drop-validation-hardening-plan

Plan: Drop Validation Hardening

Research Summary

What the documentation says (spec vs reality)

Sources reviewed:

1. `project/architecture/api-specification.md` — API contract, field examples, error codes
2. `project/architecture/architecture-document.md` — User requirements from vilkår.html (legally binding)
3. `project/docs/security-qa-audit.md` — Issue #14 (email), #19 (password), #9 (amounts)
4. `project/docs/drop-qa-rapport.md` — QA findings C-1 through L-10
5. `src/lib/middleware/validation.ts` — Existing validators (UNUSED by any route)
6. `src/app/api/auth/register/route.ts` — Current registration validation
7. `src/app/onboarding/page.tsx` — Current frontend validation

Gap Analysis

Field	Spec / Audit Requirement	Current Implementation	Gap
Email	Regex `/^[^\s@]+@[^\s@]+\.[^\s@]+$/` (audit #14)	`email.includes("@")`	YES — accepts `@`, `a@`, `@@`
Password	12+ chars, uppercase, lowercase, digit (audit #19)	`length >= 8`, no complexity	YES — "12345678" passes
First/Last Name	`validateName()` exists in validation.ts — 1-100 chars, no script tags, sanitized	`typeof === "string"` only	YES — "123", XSS, 100K chars pass
Phone	`+47` Norwegian format (architecture doc 1.4)	No validation at all	YES — any string passes
Age	18+ from vilkår.html (architecture doc 1.4)	Not implemented	YES — but deferred (needs BankID)

Field	Spec / Audit Requirement	Current Implementation	Gap
Amount (remittance)	100-50,000 NOK, 2 decimal places, finite (audit #9)	100-50,000 check, isFinite	PARTIAL — no decimal precision
Amount (QR)	1-100,000 NOK, 2 decimals	1-100,000 check, isFinite	PARTIAL — no decimal precision
Bank account	IBAN validation (validation.ts has `validateIBAN`)	No validation	YES — but separate scope
Currency	NOK, RSD, BAM, PLN, PKR, TRY, EUR	Missing RSD, TRY, PKR, NOK	YES — but unused validator

Existing assets (ready to wire up)

`src/lib/middleware/validation.ts` already has:

- `validateEmail()` — proper regex
- `validatePhone()` — international `+` format, 8-15 digits
- `validateAmount()` — positive, max 2 decimals
- `validateIBAN()` — format + mod-97 checksum
- `validatePIN()` — exactly 4 digits
- `validateName()` — 1-100 chars, no script tags
- `sanitizeText()` — strips HTML, control chars, max length
- `validateCurrency()` — needs RSD, TRY, PKR, NOK added

Key insight: The validators EXIST but are NEVER IMPORTED. The middleware/ directory is entirely dead code (QA rapport C-1). The fix is to wire existing validators into the actual routes.

Objective

Wire existing validation utilities into all API routes and frontend forms. Close the gap between documented requirements and actual implementation. Do NOT create new validators — use what exists in `validation.ts`, extend where needed.

Scope

In scope:

1. Registration API — email, password, name, phone validation
2. Registration frontend — matching client-side checks
3. Login frontend — email format check

4. Amount validation — add decimal precision check to remittance + QR payment routes
5. Update existing tests to match new validation rules
6. Currency validator — add missing currencies

Out of scope (separate tasks):

- Age verification (needs BankID integration)
- IBAN validation for recipients (separate feature)
- CSRF protection (separate security task)
- Audit logging (separate task)
- Password complexity upgrade to 12+ (BREAKING CHANGE — needs migration plan, keep 8 for now but add complexity)

Team Orchestration

Team Members

ID	Name	Role	Agent Type
B1	validation-builder	Implement validation wiring	builder
V1	validation-validator	Verify all validation works	validator

Step-by-Step Tasks

Phase 1: Backend Validation (API Routes)

Task 1: Wire validators into registration API

- Owner: B1
- BlockedBy: none
- Files: `src/app/api/auth/register/route.ts`, `src/lib/middleware/validation.ts`
- Changes:
  1. Import `validateEmail`, `validateName`, `sanitizeText`, `validatePhone` from `@/lib/middleware/validation`
  2. Replace `!email.includes("@")` with `!validateEmail(email)`
  3. Replace `!firstName || typeof firstName !== "string"` with `!validateName(firstName)`
  4. Replace `!lastName || typeof lastName !== "string"` with `!validateName(lastName)`
  5. Add phone validation: `if (phone && !validatePhone(phone))` error
  6. Sanitize names before storing: `sanitizeText(firstName, 100)`, `sanitizeText(lastName, 100)`
  7. Add password complexity: min 8 chars + at least 1 letter + at least 1 digit (soft upgrade, not full 12-char yet)
  8. Add `validateCurrency` update: add missing currencies (RSD, TRY, PKR, NOK)

- Acceptance:
  - [ ] `user@` rejected (no domain)
  - [ ] `@domain.com` rejected (no local part)
  - [ ] `user space@domain.com` rejected (spaces)
  - [ ] `valid@email.com` accepted
  - [ ] `123` as firstName rejected (contains only digits? No — validateName allows digits, just checks length + no script tags. This is acceptable.)
  - [ ] `<script>alert(1)</script>` as name rejected
  - [ ] Empty firstName rejected
  - [ ] 200+ char firstName truncated to 100
  - [ ] Phone without `+` prefix rejected (if provided)
  - [ ] Phone `+4712345678` accepted
  - [ ] Password `12345678` rejected (no letter)
  - [ ] Password `abcdefgh` rejected (no digit)
  - [ ] Password `abc12345` accepted (letter + digit + 8 chars)

Task 2: Wire validators into amount routes

- Owner: B1
- BlockedBy: none
- Files: `src/app/api/transactions/remittance/route.ts`, `src/app/api/transactions/qr-payment/route.ts`
- Changes:
  1. Import `validateAmount` from `@/lib/middleware/validation`
  2. Add decimal precision check before range check: `if (Math.round(amount * 100) !== amount * 100)` → 400 error
  3. Existing range checks stay (100-50K for remittance, 1-100K for QR)
- Acceptance:
  - [ ] `100.999` rejected (3 decimals)
  - [ ] `100.99` accepted (2 decimals)
  - [ ] `100` accepted (0 decimals)
  - [ ] Existing range tests still pass

Task 3: Validate Task 1 + Task 2

- Owner: V1
- BlockedBy: 1, 2
- Acceptance:
  - [ ] Read all modified files, verify imports are correct
  - [ ] Verify no breaking changes to existing passing tests

- [ ] Run `npm test` — all 120 Vitest tests pass
- [ ] Run `npx playwright test` — all 75 e2e tests pass (may need test updates)

Phase 2: Frontend Validation (Client-side)

Task 4: Add proper validation to registration form

- Owner: B1
- BlockedBy: 3 (backend must be validated first)
- Files: `src/app/onboarding/page.tsx`
- Changes:
  1. Replace `!email.includes("@")` with proper regex check matching backend
  2. Add inline error for email format: "Ugyldig e-postformat"
  3. Add password complexity check: show "Passord må inneholde bokstaver og tall"
  4. Add name format hint if script tags detected: "Ugyldig tegn i navn"
  5. Add phone format hint: "Norsk telefonnummer påkrevd (+47...)"
  6. Keep button disabled logic, but add per-field inline errors
- Acceptance:
  - [ ] `user@` shows email format error immediately on blur
  - [ ] `12345678` password shows complexity error
  - [ ] Valid form submits normally
  - [ ] Error messages in Norwegian

Task 5: Add email validation to login form

- Owner: B1
- BlockedBy: 3
- Files: `src/app/login/page.tsx`
- Changes:
  1. Add email format check matching backend regex
  2. Show "Ugyldig e-postformat" if email doesn't match on submit
- Acceptance:
  - [ ] Invalid email format shows Norwegian error
  - [ ] Valid email + wrong password shows credential error (not format error)

Task 6: Validate Task 4 + Task 5

- Owner: V1
- BlockedBy: 4, 5
- Acceptance:
  - [ ] Read all modified frontend files
  - [ ] Verify error messages are in Norwegian
  - [ ] Run `npx playwright test` — all 75 e2e tests pass

- [ ] Verify no regressions in user-flows or input-chaos tests

Phase 3: Test Updates

Task 7: Update e2e tests for new validation rules

- Owner: B1
- BlockedBy: 6
- Files: `tests/e2e/input-chaos.spec.ts`, `tests/e2e/user-flows.spec.ts`
- Changes:
  1. Update password boundary tests — "12345678" now fails (no letter), use "pass1234" instead
  2. Update any test assertions that depend on old weak validation
  3. Add new positive test: valid registration with proper fields
  4. Verify all 75 tests pass with the new validation
- Acceptance:
  - [ ] `npx playwright test` — 75/75 pass
  - [ ] `npm test` — 120/120 pass
  - [ ] No test uses weak input that now gets rejected without updating the assertion

Task 8: Final validation — full test suite

- Owner: V1
- BlockedBy: 7
- Acceptance:
  - [ ] `npm test` — 120/120 pass (5 consecutive runs)
  - [ ] `npx playwright test` — 75/75 pass
  - [ ] Manual check: registration form rejects `<script>` in name
  - [ ] Manual check: registration form rejects `user@` email
  - [ ] Manual check: registration accepts valid Norwegian data

Files Modified

Backend (API)

- `src/app/api/auth/register/route.ts` — wire validators
- `src/app/api/transactions/remittance/route.ts` — decimal precision
- `src/app/api/transactions/qr-payment/route.ts` — decimal precision
- `src/lib/middleware/validation.ts` — add missing currencies

Frontend

- `src/app/onboarding/page.tsx` — proper client-side validation
- `src/app/login/page.tsx` — email format check

Tests

- `tests/e2e/input-chaos.spec.ts` — update for new rules
- `tests/e2e/user-flows.spec.ts` — update if needed

Validation Commands

```
# Unit + integration tests
npm test

# E2E tests (both suites)
npx playwright test

# Quick API validation
curl -s http://localhost:3000/api/auth/register \
  -X POST -H "Content-Type: application/json" \
  -d '{"email":"user@","password":"12345678","firstName":"<script>","lastName":"Test"}' | jq .
# Expected: 422 with validation errors
```

Risk Assessment

- **Low risk:** Wiring existing validators — they're already written and tested in isolation
- **Medium risk:** Frontend changes may break e2e test assertions that depend on old behavior
- **Mitigation:** Phase 3 explicitly updates tests after backend + frontend are done
- **NOT breaking existing users:** Password min stays at 8 (just adds letter+digit requirement). Demo user "demo1234" has both letters and digits — still works.

Revision #3
Created 2026-02-18 08:44:47 UTC by John
Updated 2026-05-24 20:00:51 UTC by John