Architecture

- Architecture Document
- API Specification

Architecture Document

Architecture Document: Drop

Version: 1.1 **Date:** 2026-02-08 **Author:** dev agent (qwen2.5-coder:32b) + John **Status:** Approved **Approved by:** John (AI Director)

---

1. Overview

1.1 System Purpose

Drop je fintech aplikacija za remittance i QR plaćanja za sve stanovnike Norveške i Skandinavije. Drop koristi PSD2 pass-through model — nikada ne drži novac korisnika. AISP čita stanje računa putem Open Banking-a, a PISP inicira plaćanja direktno sa bankovnog računa korisnika.

1.2 Architecture Style

Monolith — scope je 7 stranica + API rute. Microservices bi bio over-engineering za demo.

1.3 Key Design Decisions

Decision	Choice	Rationale	Alternatives
Architecture	Monolith	Small scope, simple deploy	Microservices (overkill)
Frontend	Next.js 16 + React 19	Already built, modern stack	Remix, SvelteKit
Styling	Tailwind v4	Already in use	Styled Components
Database	PostgreSQL 16 (Drizzle ORM)	Full test/prod parity, PostgreSQL-native features, type-safe schema	SQLite (superseded by ADR-014)
Auth	JWT via jose	Lightweight, stateless	Session-based (needs Redis)
JWT Storage	httpOnly cookie	Prevents XSS token theft	localStorage (less secure)
Error Handling	Centralized middleware	Consistent responses, easy logging	Per-route try/catch

1.4 User Requirements (ENFORCED — from vilkår.html)

These are legally binding requirements published in our Terms of Service. They MUST be enforced in code.

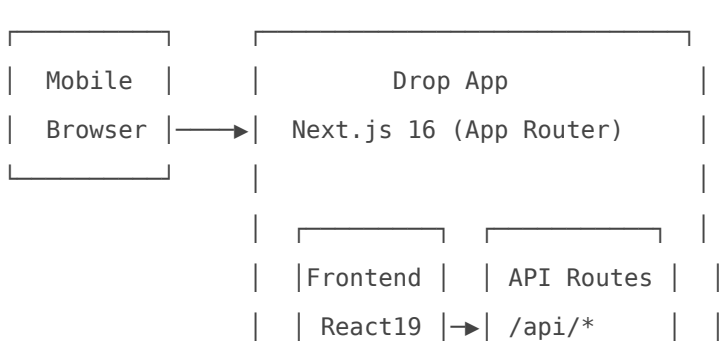
Requirement	Value	Enforcement
Minimum age	18 år	Registration: DOB field → reject if < 18. BankID returns DOB → double-check.
Residency	Bosatt i Norge	Registration: Norwegian phone (+47) + Norwegian BankID required.
Identity verification	Gyldig BankID	Onboarding: BankID verification mandatory before any transaction.
Accurate personal data	User obligation	BankID provides verified name/DOB. User confirms address.
No illegal use	User obligation	AML monitoring, transaction limits, suspicious activity detection.

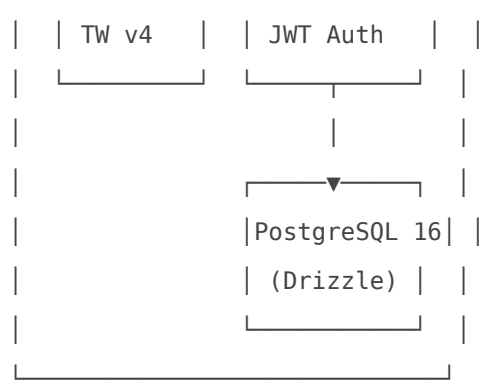
Source: `landing/pages/vilkar.html` section 3 — "Du må være minst 18 år og bosatt i Norge for å bruke Drop."

Implementation notes:

- BankID returns fødselsnummer (11-digit) which encodes DOB → extract and validate age >= 18
- In demo/MVP: mock BankID with DOB field, enforce 18+ check in `/api/auth/register`
- Pass-through model: Drop never holds money, uses Open Banking (PSD2) to read balance and initiate transfers

2. System Context





2.1 External Interfaces

Currently self-contained with local PostgreSQL (Docker). No external service integrations in MVP.

System	Purpose	Status
Exchange rates	Remittance corridors (RSD, BAM, PLN, PKR, TRY, EUR)	Local DB table
Auth	JWT via jose	Built-in
QR payments	Merchant scanning	Built-in

Post-MVP roadmap (FUTURE — not yet implemented, requires partners):

- Open Banking provider (PSD2 AISP/PISP) for production bank account access
- Card issuing provider (Stripe or similar) for physical/virtual cards — gated behind feature flags
- KYC provider (Sumsub/Onfido or partner's existing system)

3. Component Architecture

3.1 Frontend Pages

Page	Route	Description	Status
Landing	/	Marketing page	Core
Register	/register	Phone + PIN registration	Core
Login	/login	Phone + PIN auth	Core
Dashboard	/dashboard	Account overview, last 5 transactions	Core
Send Money	/send	Remittance — PISP from user's bank account	Core

Page	Route	Description	Status
QR Payments	/scan	Scan & pay via QR code — PISP from user's bank account	Core
Bank Accounts	/accounts	View linked bank account balances via AISP	Core
Transaction History	/transactions	Full transaction list with filters	Core
Notifications	/notifications	Push notifications and transaction alerts	Core
Settings	/profile	User preferences and account management	Core
Cards	/cards	Virtual/physical card management	FUTURE (feature-flagged)

3.2 API Layer

```
/api
  /auth
    /register/route.ts    — POST register (phone + PIN)
    /login/route.ts       — POST login → JWT in httpOnly cookie
  /account/route.ts       — GET balance, account info
  /transactions
    /route.ts             — GET list, POST send money
    /simulate/route.ts    — POST simulate incoming (demo)
  /cards                          — FUTURE (feature-flagged, requires partner)
    /route.ts             — GET cards, POST create virtual
    /[id]/route.ts        — PATCH freeze/unfreeze
    /[id]/physical/route.ts — POST order physical
  middleware/
    errorHandler.ts       — Centralized error responses
    authMiddleware.ts     — JWT verification
```

4. Data Architecture

4.1 Database

- **Engine:** PostgreSQL 16 (all environments — development, CI, staging, production)

- **ORM:** Drizzle ORM (`src/shared/db/schema.ts` — single source of truth)
- **Local dev:** Docker (`docker compose up -d`), port 5433
- **Rationale:** Full test/prod parity, type-safe schema, PostgreSQL-native features (ADR-014)

4.2 Schema

Total Tables: 19 (12 core + 7 compliance)

Core Tables (12)

Table	Key Fields	Relationships
users	id, email, password_hash, first_name, last_name, phone, date_of_birth, kyc_status, role, risk_level, pep_status, sanctions_cleared, kyc_method, kyc_verified_at, national_id_hash, deleted_at, created_at	→ bank_accounts, cards, recipients, transactions
bank_accounts	id, user_id, bank_name, account_number, iban, balance (cached AISP read), balance_synced_at, currency, is_primary, connected_at	→ users, transactions
cards	id, user_id, type, last_four, token_ref, expiry, status, shipping_address, pin_hash, created_at	→ users — **FUTURE (feature-flagged)**
transactions	id, user_id, type, status, amount, currency, fee, recipient_id, merchant_id, send_amount, send_currency, receive_amount, receive_currency, exchange_rate, purpose_code, created_at, completed_at	→ users, recipients, merchants
recipients	id, user_id, name, country, currency, bank_account, bank_name, created_at	→ users, transactions
merchants	id, user_id, business_name, org_number, address, bank_account, fee_rate, status, created_at	→ users, transactions
exchange_rates	id, from_currency, to_currency, rate, updated_at	—
sessions	id, user_id, token_hash, created_at, expires_at, revoked	→ users
notifications	id, user_id, type, title, body, read, created_at	→ users

Table	Key Fields	Relationships
settings	user_id, currency, language, push_enabled, email_enabled, updated_at	→ users
spending_limits	id, user_id, card_id, limit_type, amount, currency, created_at	→ users, cards
rate_limits	key, count, reset_at	—

Compliance Tables (7) — Added 2026-02-16

Table	Key Fields	Purpose
audit_log	id, timestamp, user_id, action, resource_type, resource_id, details, ip_address, user_agent	Audit trail of all user actions
aml_alerts	id, user_id, alert_type, severity, transaction_id, details, status, reviewed_by, reviewed_at, created_at	Anti-money laundering alert tracking
str_reports	id, user_id, alert_id, report_type, status, filed_at, reference_number, details, created_at	Suspicious transaction reports (SAR/STR)
screening_results	id, user_id, screening_type, provider, result, match_details, screened_at	PEP, sanctions, adverse media screening
consents	id, user_id, consent_type, granted, granted_at, withdrawn_at, ip_address	GDPR consent tracking (PSD2, marketing, data processing)
data_access_requests	id, user_id, request_type, status, requested_at, completed_at, download_url, notes	GDPR right to access/erasure/rectification
complaints	id, user_id, category, subject, description, status, resolution, created_at, resolved_at	Customer complaint handling

> **Pass-through model:** Drop NEVER holds customer money. The `bank_accounts.balance` field is a cached AISP read from the user's actual bank account (read-only in production, synced via Open Banking). User funds remain in their bank at all times. PISP initiates payments directly from user's bank account.

> **No wallet, no top-up:** Drop does not have a wallet feature or top-up functionality. Users do not maintain a balance with Drop.

4.3 PSD2 Pass-Through Model

Drop operates as a PSD2 Payment Initiation Service Provider (PISP) and Account Information Service Provider (AISP):

AISP (Account Information)

- **Purpose:** Read user's bank account balance and transaction history
- **Method:** Open Banking API via BankID consent
- **Storage:** Cached balance in `bank_accounts.balance` (read-only, synced periodically)
- **Note:** Drop never controls or holds this balance

PISP (Payment Initiation)

- **Purpose:** Initiate payments directly from user's bank account
- **Use cases:** Remittance transfers, QR merchant payments
- **Method:** Open Banking payment initiation with Strong Customer Authentication (SCA)
- **Flow:** User approves payment → PISP initiates → Bank debits user's account → Drop records transaction

Compliance Requirements (PSD2)

1. **User consent:** Explicit BankID consent required for AISP + PISP access
2. **SCA (Strong Customer Authentication):** Required for all payments
3. **Data minimization:** Only store what's necessary for compliance
4. **Audit trail:** All PISP/AISP operations logged in `audit_log` table
5. **Right to withdraw consent:** Tracked in `consents` table

Regulatory tables: audit_log, aml_alerts, str_reports, screening_results, consents, data_access_requests, complaints

5. Security Architecture (from security agent threat model)

5.1 Authentication

- **Method:** JWT (jose library)
- **Storage:** httpOnly cookie (NOT localStorage)
- **Expiry:** 1h access token

- **PIN:** bcrypt hashed, never stored plain

5.2 Threats & Mitigations

Threat	Severity	Mitigation
Broken Access Control	HIGH	JWT middleware on all /api routes
SQL Injection	HIGH	Parameterized queries via Drizzle ORM
XSS	HIGH	React auto-escapes, CSP headers
Token Theft	HIGH	httpOnly cookie, HTTPS
CSRF	MEDIUM	SameSite cookie + CSRF token
Data in localStorage	HIGH	Move sensitive data to httpOnly cookies
Replay Attacks	MEDIUM	Token expiration + jti claim
Security Misconfiguration	HIGH	Security headers (HSTS, X-Frame, CSP)

5.3 Data Protection

- **In Transit:** HTTPS/TLS (when deployed)
- **At Rest:** AWS RDS AES-256 encryption (TLS 1.3 in transit to DB)
- **PII:** Phone numbers hashed in logs

6. Infrastructure

Environment	Purpose	URL
Development	Local dev	localhost:3000
Staging	Pre-release	TBD (Vercel preview)
Production	Live demo	TBD (Vercel)

CI/CD Pipeline

```
Push → Build (next build) → TypeScript Check → Lint → Test → Deploy Staging → Manual Approval → Deploy Prod
```

7. Technology Stack

Layer	Technology	Version
Frontend	Next.js	16
UI	React	19
Styling	Tailwind CSS	4
Backend	Next.js API Routes	16
Database	PostgreSQL 16 + Drizzle ORM	16
Auth	JWT (jose)	latest
Hosting	Vercel	—

8. Performance Targets

Metric	Target
FCP	< 1.5s
LCP	< 2.5s
TTFB	< 200ms
API p95	< 300ms
Lighthouse	> 90
Build time	< 60s

9. ADRs

ADR-001: SQLite over PostgreSQL (superseded)

- **Date:** 2026-02-08
- **Status:** Superseded by ADR-014
- **Context:** Demo app needs simple DB setup
- **Decision (original):** SQLite — zero config, file-based
- **Consequence:** Could not handle concurrent writes well. Superseded before production use.

- **Current state:** PostgreSQL 16 in all environments (development, CI, staging, production). See ADR-014.

ADR-002: JWT in httpOnly Cookie

- **Date:** 2026-02-08
- **Status:** Accepted
- **Context:** Need secure token storage
- **Decision:** httpOnly cookie prevents XSS token theft
- **Consequence:** Slightly more complex CSRF handling needed.

ADR-003: Monolith Architecture

- **Date:** 2026-02-08
- **Status:** Accepted
- **Context:** 7 pages, simple API
- **Decision:** