drop-analytics-bi-spec

Drop Analytics & Business Intelligence — Architecture Specification

Version: 1.0 **Date:** 2026-02-17 **Author:** architect agent (Sonnet 4.5) **Status:** Draft **MC Task:** #1195

Executive Summary

This document specifies the analytics and business intelligence system for Drop, a Norwegian fintech payment app operating under PSD2 pass-through model. The system must track user engagement, transaction metrics, conversion funnels, and operational health while maintaining strict GDPR compliance (Datatilsynet requirements).

Recommendation: PostHog self-hosted (EU instance) for event tracking + Custom SQLite/PostgreSQL queries for financial KPIs + Admin dashboard built on existing Next.js stack.

---

1. Platform Decision

1.1 Requirements Matrix

Requirement	Weight	PostHog (Self-hosted)	Mixpanel	Custom (SQL + Dashboard)
GDPR compliance (Datatilsynet)	CRITICAL	Data stays in Norway	⚠ US-based, SCCs required	Full control
PSD2 audit trail	CRITICAL	⚠ Not built for fintech	⚠ Not built for fintech	Native to DB
Cost (10K MAU)	HIGH	€0 (self-hosted)	~$1000/month	€0 (existing infra)

Requirement	Weight	PostHog (Self-hosted)	Mixpanel	Custom (SQL + Dashboard)
Real-time dashboards	MEDIUM	Built-in	Built-in	⚠ Build from scratch
Session replay	LOW	Built-in	Paid extra	N/A
Funnel analysis	HIGH	Built-in	Built-in	⚠ Custom SQL
Setup complexity	MEDIUM	Medium (Docker)	Low (SaaS)	Low (existing stack)
Financial metrics	CRITICAL	Not fintech-specific	Not fintech-specific	Direct DB access

1.2 Decision: Hybrid Approach

Recommended Stack:

1. **PostHog (self-hosted)** — User behavior, funnels, session replay, A/B testing
2. **Custom SQL queries** — Financial KPIs (transaction volume, revenue, error rates)
3. **Admin Dashboard (Next.js)** — Internal BI dashboard at `/admin/analytics`

Rationale:

- **GDPR:** Self-hosted PostHog keeps data in Norway (Datatilsynet compliant)
- **PSD2 Audit:** Financial metrics pulled directly from `transactions`, `audit_log` tables (source of truth)
- **Cost:** PostHog self-hosted is free (infrastructure cost only), custom queries have zero marginal cost
- **Separation of concerns:** Product analytics (PostHog) vs financial compliance (SQL)

Cost Analysis:

Solution	Setup	Monthly Cost (10K MAU)
PostHog Cloud	1 day	~€299/month (EU hosting)
PostHog Self-hosted	2 days	~€50/month (Fly.io 2GB RAM)
Mixpanel	1 day	~$1000/month
Custom only	5 days	€0 (existing infra)
Hybrid (recommended)	**3 days**	**~€50/month**

---

2. KPI Definitions (Precise Logic)

2.1 User Engagement Metrics

DAU (Daily Active Users)

Definition: Unique users who performed any action in the app within the last 24 hours.

SQL Query:

```
SELECT COUNT(DISTINCT user_id) as dau
FROM audit_log
WHERE timestamp >= datetime('now', '-1 day');
```

PostHog Event: Track `app_opened` event on every session start.

WAU (Weekly Active Users)

Definition: Unique users active in the last 7 days.

SQL Query:

```
SELECT COUNT(DISTINCT user_id) as wau
FROM audit_log
WHERE timestamp >= datetime('now', '-7 days');
```

MAU (Monthly Active Users)

Definition: Unique users active in the last 30 days.

SQL Query:

```
SELECT COUNT(DISTINCT user_id) as mau
FROM audit_log
WHERE timestamp >= datetime('now', '-30 days');
```

Stickiness Ratio

Definition: DAU/MAU — measures how frequently users return.

Target: >20% (industry benchmark for fintech apps)

Calculation:

```
WITH daily AS (
  SELECT COUNT(DISTINCT user_id) as dau
```

```
  FROM audit_log
  WHERE timestamp >= datetime('now', '-1 day')
),
monthly AS (
  SELECT COUNT(DISTINCT user_id) as mau
  FROM audit_log
  WHERE timestamp >= datetime('now', '-30 days')
)
SELECT ROUND((daily.dau * 100.0 / monthly.mau), 2) as stickiness_pct
FROM daily, monthly;
```

2.2 Transaction Metrics

Transaction Volume (Count)

Definition: Total number of completed transactions per day.

SQL Query:

```
SELECT
  DATE(created_at) as date,
  COUNT(*) as tx_count
FROM transactions
WHERE status = 'completed'
  AND created_at >= datetime('now', '-30 days')
GROUP BY DATE(created_at)
ORDER BY date DESC;
```

Transaction Value (Amount)

Definition: Total monetary value of completed transactions per day (in NOK).

Note: All amounts stored as øre (1 NOK = 100 øre).

SQL Query:

```
SELECT
  DATE(created_at) as date,
  SUM(amount) / 100.0 as total_nok,
  COUNT(*) as tx_count,
```

```
  AVG(amount) / 100.0 as avg_nok
FROM transactions
WHERE status = 'completed'
  AND created_at >= datetime('now', '-30 days')
GROUP BY DATE(created_at)
ORDER BY date DESC;
```

Transaction Volume by Type

Definition: Split by `remittance` vs `qr_payment`.

