drop-full-delivery-plan

Plan: Drop Full Delivery (#7187)

Research Summary

Full codebase audit completed (2026-04-06). Drop is 65-70% feature complete, 40% production ready.

What exists:

- Landing page with waitlist form (PostgreSQL-backed) — 13 languages
- Next.js web app: 27 pages, 72 API routes, CSRF/CSP security
- Hono/TS backend: 27 route modules, 20 DB tables, demo mode
- React Native/Expo mobile: 18 screens, BankID WebView, biometrics
- BankID mock + Bank mock (PISP/AISP simulation) — fully implemented
- Docker Compose (postgres, redis, api, app)
- 11 GitHub Actions workflows (CI/CD)
- 561 test files (mixed quality)

What's missing:

- Production deploy to Azure (currently Vercel/Fly.io)
- Comprehensive E2E test suite (~100 scenarios, currently ~3)
- CI/CD targeting Azure (currently Fly.io)
- Browser verification that bugs are fixed post-deploy
- Native mobile builds tested on device

CEO constraints:

- Deploy to Azure (NOT Vercel)
- Everything that exists must WORK in demo
- ~100 E2E scenarios covering all user flows
- CI/CD triggers Playwright on every push to dev/main
- Native apps for iOS + Android

Objective

Deploy Drop to Azure with a fully functional demo (BankID mock + Bank mock), 100+ E2E test scenarios with CI/CD integration, and verified native mobile builds.

Team Orchestration

Team Members

ID	Name	Role	Agent Type	Model
TL	Petter Graff	Team Lead / Architect	persona (petter-graff)	sonnet
B1	infra-builder	Azure deploy + Docker + CI/CD	builder (flowforge)	sonnet
B2	e2e-builder	Playwright E2E test suite (~100 scenarios)	builder (codecraft)	sonnet
B3	mobile-builder	Native mobile build verification + fixes	builder (paul-hudson)	sonnet
B4	landing-builder	Landing page Azure deploy + form verification	builder (frontend-builder)	sonnet
V1	drop-validator	Validate all phases	validator	sonnet
D1	Jake Wharton	Android advisory (consulted by B3)	persona (jake-wharton)	sonnet
D2	Thaer Sabri	Payments domain advisory (consulted by TL)	persona (thaer-sabri)	sonnet

Step-by-Step Tasks

Phase 1: Azure Infrastructure + Deploy (B1)

Task 1.1: Dockerize and deploy Drop full stack to Azure

- Owner: B1 (flowforge)
- BlockedBy: none
- Description:
  - Use existing Docker Compose as base (postgres:16, redis:7, drop-api, drop-app)
  - Deploy to Azure VM 4.223.110.181 or new Azure Container Apps
  - Configure DNS: app.getdrop.no → Azure (not Vercel)
  - Configure DNS: getdrop.no → Azure (static landing from landing/)

    - Set DROP_MODE=demo, SEED_DEMO=true
    - Ensure PostgreSQL persistent volume
    - Ensure Redis for rate limiting/sessions
    - Configure SSL/TLS (Let's Encrypt or Azure managed cert)
    - Set environment variables (DATABASE_URL, JWT_SECRET, PII_ENCRYPTION_KEY, etc.)
- Files owned: docker-compose.production.yml, infrastructure/, .github/workflows/deploy*.yml
- Acceptance:
  - ☐ app.getdrop.no loads on Azure (not white page)
  - ☐ getdrop.no landing loads with waitlist form
  - ☐ Waitlist form submits and stores in PostgreSQL
  - ☐ Demo login works at app.getdrop.no/login
  - ☐ Health endpoint returns 200: app.getdrop.no/api/health
  - ☐ SSL certificate valid

Task 1.2: Configure CI/CD for Azure deploy

- Owner: B1 (flowforge)
- BlockedBy: 1.1
- Description:
    - Update .github/workflows/ci.yml to run Playwright E2E on every push to dev/main
    - Create .github/workflows/deploy-azure.yml for production deploy to Azure
    - Trigger: push to main → build → test → deploy
    - Trigger: push to dev → build → test (no deploy)
    - Use GitHub Actions with SSH deploy or Azure CLI
- Files owned: .github/workflows/
- Acceptance:
  - ☐ Push to dev triggers lint + test + Playwright
  - ☐ Push to main triggers lint + test + Playwright + Azure deploy
  - ☐ Failed tests block deploy

