- `node ~/system/agents/hivemind/hivemind.js post "Drop: mock service mode abstraction (NEXT_PUBLIC_SERVICE_MODE) enables full E2E testing without BaaS partner — extend to SUMSUB_MODE and BANKID_MODE in Phase 2"`

CLAUDE.md updated:

- `~/ALAI/products/Drop/CLAUDE.md` — Pass-through model rules, mock mode documentation

Related Documents

- Tech Debt Log
- Security Audit Report
- Definition of Done
- Architecture Decision Records
- Risk Register

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	

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
Security	Known vulnerabilities not yet patched, security shortcuts taken	Urgency, lack of security knowledge	Security scans, CVE alerts

Category	Description	Common Causes	Detection Method
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)			

Change Request

Change Request: {{CR_TITLE}}

> **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. Change Request Metadata

Field	Value
CR ID	CR-{{XXX}} *(assigned by PM)*
Date Submitted	{{DATE}}
Submitted By	{{NAME}} — {{ROLE}}
Project	{{PROJECT_NAME}}
Priority	Critical / High / Medium / Low
CR Type	Scope Change / Budget Change / Timeline Change / Requirements Change / Design Change / Technical Change
Current Phase	Planning / Design / Development / Testing / Deployment
Decision Deadline	{{DATE}} *(by when a decision must be made to avoid impact)*

---

2. Change Description

2.1 Summary of Change

{{ONE_PARAGRAPH_SUMMARY_OF_WHAT_CHANGES}}

2.2 Current State (Before)

{{DESCRIPTION_OF_CURRENT_STATE}}

Currently approved in: `{{DOCUMENT_NAME_AND_VERSION}}`, Section {{X.X}}

2.3 Proposed State (After)

{{DESCRIPTION_OF_PROPOSED_STATE}}

2.4 Out of Scope for This Change

This CR does NOT include:

- {{EXPLICITLY_EXCLUDED_ITEM_1}}
- {{EXPLICITLY_EXCLUDED_ITEM_2}}

---

3. Reason & Justification

3.1 Reason for Change

Reason Category	Applies?	Details
New business requirement discovered	Yes / No	{{DETAILS}}
Regulatory / compliance mandate	Yes / No	{{DETAILS}}
Client feedback from UAT / prototype	Yes / No	{{DETAILS}}
Technical blocker / infeasibility	Yes / No	{{DETAILS}}
Market opportunity / competitive pressure	Yes / No	{{DETAILS}}
Error/omission in original requirements	Yes / No	{{DETAILS}}
Performance / quality improvement	Yes / No	{{DETAILS}}

Primary Justification: {{CLEAR_BUSINESS_JUSTIFICATION_WHY_THIS_CHANGE_IS_NECESSARY}}

3.2 Consequence of Not Changing

If this change is **not** approved: {{CONSEQUENCE_OF_REJECTION}}

---

4. Impact Analysis

4.1 Scope Impact

Deliverable Affected	Current Scope	Proposed Scope	Impact Type
{{DELIVERABLE}}	{{CURRENT}}	{{PROPOSED}}	Added / Removed / Modified
{{DELIVERABLE_2}}			

Scope Change Size: Small (< 1 day) / Medium (1–3 days) / Large (3–10 days) / Major (> 10 days)

4.2 Timeline Impact

Milestone	Current Date	New Date (if approved)	Delay
{{MILESTONE}}	{{DATE}}	{{NEW_DATE}}	{{DAYS}} days
Project completion	{{DATE}}	{{NEW_DATE}}	{{DAYS}} days

Timeline Impact: None / Minor (≤ 3 days) / Moderate (4–14 days) / Major (> 14 days)

Critical Path Impact: Yes / No If yes: {{WHICH_CRITICAL_PATH_ITEMS_AFFECTED}}

4.3 Budget Impact

Cost Category	Current Budget (NOK)	Additional Cost (NOK)	Notes
Development	{{CURRENT}}	{{ADDITIONAL}}	{{NOTES}}
Design			
Testing			
Infrastructure			
Total Additional Cost		**{{TOTAL_ADDITIONAL}}**	

Budget Impact: None / Minor (< 5%) / Moderate (5–15%) / Major (> 15%)

Funding Source for Additional Cost: {{HOW_WILL_ADDITIONAL_COST_BE_COVERED}} *(e.g., contingency reserve, client additional purchase order, scope reduction elsewhere)*

4.4 Resource Impact

Resource	Current Allocation	Required if Approved	Impact
{{ROLE}}	{{CURRENT_ALLOCATION}}	{{NEW_ALLOCATION}}	{{NOTES}}

4.5 Risk Impact

Risk	Probability	Impact	Notes
{{NEW_RISK_INTRODUCED}}	H/M/L	H/M/L	
{{EXISTING_RISK_CHANGES}}	H/M/L	H/M/L	Risk score changes from {{OLD}} to {{NEW}}

4.6 Quality Impact

- Test cases affected: {{LIST_OF_TEST_CASES_NEEDING_UPDATE}}
- Regression risk: {{HIGH/MEDIUM/LOW}} — {{EXPLANATION}}
- NFRs affected: {{LIST_ANY_NFRs_IMPACTED}}

4.7 Affected Deliverables / Documents

Document	Section	Type of Change	Owner
{{DOCUMENT}}	{{SECTION}}	Update / Add / Remove	{{OWNER}}
`functional-requirements.md`	FR-{{XXX}}	Modify	BA
`user-stories.md`	US-{{XXX}}	Add new story	BA
Test cases	TC-{{XXX}}	Update	QA

---

5. Alternative Approaches Considered

Alternative	Description	Why Rejected
Option A (Proposed)	{{THIS_CR}}	*Recommended*
Option B	{{ALTERNATIVE}}	{{WHY_NOT_CHOSEN}}

Alternative	Description	Why Rejected
Option C — Do Nothing	Reject the change	{{CONSEQUENCE_OF_REJECTION}}

Recommendation: Option {{A/B/C}} **Rationale:** {{WHY_THIS_IS_THE_BEST_OPTION}}

---

6. Implementation Plan

6.1 Implementation Steps

#	Task	Owner	Effort	Target Date
1	{{TASK}}	{{OWNER}}	{{EFFORT}}	{{DATE}}
2				
3	Update test cases for affected features	QA	{{EFFORT}}	{{DATE}}
4	Update requirements documents	BA	{{EFFORT}}	{{DATE}}
5	Regression testing	QA	{{EFFORT}}	{{DATE}}

6.2 Dependencies

Dependency	Type	Blocking?
{{DEPENDENCY}}	Internal / External	Yes / No

6.3 Test Plan for This Change

- Unit tests: {{WHICH_UNITS_NEED_NEW/UPDATED_TESTS}}
- Integration tests: {{WHAT_INTEGRATIONS_TO_RETEST}}
- Regression scope: {{WHICH_EXISTING_FEATURES_TO_REGRESSION_TEST}}
- UAT: {{DOES_THIS_REQUIRE_CLIENT_UAT? Y/N — WHICH_SCENARIOS}}

---

7. Rollback Plan

Rollback Trigger: {{WHAT_CONDITION_TRIGGERS_ROLLBACK}}

Rollback Steps:

1. {{STEP_1}}
2. {{STEP_2}}
3. {{STEP_3}}

Rollback Owner: {{WHO_EXECUTES_ROLLBACK}} **Rollback Time Required:** {{ESTIMATED_TIME}} **Data Recovery Needed:** Yes / No — {{IF_YES_HOW}}

---

8. Approval Workflow

8.1 Approval Matrix

Budget / Timeline Impact	Required Approvals	Target Decision Time
None	PO + PM	1 business day
< 5% budget OR < 3 days	PO + PM + John	2 business days
5–15% budget OR 3–14 days	PO + PM + John + Client Sponsor	3 business days
> 15% budget OR > 14 days	PO + PM + John + Client Sponsor + Alem	5 business days

8.2 This Change Requires

- ☐ **PM Review** — Impact analysis complete and accurate
- ☐ **Tech Lead Review** — Technical feasibility and effort confirmed
- ☐ **Product Owner** — Requirements and priority alignment
- ☐ **John (AI Director)** — Delivery accountability
- ☐ **Client Sponsor** — Business justification and budget approval *(if client-side change)*
- ☐ **Alem (CEO)** — Budget changes > 15% *(only if required)*

8.3 Decision Record

Level	Reviewer	Decision	Date	Comments
PM	{{NAME}}	Approved / Rejected / Deferred	{{DATE}}	{{COMMENTS}}
Tech Lead	{{NAME}}	Approved / Rejected	{{DATE}}	
Product Owner	{{NAME}}	Approved / Rejected	{{DATE}}	
AI Director (John)	John	Approved / Rejected	{{DATE}}	

Level	Reviewer	Decision	Date	Comments
Client Sponsor	{{NAME}}	Approved / Rejected	{{DATE}}	
CEO (Alem)	Alem	Approved / Rejected	{{DATE}}	*(if required)*

Final Decision: APPROVED / REJECTED / DEFERRED **Decision Date:** {{DATE}} **Effective From Sprint:** Sprint {{X}}

---

9. Change Log

Date	Changed By	What Changed
{{DATE}}	{{NAME}}	Updated impact analysis after Tech Lead review

---

Approval

Role	Name	Date	Signature
Author			
Reviewer			
Project Manager			
AI Director (John)			
Approver			

Lessons Learned

Lessons Learned: {{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. Session Metadata

Field	Value
Session Type	Sprint Retrospective / Phase Review / Post-Mortem / Project Close
Project	{{PROJECT_NAME}}
Phase / Sprint	{{PHASE_OR_SPRINT}}
Period Covered	{{START_DATE}} – {{END_DATE}}
Facilitator	{{NAME}}
Participants	{{LIST_OF_PARTICIPANTS}}
Date of Session	{{DATE}}
Duration	{{DURATION}}

---

2. What Went Well (Keep Doing)

#	What Went Well	Why It Worked	How to Preserve	Category
KD-01	{{DESCRIPTION}}	{{ROOT_CAUSE_OF_SUCCESS}}	{{PROCESS/RULE/HABIT}}	Process / Technical / Communication / Tools
KD-02				
KD-03				
KD-04				
KD-05				

Highlights

KD-01 Detail: {{EXTENDED_NARRATIVE_OF_WHAT_WORKED_AND_WHY}}

---

3. What Didn't Go Well (Stop Doing)

#	What Didn't Work	Root Cause	Impact (1-10)	Action Item	Owner	Deadline
SD-01	{{DESCRIPTION}}	{{ROOT_CAUSE}}	{{IMPACT}}	{{SPECIFIC_ACTION}}	{{OWNER}}	{{DATE}}
SD-02						
SD-03						
SD-04						

Root Cause Analysis — Deep Dives

SD-{{XX}}: {{ISSUE_TITLE}}

Problem: {{CLEAR_PROBLEM_STATEMENT}}

5 Whys:

1. Why did this happen? → {{ANSWER_1}}
2. Why did {{ANSWER_1}} happen? → {{ANSWER_2}}
3. Why did {{ANSWER_2}} happen? → {{ANSWER_3}}
4. Why did {{ANSWER_3}} happen? → {{ANSWER_4}}

5. Why did {{ANSWER_4}} happen? → {{ROOT_CAUSE}}

Root Cause: {{DEFINITIVE_ROOT_CAUSE}} **Systemic Fix:** {{HOOK/TOOL/RULE/PROCESS_CHANGE}} **Ljestvica Fikseva (ZAKON #1):** Hook > Tool > Rule > CLAUDE.md > Memory

---

4. What to Try Next (Start Doing)

#	What to Try	Expected Benefit	Hypothesis	Success Metric	Trial Period
ST-01	{{NEW_PRACTICE}}	{{EXPECTED_BENEFIT}}	If we do {{X}}, then {{Y}} will improve by {{Z}}%	{{MEASURABLE_METRIC}}	{{SPRINT_COUNT}} sprints
ST-02					
ST-03					

---

5. Key Findings by Category

5.1 Technical Findings

#	Finding	Impact	Recommendation
TF-01	{{TECHNICAL_FINDING}}	{{HIGH/MEDIUM/LOW}}	{{RECOMMENDATION}}
TF-02			

Technical Patterns Observed:

- {{PATTERN_1}} — {{IMPLICATION_AND_RECOMMENDATION}}
- {{PATTERN_2}}

5.2 Process Findings

#	Finding	Root Cause	Recommendation
PF-01	{{PROCESS_FINDING}}	{{ROOT_CAUSE}}	{{RECOMMENDATION}}
PF-02			

Process Patterns Observed:

- {{PATTERN}}

5.3 Communication Findings

#	Finding	Stakeholder Impact	Recommendation
CF-01	{{COMMUNICATION_FINDING}}	{{IMPACT}}	{{RECOMMENDATION}}
CF-02			

5.4 Tools & Infrastructure Findings

#	Finding	Impact	Recommendation
IF-01	{{TOOLING_FINDING}}	{{IMPACT}}	{{RECOMMENDATION}}
IF-02			

5.5 Team & Collaboration Findings

#	Finding	Impact	Recommendation
TM-01	{{TEAM_FINDING}}	{{IMPACT}}	{{RECOMMENDATION}}

---

6. Action Items

#	Action Item	Category	Owner	Sprint / Deadline	Priority	Status
AI-01	{{SPECIFIC_ACTIONABLE_ITEM}}	Process / Technical / Communication / Tools	{{OWNER}}	Sprint {{X}}	P1/P2/P3	Open
AI-02						
AI-03						
AI-04						
AI-05						

Action Item Review: At the START of next sprint planning, PM reviews status of all open action items. Incomplete P1 items are escalated to John.

---

7. Metrics Comparison — Planned vs. Actual

7.1 Timeline

Milestone	Planned	Actual	Variance	Root Cause (if significant)
Requirements Complete	{{DATE}}	{{DATE}}	{{+/- DAYS}}	{{ROOT_CAUSE}}
Design Complete				
MVP Release				
UAT Complete				
Launch				

Timeline Accuracy: {{ACTUAL_DURATION}} vs. planned {{PLANNED_DURATION}} ({{VARIANCE_PCT}}% {{over/under}})

7.2 Budget

Category	Planned (NOK)	Actual (NOK)	Variance	Notes
Development	{{AMOUNT}}	{{AMOUNT}}	{{+/-}}	
Design				
Infrastructure				
Testing				
Total	**{{TOTAL}}**	**{{TOTAL}}**	**{{+/-}}**	

Budget Accuracy: {{ACTUAL}} vs. planned {{PLANNED}} ({{VARIANCE_PCT}}% {{over/under}})

7.3 Quality

Metric	Target	Actual	Status
Test coverage	≥ 80%	{{PCT}}%	{{ /⚠/ }}
Critical bugs at launch	0	{{COUNT}}	{{ /⚠/ }}
Post-launch bugs (30 days)	< 5	{{COUNT}}	{{ /⚠/ }}

Metric	Target	Actual	Status
Lighthouse performance score	≥ 90	{{SCORE}}	{{ /⚠/ }}
Accessibility (WCAG AA violations)	0	{{COUNT}}	{{ /⚠/ }}

7.4 Velocity & Estimation

Sprint	Estimated Points	Completed Points	Accuracy
Sprint 1	{{ESTIMATE}}	{{ACTUAL}}	{{PCT}}%
Sprint 2			
Sprint 3			
Average			**{{AVG_PCT}}%**

Estimation Accuracy Notes: {{PATTERNS_IN_OVER_UNDER_ESTIMATION}}

---

8. Recommendations for Future Projects

#	Recommendation	Category	Confidence	Applicable When
REC-01	{{RECOMMENDATION}}	Technical / Process / Communication	High / Medium	{{WHEN_TO_APPLY}}
REC-02				
REC-03				
REC-04				
REC-05				

Top 3 Most Important Learnings

1. **{{LEARNING_1_TITLE}}:** {{EXPLANATION_AND_RECOMMENDATION}}
2. **{{LEARNING_2_TITLE}}:** {{EXPLANATION_AND_RECOMMENDATION}}
3. **{{LEARNING_3_TITLE}}:** {{EXPLANATION_AND_RECOMMENDATION}}

---

9. Knowledge Base Contribution

Finding	Contribution Type	Target System	Status	Owner
{{FINDING}}	New rule / Updated tool / New hook / HiveMind entry / CLAUDE.md update	{{SYSTEM}}	Open / Done	{{OWNER}}

HiveMind entries added:

- `node ~/system/agents/hivemind/hivemind.js post "{{KEY_LEARNING}}"` *(run this)*

Rules updated:

- `~/system/rules/{{RULE_FILE}}` — {{WHAT_WAS_ADDED}}

Memory updated:

- `~/.claude/projects/-Users-makinja/memory/{{MEMORY_FILE}}` — {{WHAT_WAS_ADDED}}

Approval

Role	Name	Date	Signature
Author			
Reviewer			
Project Manager			
AI Director (John)			
Team	(retrospective agreement)		