SQL Query:

```
SELECT
  DATE(created_at) as date,
  type,
  COUNT(*) as tx_count,
  SUM(amount) / 100.0 as total_nok
FROM transactions
WHERE status = 'completed'
  AND created_at >= datetime('now', '-30 days')
GROUP BY DATE(created_at), type
ORDER BY date DESC, type;
```

Average Transaction Value (ATV)

Definition: Mean transaction amount per completed transaction.

SQL Query:

```
SELECT
  DATE(created_at) as date,
  AVG(amount) / 100.0 as avg_tx_nok
FROM transactions
WHERE status = 'completed'
  AND created_at >= datetime('now', '-30 days')
GROUP BY DATE(created_at)
ORDER BY date DESC;
```

2.3 Conversion Funnel

Funnel Stages

1. **Register** — User creates account (`/register` page, `register_success` audit action)
2. **Verify KYC** — User completes BankID verification (`kyc_approved` audit action)
3. **Link Bank** — User connects bank account via AISP (`bank_account_linked` audit action)
4. **First Transfer** — User completes first transaction (`first_transaction` audit action)

Funnel SQL Query:

```
WITH funnel AS (
  SELECT
    COUNT(DISTINCT CASE WHEN action = 'register_success' THEN user_id END) as registered,
    COUNT(DISTINCT CASE WHEN action = 'kyc_approved' THEN user_id END) as kyc_verified,
    COUNT(DISTINCT CASE WHEN action = 'bank_account_linked' THEN user_id END) as bank_linked,
    COUNT(DISTINCT CASE WHEN action = 'first_transaction' THEN user_id END) as first_tx
  FROM audit_log
  WHERE timestamp >= datetime('now', '-30 days')
)
SELECT
  registered,
  kyc_verified,
  ROUND(kyc_verified * 100.0 / registered, 2) as kyc_conversion_pct,
  bank_linked,
  ROUND(bank_linked * 100.0 / kyc_verified, 2) as bank_conversion_pct,
  first_tx,
  ROUND(first_tx * 100.0 / bank_linked, 2) as tx_conversion_pct,
  ROUND(first_tx * 100.0 / registered, 2) as overall_conversion_pct
FROM funnel;
```

PostHog Funnel:

- Event 1: `register_success`
- Event 2: `kyc_approved`
- Event 3: `bank_account_linked`
- Event 4: `first_transaction`

Time to First Transaction (TTFT)

Definition: Median time from registration to first completed transaction.

SQL Query:

```
WITH user_events AS (
  SELECT
    user_id,
    MIN(CASE WHEN action = 'register_success' THEN timestamp END) as registered_at,
    MIN(CASE WHEN action = 'first_transaction' THEN timestamp END) as first_tx_at
  FROM audit_log
  WHERE timestamp >= datetime('now', '-90 days')
  GROUP BY user_id
  HAVING first_tx_at IS NOT NULL
)
SELECT
  AVG(JULIANDAY(first_tx_at) - JULIANDAY(registered_at)) * 24 * 60 as avg_minutes,
  MIN(JULIANDAY(first_tx_at) - JULIANDAY(registered_at)) * 24 * 60 as min_minutes,
  MAX(JULIANDAY(first_tx_at) - JULIANDAY(registered_at)) * 24 * 60 as max_minutes
FROM user_events;
```

Target: <10 minutes (industry best practice for fintech onboarding)

2.4 Revenue Metrics

Gross Revenue

Definition: Total fees collected (before costs).

SQL Query:

```
SELECT
  DATE(created_at) as date,
  SUM(fee) / 100.0 as gross_revenue_nok
FROM transactions
WHERE status = 'completed'
  AND created_at >= datetime('now', '-30 days')
GROUP BY DATE(created_at)
ORDER BY date DESC;
```

Revenue by Type

Definition: Fee breakdown by transaction type.

SQL Query:

```
SELECT
  type,
  SUM(fee) / 100.0 as revenue_nok,
  COUNT(*) as tx_count,
  AVG(fee) / 100.0 as avg_fee_nok
FROM transactions
WHERE status = 'completed'
  AND created_at >= datetime('now', '-30 days')
GROUP BY type;
```

Average Revenue Per User (ARPU)

Definition: Total revenue divided by active users in period.

SQL Query:

```
WITH revenue AS (
  SELECT SUM(fee) / 100.0 as total_revenue
  FROM transactions
  WHERE status = 'completed'
    AND created_at >= datetime('now', '-30 days')
),
active_users AS (
  SELECT COUNT(DISTINCT user_id) as mau
  FROM audit_log
  WHERE timestamp >= datetime('now', '-30 days')
)
SELECT
  total_revenue,
  mau,
  ROUND(total_revenue / mau, 2) as arpu_nok
FROM revenue, active_users;
```

2.5 Error Rate Metrics

API Error Rate (by endpoint)

Definition: Percentage of API requests returning 5xx errors.