Phase 2: E2E Test Suite (B2) — PARALLEL with Phase 1

Task 2.1: Create comprehensive Playwright E2E test suite (~100 scenarios)

- Owner: B2 (codecraft)
- BlockedBy: none (can write tests against local docker compose)
- Description:
    - Use existing playwright.config.ts as base
    - Create organized test structure in src/drop-app/tests/e2e/
    - All tests run against demo mode (DROP_MODE=demo)
    - Test categories and approximate scenario counts:

  Authentication (12 scenarios):

- Demo login happy path
- Demo login with different users (standard, merchant, admin)
- Login form validation (empty, invalid email, wrong password)
- Logout
- Session expiry handling
- CSRF protection verification
- Rate limiting on login
- BankID mock flow initiation
- Register new account
- Register validation (age <18, invalid phone)
- Auth redirect (unauthenticated user → login)
- Multiple device session handling

Onboarding (8 scenarios):

- Complete onboarding flow
- Skip optional steps
- KYC verification mock
- Phone number validation (+47 only)
- Age verification (must be 18+)
- Terms acceptance required
- Back navigation during onboarding
- Resume incomplete onboarding

Dashboard (8 scenarios):

- Dashboard loads with balance
- Recent transactions display
- Quick action buttons work
- Pull-to-refresh behavior
- Empty state (new user)
- Currency display (NOK)
- Navigation to all sections
- Notification badge count

Send Money / Remittance (15 scenarios):

- Send to existing recipient
- Add new recipient + send
- Amount validation (min, max, decimal)
- Currency selection (30+ countries)
- Fee calculation display (0.5%)
- Exchange rate display
- Confirmation screen
- Transaction success
- Transaction failure handling
- Receipt generation
- Send again from history
- Rate lock (30-min expiry)
- Insufficient balance handling
- AML threshold warning
- Cancel mid-transaction

QR Payments (8 scenarios):

- Generate QR code (merchant)
- Scan QR code (customer)
- QR payment confirmation
- QR payment success
- Invalid QR code handling
- QR amount pre-filled
- QR payment receipt
- Merchant dashboard after QR payment

Bank Accounts (6 scenarios):

- View linked accounts
- Link new bank account (AISP mock)
- Refresh balance
- Multiple accounts display
- Unlink account
- Consent renewal flow

Transaction History (10 scenarios):

- List all transactions
- Filter by date range
- Filter by type (sent/received)
- Search transactions
- Transaction detail view
- Receipt download/view
- CSV export
- Pagination/infinite scroll
- Empty state
- Transaction status indicators

Merchant (8 scenarios):

- Register as merchant
- Merchant dashboard loads
- QR code generation
- Sales overview
- Transaction list (merchant view)
- Settlement statement
- Merchant profile edit
- Merchant fee display

Profile & Settings (12 scenarios):

- View profile
- Edit personal details
- Change language (nb, en, etc.)
- Notification preferences toggle
- Security settings
- Biometric lock toggle (face unlock)
- Privacy policy accessible
- Terms of service accessible
- GDPR data export request

    - GDPR objection/rectification
    - Help/FAQ accessible
    - Complaint filing

  Notifications (5 scenarios):
    - Notification list loads
    - Mark as read
    - Mark all as read
    - Unread count badge
    - Empty notifications state

  Edge Cases & Error Handling (8 scenarios):
    - Network error handling
    - 500 server error display
    - 404 page
    - Session expired mid-action
    - Concurrent tab handling
    - Mobile viewport responsiveness
    - Accessibility (keyboard navigation)
    - Input sanitization (XSS prevention)

  TOTAL: ~100 scenarios
- Files owned: src/drop-app/tests/e2e/**
- Acceptance:
  - ☐ 100+ test scenarios written
  - ☐ All tests pass against local Docker compose (demo mode)
  - ☐ Tests organized by feature area
  - ☐ Each test has descriptive name
  - ☐ Tests are independent (no order dependency)
  - ☐ Test report generated (HTML reporter)

Phase 3: Landing Page Azure Deploy (B4) — PARALLEL with Phase 1 & 2

Task 3.1: Deploy landing page to Azure

- Owner: B4 (frontend-builder)
- BlockedBy: none
- Description:
  - Landing page is static HTML (landing/index.html + api/waitlist.js)
  - Deploy static files via Nginx on Azure VM or Azure Static Web Apps
  - Waitlist API needs Node.js runtime (serverless or Express wrapper)
  - Configure getdrop.no DNS → Azure
  - Ensure DATABASE_URL for waitlist PostgreSQL
  - Verify form submission works end-to-end
- Files owned: landing/, nginx configs

