Definition of Done: Drop — Fintech Payment App

Definition of Done: 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 DoD — fintech-specific with pass-through model invariants

---

1. Purpose

The Definition of Done (DoD) is a shared agreement across the Drop 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 (Builder agent, Validator agent, John)
- Create a shared language for "done" that developers, QA, and Alem (CEO) all agree on
- Prevent issues from reaching production that could have been caught earlier
- Protect the pass-through model: Drop NEVER stores customer money, card numbers, or CVV data

Enforcement: The DoD is non-negotiable for Drop. Any exception requires explicit sign-off from John (AI Director) and must be tracked as a Mission Control task with a linked risk acceptance from Alem Bašić (CEO). Undocumented shortcuts are not acceptable in a fintech product.

---

2. Feature-Level DoD

When is a **feature** done?

Code Quality

- ☐ Code is complete and implements all acceptance criteria from the AC document (`docs/BUSINESS-REQUIREMENTS/acceptance-criteria.md`)
- ☐ Code reviewed and approved by Validator agent (read-only reviewer)
- ☐ All review comments resolved or explicitly deferred with a Mission Control task
- ☐ No TODO/FIXME comments left without corresponding Mission Control tasks
- ☐ Code follows Drop coding standards — TypeScript, Next.js App Router patterns, parameterized SQL
- ☐ No unnecessary dead code added
- ☐ No `console.log` / debug statements left in production code

Testing — Drop Specific

- ☐ Unit tests written for all new business logic (≥ 80% coverage; 100% for auth + transaction logic)
- ☐ Integration tests written for all new API endpoints (`api-endpoints.test.ts`)
- ☐ DB compliance assertions added if any schema changes (`db.test.ts` — no balance, no CVV)
- ☐ All existing tests still pass (no regressions — all 40+ Vitest tests green)
- ☐ Test coverage has not decreased below project minimum (80% overall)
- ☐ Edge cases tested: age boundary (exactly 18), amount boundaries (100 NOK min, 50,000 NOK max)
- ☐ **Pass-through invariant test**: if new DB schema changes, `db.test.ts` must assert:
  - `users` table has NO `balance` column
  - `cards` table has NO `card_number` or `cvv` columns

Security — Drop Fintech Standards

- [ ] OWASP Top 10 reviewed for this feature (especially if touching auth or payments)
- [ ] Input validation in place for all user inputs (Zod schema — server-side)
- [ ] Authorization checks correct (JWT cookie required for all protected endpoints)
- [ ] No sensitive data logged (no passwords, no card numbers, no full JWT tokens in logs)
- [ ] No sensitive data in API responses (no password hashes, no full card numbers)
- [ ] Rate limiting applies to new endpoints if they handle auth or financial actions
- [ ] CSRF token required on all new POST/PATCH/DELETE endpoints
- [ ] Parameterized SQL only — no string concatenation in DB queries
- [ ] `npm audit` passes — no new HIGH/CRITICAL CVEs introduced

Performance

- [ ] Performance budget met — no P95 regression > 15% vs baseline (api-benchmarks.test.ts)
- [ ] No N+1 query problems (each API call makes a bounded number of DB queries)
- [ ] bcrypt timing within 1,000ms P95 (after registration/login changes)
- [ ] No synchronous heavy operations blocking the event loop

Documentation

- [ ] Code is self-documenting or has inline comments for non-obvious Drop business logic
- [ ] API reference updated if new endpoints added (`docs/backend/API-REFERENCE.md`)
- [ ] Architecture Decision Record created for significant architectural choices (e.g., pass-through model changes)
- [ ] CLAUDE.md or relevant project docs updated if dev workflow changes

API (if applicable)

- [ ] API documentation updated (`docs/backend/API-REFERENCE.md`)
- [ ] Breaking changes explicitly documented in release notes
- [ ] Response schema consistent with existing patterns (error format, HTTP status codes)

Error Handling

- [ ] All error conditions handled gracefully (no unhandled promise rejections)
- [ ] Error messages are user-friendly and in Norwegian where applicable

- [ ] No stack traces in production API responses
- [ ] External service failures handled (mock BaaS timeout → user-friendly error)
- [ ] Financial errors use correct HTTP status codes (402 Insufficient Balance; 403 KYC Required)

Compliance Checklist (Drop-specific — MANDATORY)

- [ ] **PCI-DSS**: No `card_number` or `cvv` stored or returned in full
- [ ] **Pass-through model (ADR-003)**: No `balance` column added to users table
- [ ] **AML**: Transaction amounts ≤ 50,000 NOK enforced
- [ ] **Age verification**: Age ≥ 18 validation in place for any new user-facing flows
- [ ] **Audit trail**: New financial events logged in `audit_logs` table

Deployment

- [ ] Deployed to staging (https://drop-staging.fly.dev/) and manually verified
- [ ] Database migrations tested on staging (up and down migrations)
- [ ] Environment variables documented in `docs/DEVELOPER-EXPERIENCE/local-development-setup.md`
- [ ] Feature flag configured if the feature is gated (e.g., Cards, BankID)
- [ ] John (AI Director) has reviewed and accepted the feature in staging

---

3. Sprint-Level DoD

When is a **sprint** done?