SQL Query (requires structured logging with `request_id`):

```
-- Assumes logger.ts logs API requests to audit_log with details JSON
SELECT
  JSON_EXTRACT(details, '$.endpoint') as endpoint,
  COUNT(*) as total_requests,
  SUM(CASE WHEN JSON_EXTRACT(details, '$.status') >= 500 THEN 1 ELSE 0 END) as error_count,
  ROUND(SUM(CASE WHEN JSON_EXTRACT(details, '$.status') >= 500 THEN 1 ELSE 0 END) * 100.0 /
COUNT(*), 2) as error_rate_pct
FROM audit_log
WHERE action = 'api_request'
  AND timestamp >= datetime('now', '-1 day')
GROUP BY JSON_EXTRACT(details, '$.endpoint')
HAVING error_count > 0
ORDER BY error_rate_pct DESC;
```

Sentry Integration: Pull error counts from Sentry API (already implemented in Drop).

Transaction Failure Rate

Definition: Percentage of transactions that fail.

SQL Query:

```
SELECT
  DATE(created_at) as date,
  COUNT(*) as total_tx,
  SUM(CASE WHEN status = 'failed' THEN 1 ELSE 0 END) as failed_tx,
  ROUND(SUM(CASE WHEN status = 'failed' THEN 1 ELSE 0 END) * 100.0 / COUNT(*), 2) as
failure_rate_pct
FROM transactions
WHERE created_at >= datetime('now', '-30 days')
GROUP BY DATE(created_at)
ORDER BY date DESC;
```

Target: <2% (industry benchmark for payment apps)

2.6 Retention Metrics

D1, D7, D30 Retention

Definition: Percentage of users who return 1, 7, or 30 days after registration.

SQL Query (D7 retention):

```
WITH cohort AS (
  SELECT
    user_id,
    DATE(MIN(timestamp)) as cohort_date
  FROM audit_log
  WHERE action = 'register_success'
  GROUP BY user_id
),
activity AS (
  SELECT
    user_id,
    DATE(timestamp) as activity_date
  FROM audit_log
  WHERE timestamp >= datetime('now', '-90 days')
)
SELECT
  cohort.cohort_date,
  COUNT(DISTINCT cohort.user_id) as cohort_size,
  COUNT(DISTINCT CASE
    WHEN JULIANDAY(activity.activity_date) - JULIANDAY(cohort.cohort_date) BETWEEN 6 AND 8
    THEN activity.user_id
  END) as retained_d7,
  ROUND(COUNT(DISTINCT CASE
    WHEN JULIANDAY(activity.activity_date) - JULIANDAY(cohort.cohort_date) BETWEEN 6 AND 8
    THEN activity.user_id
  END) * 100.0 / COUNT(DISTINCT cohort.user_id), 2) as retention_d7_pct
FROM cohort
LEFT JOIN activity ON cohort.user_id = activity.user_id
WHERE cohort.cohort_date >= datetime('now', '-90 days')
GROUP BY cohort.cohort_date
ORDER BY cohort.cohort_date DESC;
```

PostHog: Built-in retention analysis feature (preferred for visualization).

---

3. Event Taxonomy (PostHog)

3.1 Core Events

Event Name	Triggered When	Properties
`app_opened`	User opens app (any session start)	`user_id`, `device_type`, `os_version`
`page_viewed`	User navigates to any page	`user_id`, `page_path`, `referrer`
`register_started`	User clicks "Register" button	`user_id`, `source` (organic, referral)
`register_success`	Registration API returns 201	`user_id`, `phone_country_code`
`login_success`	Login API returns JWT	`user_id`
`login_failed`	Login API returns 401	`reason` (invalid_pin, not_found)
`kyc_started`	User starts BankID flow	`user_id`, `kyc_method` (bankid)
`kyc_approved`	KYC status set to 'approved'	`user_id`, `kyc_method`
`bank_account_linked`	User links bank via AISP	`user_id`, `bank_name`
`transaction_initiated`	User submits payment form	`user_id`, `type` (remittance, qr_payment), `amount_nok`
`transaction_completed`	Transaction status = 'completed'	`user_id`, `tx_id`, `type`, `amount_nok`, `fee_nok`
`transaction_failed`	Transaction status = 'failed'	`user_id`, `tx_id`, `type`, `error_code`
`first_transaction`	User completes first-ever transaction	`user_id`, `type`, `amount_nok`
`notification_received`	Push notification sent	`user_id`, `notification_type`
`notification_clicked`	User opens notification	`user_id`, `notification_type`
`qr_scanned`	User scans QR code	`user_id`, `merchant_id`
`settings_changed`	User updates settings	`user_id`, `setting_key`

3.2 Error Events

Event Name	Triggered When	Properties
`api_error`	API returns 5xx error	`user_id`, `endpoint`, `status_code`, `error_message`
`network_error`	Client-side network failure	`user_id`, `endpoint`
`validation_error`	Form validation fails	`user_id`, `form_name`, `field_name`

3.3 Custom Properties (User-level)

Property	Source	Purpose
`user_id`	DB `users.id`	Identify user across sessions
`kyc_status`	DB `users.kyc_status`	Segment by verification state
`role`	DB `users.role`	Segment by user/merchant
`risk_level`	DB `users.risk_level`	AML segmentation
`created_at`	DB `users.created_at`	Cohort analysis
`first_transaction_at`	DB custom query	Activation metric

---

4. Database Schema (Analytics Extensions)

4.1 New Table: `analytics_events`

Purpose: Store custom events not captured in `audit_log` (e.g., client-side events before auth).

```
CREATE TABLE IF NOT EXISTS analytics_events (
  id TEXT PRIMARY KEY,
  event_name TEXT NOT NULL,
  user_id TEXT REFERENCES users(id), -- NULL for pre-auth events
  session_id TEXT,
  properties TEXT, -- JSON blob
  device_type TEXT,
  os_version TEXT,
  app_version TEXT,
  ip_address TEXT,
  user_agent TEXT,
  created_at TEXT DEFAULT (datetime('now'))
);

CREATE INDEX IF NOT EXISTS idx_analytics_events_user ON analytics_events(user_id);
CREATE INDEX IF NOT EXISTS idx_analytics_events_event ON analytics_events(event_name);
CREATE INDEX IF NOT EXISTS idx_analytics_events_created ON analytics_events(created_at);
```

GDPR Note: `ip_address` is PII — anonymize last octet (e.g., `192.168.1.0`).

4.2 New View: `daily_metrics`

Purpose: Materialized view for fast dashboard queries (regenerate daily via cron).

```
CREATE VIEW IF NOT EXISTS daily_metrics AS
SELECT
  DATE(created_at) as date,
  COUNT(DISTINCT user_id) as dau,
  COUNT(DISTINCT CASE WHEN type = 'remittance' THEN id END) as remittance_count,
  COUNT(DISTINCT CASE WHEN type = 'qr_payment' THEN id END) as qr_payment_count,
  SUM(CASE WHEN status = 'completed' THEN amount ELSE 0 END) / 100.0 as total_volume_nok,
  SUM(CASE WHEN status = 'completed' THEN fee ELSE 0 END) / 100.0 as total_revenue_nok,
  AVG(CASE WHEN status = 'completed' THEN amount END) / 100.0 as avg_tx_nok,
  SUM(CASE WHEN status = 'failed' THEN 1 ELSE 0 END) as failed_tx_count,
  ROUND(SUM(CASE WHEN status = 'failed' THEN 1 ELSE 0 END) * 100.0 / COUNT(*), 2) as
failure_rate_pct
FROM transactions
WHERE created_at >= datetime('now', '-365 days')
GROUP BY DATE(created_at);
```

Optimization: For large datasets (>1M transactions), create a materialized table instead of a view.

4.3 New View: `funnel_stages`

Purpose: Pre-aggregated funnel data for dashboard.

```
CREATE VIEW IF NOT EXISTS funnel_stages AS
SELECT
  user_id,
  MIN(CASE WHEN action = 'register_success' THEN timestamp END) as registered_at,
  MIN(CASE WHEN action = 'kyc_approved' THEN timestamp END) as kyc_at,
  MIN(CASE WHEN action = 'bank_account_linked' THEN timestamp END) as bank_linked_at,
  MIN(CASE WHEN action = 'first_transaction' THEN timestamp END) as first_tx_at
FROM audit_log
WHERE action IN ('register_success', 'kyc_approved', 'bank_account_linked',
'first_transaction')
GROUP BY user_id;
```

---

5. API Endpoints (Admin Dashboard)

5.1 Admin Authentication

Middleware: Extend `requireAuth()` to check `role = 'admin'` (new role in `users` table).

Route: `/api/admin/*` — All admin endpoints protected by `requireAdmin()` middleware.

5.2 KPI Endpoints

GET `/api/admin/analytics/overview`

Description: High-level dashboard summary.

Response:

```
{
  "data": {
    "dau": 1234,
    "wau": 5678,
    "mau": 10234,
    "stickiness_pct": 12.05,
    "total_users": 15234,
    "total_transactions": 45678,
    "total_volume_nok": 1234567.89,
    "total_revenue_nok": 12345.67,
    "avg_tx_nok": 270.45,
    "failure_rate_pct": 1.23
  }
}
```

GET `/api/admin/analytics/transactions?period=30d`

Description: Transaction metrics over time.

Query Params:

- `period` — `7d`, `30d`, `90d`, `1y`

Response:

```
{
  "data": [
    {
      "date": "2026-02-17",
      "tx_count": 234,
      "volume_nok": 67890.12,
      "revenue_nok": 678.90,
      "avg_tx_nok": 290.17,
      "failure_rate_pct": 1.28
    },
    // ... more days
  ]
}
```

GET `/api/admin/analytics/funnel?period=30d`

Description: Conversion funnel metrics.

Response:


```
{
  "data": {
    "registered": 1000,
    "kyc_verified": 850,
    "kyc_conversion_pct": 85.0,
    "bank_linked": 720,
    "bank_conversion_pct": 84.71,
    "first_tx": 640,
    "tx_conversion_pct": 88.89,
    "overall_conversion_pct": 64.0
  }
}
```

GET `/api/admin/analytics/retention?cohort_date=2026-01-01`

Description: Cohort retention analysis.

Response:

```
{
  "data": {
    "cohort_date": "2026-01-01",
    "cohort_size": 500,
    "d1_retained": 350,
    "d1_retention_pct": 70.0,
    "d7_retained": 200,
    "d7_retention_pct": 40.0,
    "d30_retained": 150,
    "d30_retention_pct": 30.0
  }
}
```

GET `/api/admin/analytics/errors?period=24h`

Description: Error rate by endpoint.

Response:

```
{
  "data": [
    {
      "endpoint": "/api/transactions",
      "total_requests": 5000,
      "error_count": 25,
      "error_rate_pct": 0.5
    },
    // ... more endpoints
  ]
}
```

5.3 Rate Limiting

Admin endpoints: 100 requests/minute per admin user (higher than public API).

Implementation: Reuse existing `rate_limits` table + middleware.

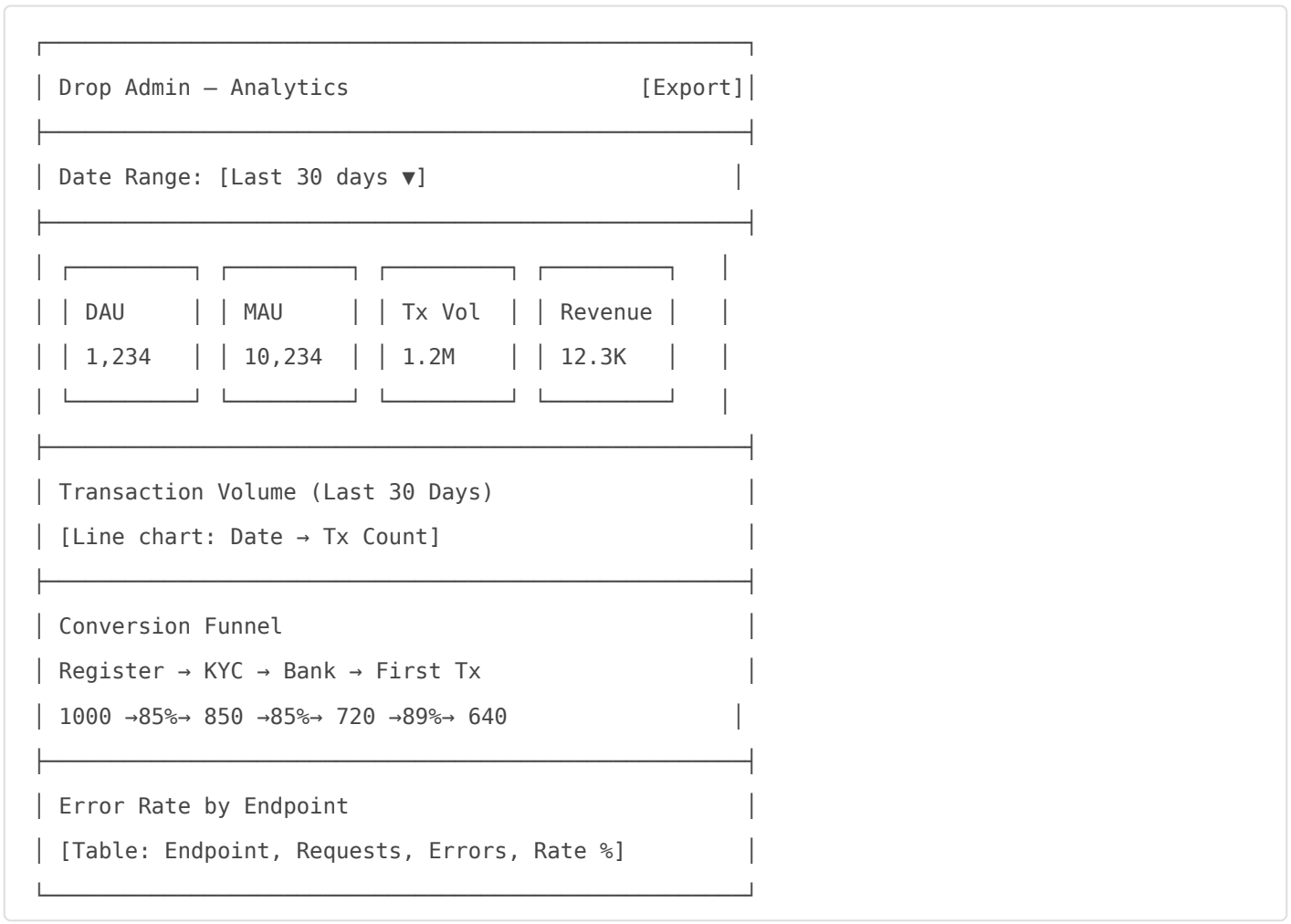
---

6. Dashboard Spec (Admin UI)

6.1 Route: `/admin/analytics`

Access: Role-based (only `role = 'admin'` users can access).

Layout:



6.2 Chart Library

Recommendation: Recharts (React-based, lightweight, no external dependencies).

Alternative: Chart.js (more features, heavier).

Install:

```
npm install recharts
```

6.3 Export Feature

Button: "Export CSV" — Generates CSV of current view for Excel analysis.

Implementation: Client-side CSV generation (no server processing).

6.4 Filters

Filter	Options
Date Range	Last 7 days, Last 30 days, Last 90 days, Custom
Transaction Type	All, Remittance, QR Payment
User Segment	All, Verified, Unverified
Country (Remittance)	All, RS (Serbia), BA (Bosnia), TR (Turkey), etc.

---

7. Privacy & GDPR Compliance (Datatilsynet)

7.1 What's Tracked

Personal Data Tracked

Data	Purpose	Legal Basis	Retention
`user_id`	Product analytics	Legitimate interest (GDPR Art. 6(1)(f))	Until account deletion
`ip_address` (anonymized)	Fraud prevention	Legitimate interest	90 days
`device_type`, `os_version`	Product optimization	Legitimate interest	Until account deletion
`transaction_amount`	Business metrics	Contract performance (GDPR Art. 6(1)(b))	5 years (PSD2 requirement)

Anonymized Data

- Last octet of IP address replaced with `0` (e.g., `192.168.1.123` → `192.168.1.0`)
- PostHog `$ip` property disabled (no IP tracking)
- Session recordings disabled by default (opt-in only for support cases)

7.2 What's NOT Tracked

Explicitly excluded from analytics:

- `password`, `pin` (never logged anywhere)
- `bank_account`, `iban` (PII, not needed for analytics)
- `national_id_hash` (sensitive PII)
- Full IP addresses (only anonymized)
- Email addresses (not needed for event tracking)

7.3 User Rights (GDPR)

Right to Access (Art. 15)

Implementation: Export all analytics events for a user via `/api/admin/analytics/user/{user_id}/export`.

Format: JSON file with all events, properties, timestamps.

Right to Erasure (Art. 17)

Implementation:

- Delete from `analytics_events` table: `DELETE FROM analytics_events WHERE user_id = ?`
- PostHog: Use PostHog API to delete user: `POST /api/person/{person_id}/delete`
- Audit log: Retain for 5 years (legal requirement), but anonymize `user_id` → `anonymized_{timestamp}`

Right to Object (Art. 21)

Implementation: Add `analytics_consent` field to `consents` table. If withdrawn, stop tracking events.

7.4 Datatilsynet Requirements

Data Processing Agreement (DPA)

PostHog self-hosted: No DPA needed (data stays in-house).

PostHog Cloud (if used): DPA required (PostHog provides standard agreement).

Data Localization

Requirement: PSD2 payment data must stay in EEA.

Implementation: PostHog self-hosted on Fly.io EU region (Frankfurt or Amsterdam).

Consent Banner

Required: Display cookie consent banner on landing page (already implemented in `src/components/cookie-consent.tsx`).

Categories:

- **Necessary:** Session cookies, auth tokens (no consent required)
- **Analytics:** PostHog tracking (opt-in required)
- **Marketing:** None (Drop doesn't use third-party ads)

7.5 Audit Trail (PSD2 Requirement)

Requirement: All analytics queries accessing financial data must be logged.

Implementation:

```
-- Log every admin analytics query
INSERT INTO audit_log (id, user_id, action, resource_type, details, ip_address)
VALUES (
  ?,
  ?, -- admin user_id
  'admin_analytics_query',
  'analytics',
  JSON_OBJECT('endpoint', '/api/admin/analytics/overview', 'params', '{}'),
  ?
);
```

Retention: 5 years (PSD2 Art. 97).

---

8. Real-time vs Batch Processing

8.1 Real-time Metrics

Metric	Source	Latency
DAU/WAU/MAU	`audit_log` query	<100ms
Transaction count (today)	`transactions` query	<100ms
Error rate (last hour)	Sentry API	~5 seconds
Current active users	PostHog live view	Real-time

Implementation: Direct SQL queries on every dashboard load (no caching needed for <100K transactions).

8.2 Batch Processing (Daily)

Metric	Source	Schedule
Daily aggregates (`daily_metrics` view)	Cron job	00:05 UTC
Retention cohorts	Cron job	01:00 UTC
Funnel snapshots	Cron job	02:00 UTC

Implementation:

```
# Cron job (runs inside Docker container)
0 0 * * * node /app/scripts/analytics/daily-aggregates.js
```

Script: Queries last 24h of data, writes to `daily_metrics` table (denormalized for speed).

8.3 Caching Strategy

Admin dashboard: Cache API responses for 5 minutes (Vercel Edge Cache or Redis).

Reason: Admin dashboards don't need real-time updates, 5-minute staleness is acceptable.

Implementation:

```
// src/app/api/admin/analytics/overview/route.ts
export const revalidate = 300; // 5 minutes
```

---

9. Alerting (Integration with Existing Slack Alerts)

9.1 Anomaly Detection

Triggered Alerts:

1. **Transaction volume spike** — >2x daily average
2. **Transaction volume drop** — <50% daily average
3. **Error rate spike** — >5% failure rate
4. **Conversion funnel drop** — >20% decrease in any stage

Implementation: Cron job checks thresholds every hour, sends Slack alert via `src/lib/alerts.ts`.