- Acceptance:
  - ☐ getdrop.no loads landing page
  - ☐ Waitlist form submits successfully
  - ☐ Email stored in PostgreSQL waitlist_signups table
  - ☐ All 13 language variants accessible
  - ☐ SSL certificate valid
  - ☐ Mobile responsive

Phase 4: Mobile Build Verification (B3) — PARALLEL

Task 4.1: Verify and fix React Native mobile builds

- Owner: B3 (paul-hudson persona for iOS guidance)
- BlockedBy: none
- Description:
  - Verify iOS build compiles (EAS Build)
  - Verify Android build compiles (EAS Build)
  - Test demo mode on iOS Simulator
  - Test demo mode on Android Emulator
  - Verify BankID WebView integration in demo mode
  - Verify biometric auth flow
  - Fix any build errors
  - Document build steps
  - Consult jake-wharton persona for Android-specific issues (Samsung Knox, biometric)
- Files owned: src/drop-mobile/
- Acceptance:
  - ☐ iOS build compiles without errors
  - ☐ Android build compiles without errors
  - ☐ Demo login works on iOS Simulator
  - ☐ Demo login works on Android Emulator
  - ☐ Navigation through all 18 screens works
  - ☐ Biometric auth prompt appears

Phase 5: Validation (V1) — AFTER Phases 1-4

Task 5.1: Validate entire Drop deployment

- Owner: V1 (validator)
- BlockedBy: 1.1, 1.2, 2.1, 3.1, 4.1
- Description:
  - Browser verification: app.getdrop.no loads and demo works

- Browser verification: getdrop.no form submits
- Run full Playwright E2E suite against Azure deployment
- Verify CI/CD pipeline triggers correctly
- Verify mobile builds
- Check HiveMind updates from all builders
- Run qa-19.js quality gate

- Acceptance:
  - [ ] app.getdrop.no renders (NOT white page) — Playwright screenshot
  - [ ] getdrop.no waitlist form submits — Playwright screenshot
  - [ ] E2E suite: 90%+ pass rate on Azure
  - [ ] CI/CD: test push to dev triggers workflow
  - [ ] Mobile: builds compile
  - [ ] qa-19.js score >= 17/19

Validation Commands

```
# Health check
curl -s https://app.getdrop.no/api/health | jq .

# Landing page
curl -s -o /dev/null -w "%{http_code}" https://getdrop.no

# Waitlist form
curl -s -X POST https://getdrop.no/api/waitlist -H "Content-Type: application/json" -d
'{"email":"test@test.com"}'

# Run E2E tests against Azure
cd ~/ALAI/products/Drop/src/drop-app && BASE_URL=https://app.getdrop.no npx playwright test

# CI/CD verification
gh workflow list -R ALAI/Drop
gh run list -R ALAI/Drop --limit 5

# QA gate
node ~/system/tools/qa-19.js check 7187

# Mobile builds
```

```
cd ~/ALAI/products/Drop/src/drop-mobile && eas build --platform all --profile preview --non-interactive
```

Execution Order

```
Phase 1 (B1: Azure infra)  ──┐
Phase 2 (B2: E2E tests)    ──┼──→ Phase 5 (V1: Validate all)
Phase 3 (B4: Landing)      ──┤
Phase 4 (B3: Mobile)       ──┘
```

All 4 builder phases run IN PARALLEL. Validator runs after all complete.

Risk Mitigations

Risk	Mitigation
Azure VM resource limits (4GB RAM)	Use Container Apps if needed, or scale VM
DNS propagation delay	Pre-configure Azure IP, use low TTL
Playwright flaky on CI	Use retry: 1, serial mode for auth tests
Mobile EAS build queue	Use local builds as fallback
Hook permission blocks	Builders use worktree isolation

Notes

- Backend is Hono/TS (migration to Kotlin pending MC #5124) — we deploy CURRENT stack, not migrated
- BankID + Bank mock already exist — no need to build from scratch
- Demo mode (DROP_MODE=demo) is the target, not production with real banking
- Cards feature stays disabled (feature-flagged off)

Revision #2
Created 2026-04-06 15:00:09 UTC by John
Updated 2026-05-31 20:05:39 UTC by John