- [ ] All committed Mission Control tasks meet their individual Definition of Done
- [ ] All automated tests passing in CI (Vitest: 40+ tests; Playwright: 3 projects)
- [ ] Staging environment is stable (no broken features on https://drop-staging.fly.dev/)
- [ ] Technical debt created during the sprint is tracked in Mission Control backlog
- [ ] Sprint review conducted — John (AI Director) and Alem (CEO if available) have seen deliverables
- [ ] Retrospective findings logged in `docs/CROSS-CUTTING/lessons-learned.md`
- [ ] Next sprint's tasks refined and priority-ordered in Mission Control

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 (all 26 API routes via `api-endpoints.test.ts`)
- [ ] Playwright E2E tests passing for all 5 critical journeys (user-flows, full-flows, input-chaos)
- [ ] Performance benchmarks passing — all NFR-P01..P06 targets met (`api-benchmarks.test.ts`)
- [ ] Security scan complete — no unresolved HIGH/CRITICAL findings (`npm audit` clean)
- [ ] DB compliance tests passing — `db.test.ts` all assertions green (no balance, no CVV)
- [ ] UAT conducted with Alem Bašić (CEO) sign-off (or documented conditional approval)
- [ ] Release notes written: `docs/RELEASE/release-notes.md`
- [ ] Deployment checklist complete: `docs/RELEASE/deployment-checklist.md`
- [ ] Rollback plan prepared and tested: `docs/RELEASE/rollback-plan.md`
- [ ] John (AI Director) briefed on all changes and potential failure modes
- [ ] Security audit score ≥ 80/100 (Phase 0.5 requirement)

---

5. Bug Fix DoD

When is a **bug fix** done?

- [ ] Root cause understood and documented in the Mission Control task
- [ ] Fix addresses root cause (not just symptoms)
- [ ] Unit/integration test written that would have caught this bug (regression test added)
- [ ] Fix does not introduce new failures (all 40+ tests still passing)
- [ ] Fix verified in staging environment
- [ ] Related areas spot-checked for similar issues
- [ ] Mission Control task updated with root cause and resolution notes
- [ ] `docs/CROSS-CUTTING/lessons-learned.md` updated if the bug reveals a systemic pattern

---

6. Hotfix DoD

When is a **hotfix** done?

- [ ] Fix verified locally and on staging (cannot skip staging — even for hotfixes)
- [ ] Validator agent has reviewed the fix (at minimum async review)
- [ ] Smoke tests passing post-deployment (Playwright user-flows project)
- [ ] Mission Control incident task updated with fix details
- [ ] Full regression test run scheduled within 4 hours
- [ ] Post-mortem scheduled within 24 hours if P1/P2 severity (DORA compliance)
- [ ] Rollback plan confirmed (Fly.io rollback ready in case hotfix causes regression)

---

7. Drop-Specific Additions

Pass-Through Model Invariant (Non-Negotiable)

The following assertions must pass on **every commit** — they are not optional:

```
// db.test.ts assertions — NEVER disable these:
expect(columns).not.toContain('balance')       // users table
expect(columns).not.toContain('card_number')   // cards table
expect(columns).not.toContain('cvv')           // cards table
```

If these tests fail, the build fails. No exceptions. No PR merges until fixed.

Financial Calculation Verification

All fee changes (currently 0.5% remittance, 1% QR) must include:

1. Unit test in `transactions.test.ts` verifying exact fee amount
2. Boundary test at minimum amount (100 NOK) and maximum amount (50,000 NOK)
3. A comment in the code referencing the business rule (BRD section)

Norwegian Language Compliance

User-facing error messages for age validation must include Norwegian text:

- Age rejection: "Du må være minst 18 år"
- Other messages: Norwegian primary, English secondary
- Check: Playwright input-chaos.spec.ts includes Norwegian error message assertions

Related Documents

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

Approval

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

Revision #5
Created 2026-02-23 12:06:08 UTC by John
Updated 2026-05-31 20:03:28 UTC by John