9.2 Alert Format (Slack)

```
□□ALERT: Transaction Volume Spike
Volume: 5,234 transactions (avg: 2,100)
Spike: +150% from daily average
Time: 2026-02-17 14:35 UTC
Link: https://drop.no/admin/analytics
```

9.3 Thresholds (Configurable)

Config file: `src/config/analytics-alerts.json`

```
{
  "transaction_volume_spike_multiplier": 2.0,
  "transaction_volume_drop_threshold": 0.5,
  "error_rate_threshold_pct": 5.0,
  "funnel_drop_threshold_pct": 20.0
}
```

---

10. Cost Analysis

10.1 PostHog Self-Hosted (Recommended)

Infrastructure:

- **Fly.io 2GB RAM instance:** ~€40/month
- **PostgreSQL (for PostHog):** Included (shared instance)
- **ClickHouse (analytics DB):** Included (single-node)

Total: ~€40-50/month for 10K MAU

Scalability: Up to 100K MAU on same instance (ClickHouse is highly efficient).

10.2 PostHog Cloud (Alternative)

Pricing: €299/month for 10K MAU (EU hosting).

Pros: Zero maintenance, auto-scaling.

Cons: 6x more expensive than self-hosted.

10.3 Custom Only (No PostHog)

Cost: €0 (runs on existing infrastructure).

Effort: 5 additional days to build funnel analysis, session tracking, etc.

Recommendation: Not worth the time savings — PostHog provides 90% of features out-of-the-box.

---

11. Implementation Plan (Phased Approach)

Phase 1: Foundation (Day 1-2)

Goal: Basic KPI queries + Admin API endpoints.

Tasks:

1. Add `analytics_events` table to schema
2. Create `daily_metrics` and `funnel_stages` views
3. Implement 5 core API endpoints:
   - `/api/admin/analytics/overview`
   - `/api/admin/analytics/transactions`
   - `/api/admin/analytics/funnel`
   - `/api/admin/analytics/retention`
   - `/api/admin/analytics/errors`
4. Add `requireAdmin()` middleware

Deliverable: Working API endpoints (testable via curl/Postman).

Phase 2: Admin Dashboard (Day 3-4)

Goal: Internal BI dashboard with charts.

Tasks:

1. Create `/admin/analytics` page
2. Implement 4 chart components (Recharts):
   - KPI summary cards
   - Transaction volume line chart
   - Conversion funnel bar chart
   - Error rate table
3. Add date range filter
4. Add CSV export button

Deliverable: Functional admin dashboard (internal use only).

Phase 3: PostHog Integration (Day 5-6)

Goal: Self-hosted PostHog for user behavior tracking.

Tasks:

1. Deploy PostHog to Fly.io (Docker Compose)
2. Integrate PostHog client SDK (`posthog-js`)
3. Instrument 10 core events (see Event Taxonomy)
4. Set up funnel analysis in PostHog UI
5. Configure GDPR settings (disable IP tracking, session recordings opt-in)

Deliverable: PostHog dashboard with live user events.

Phase 4: Alerting (Day 7)

Goal: Automated anomaly detection.

Tasks:

1. Create `scripts/analytics/anomaly-detector.js`
2. Implement threshold checks (see Alerting section)
3. Integrate with `src/lib/alerts.ts` (Slack)
4. Set up cron job (hourly)

Deliverable: Slack alerts for transaction spikes/drops.

Phase 5: GDPR Compliance (Day 8)

Goal: User data export and deletion.

Tasks:

1. Implement `/api/admin/analytics/user/{user_id}/export`
2. Implement user deletion in `analytics_events` table
3. Add PostHog person deletion API call
4. Update cookie consent banner to include analytics opt-in
5. Document data retention policy

Deliverable: GDPR-compliant analytics system.

Total Effort: 8 days (1 developer)

MVP (Phases 1-2 only): 4 days (if PostHog is deferred)

---

12. Testing Strategy

12.1 Unit Tests

File: `src/lib/__tests__/analytics.test.ts`

Coverage:

- KPI calculation functions (DAU, MAU, ATV, etc.)

- SQL query builders
- Date range parsing

Tool: Vitest (already used in Drop).

12.2 Integration Tests

File: `tests/integration/admin-analytics.test.ts`

Coverage:

- `/api/admin/analytics/*` endpoints
- Response format validation
- Role-based access control (non-admin should get 403)

12.3 E2E Tests

File: `tests/e2e/admin-dashboard.spec.ts`

Coverage:

- Admin login → Navigate to `/admin/analytics`
- Charts render correctly
- Date range filter works
- CSV export downloads

Tool: Playwright (already used in Drop).

---

13. Monitoring & Observability

13.1 PostHog Self-Monitoring

Health Check: `/api/posthog/health` — Check PostHog is reachable.

Metrics:

- Event ingestion rate (events/second)
- Query latency (avg response time for dashboards)
- Database size (ClickHouse disk usage)

Tool: PostHog built-in `/instance/status` page.

13.2 Admin Dashboard Monitoring

Sentry: Track errors in admin dashboard (already integrated).

Alert: If admin API error rate >5%, send Slack alert.

---

14. Security Considerations

14.1 Admin Access Control

Role: Add `admin` role to `users` table.

Migration:

```
-- Add admin role to existing user
UPDATE users SET role = 'admin' WHERE email = 'alem@drop.no';
```

Middleware:

```
export async function requireAdmin() {
  const { user, error } = await requireAuth();
  if (error) return error;
  if (user.role !== 'admin') {
    return jsonError('forbidden', 'Admin access required', 403);
  }
  return { user };
}
```

14.2 Sensitive Data Exposure

Risk: Admin dashboard exposes transaction amounts, user IDs.

Mitigation:

- Admin routes protected by `requireAdmin()` middleware
- No PII in chart labels (use aggregated data only)

- Audit log all admin queries (see GDPR section)

14.3 SQL Injection (Admin Queries)

Risk: Dynamic date range filtering could be exploited.

Mitigation:

- Use parameterized queries ONLY (no string concatenation)
- Whitelist allowed period values (`7d`, `30d`, `90d`, `1y`)

Example (SAFE):

```
const allowedPeriods = { '7d': '-7 days', '30d': '-30 days' };
const period = allowedPeriods[req.query.period] || '-30 days';
const sql = `SELECT COUNT(*) FROM transactions WHERE created_at >= datetime('now', ?)`;
const result = await query(sql, [period]);
```

---

15. Future Enhancements (Post-MVP)

15.1 Predictive Analytics

Goal: Forecast transaction volume using ML.

Tool: TensorFlow.js (client-side) or Prophet (Python backend).

Use Case: Predict cash flow for liquidity planning (relevant when Drop holds funds in future).

15.2 Real-time Dashboard (WebSockets)

Goal: Live-updating dashboard without refresh.

Tool: Socket.io or Server-Sent Events (SSE).

Use Case: Monitor transaction spikes during marketing campaigns.

15.3 A/B Testing

Goal: Test onboarding flow variations.

Tool: PostHog Feature Flags + Experiments.

Use Case: Test "Skip KYC for low-value transactions" flow.

15.4 Custom Funnels (Dynamic)

Goal: Allow admins to create custom funnels in UI.

Tool: PostHog Funnels API.

Use Case: Analyze drop-off in specific remittance corridors (e.g., Norway → Serbia).

---

16. Success Metrics (How to Measure Analytics System)

Metric	Target	Measurement
Dashboard load time	<2 seconds	Chrome DevTools Performance
API response time (p95)	<500ms	Sentry Performance Monitoring
PostHog event ingestion lag	<10 seconds	PostHog /instance/status
Admin dashboard uptime	>99.5%	UptimeRobot (monitor `/admin/analytics`)
GDPR compliance audit	100% pass	Manual checklist (see section 7)

---

17. References

17.1 Norwegian Regulations

- **Datatilsynet (Data Protection Authority):** https://www.datatilsynet.no/en/
- **PSD2 Audit Requirements:** Regulation (EU) 2018/389 (RTS on SCA and CSC)
- **Financial Supervision Authority (Finanstilsynet):** https://www.finanstilsynet.no/en/

17.2 Tools & Libraries

- **PostHog:** https://posthog.com/docs/self-host
- **Recharts:** https://recharts.org/en-US/
- **Sentry:** https://docs.sentry.io/platforms/javascript/guides/nextjs/

17.3 Industry Benchmarks

- **Fintech KPIs:** Andreessen Horowitz "16 Startup Metrics" (a16z.com)
- **Conversion Funnel Benchmarks:** Mixpanel "Product Benchmarks Report 2025"
- **Retention Standards:** Lenny's Newsletter "Retention Benchmarks by Industry"

---

18. Appendix: SQL Query Examples

A. Top 10 Users by Transaction Volume (Last 30 Days)

```
SELECT
  u.id,
  u.first_name || ' ' || u.last_name as name,
  COUNT(t.id) as tx_count,
  SUM(t.amount) / 100.0 as total_volume_nok,
  SUM(t.fee) / 100.0 as total_fees_nok
FROM users u
JOIN transactions t ON u.id = t.user_id
WHERE t.status = 'completed'
  AND t.created_at >= datetime('now', '-30 days')
GROUP BY u.id
ORDER BY total_volume_nok DESC
LIMIT 10;
```

B. Transaction Volume by Country (Remittance Corridors)

```
SELECT
  r.country,
  COUNT(t.id) as tx_count,
  SUM(t.amount) / 100.0 as total_sent_nok,
  AVG(t.amount) / 100.0 as avg_tx_nok
FROM transactions t
JOIN recipients r ON t.recipient_id = r.id
WHERE t.type = 'remittance'
  AND t.status = 'completed'
  AND t.created_at >= datetime('now', '-30 days')
GROUP BY r.country
ORDER BY tx_count DESC;
```

C. Hourly Transaction Distribution (Peak Hours)

```
SELECT
  CAST(strftime('%H', created_at) AS INTEGER) as hour,
  COUNT(*) as tx_count,
  SUM(amount) / 100.0 as volume_nok
FROM transactions
WHERE status = 'completed'
  AND created_at >= datetime('now', '-7 days')
GROUP BY hour
ORDER BY hour;
```

---

19. Approval & Sign-off

Prepared by: architect agent (Sonnet 4.5) **Reviewed by:** [Pending — John] **Approved by:** [Pending — Alem]

Next Steps:

1. Review spec with Alem (prioritize phases)
2. Create MC tasks for each phase
3. Assign builder agent for Phase 1 implementation

Document Status: Draft (awaiting approval) **Last Updated:** 2026-02-17 **Version:** 1.0

Revision #4
Created 2026-02-18 08:44:43 UTC by John
Updated 2026-06-07 20:00:19 UTC by John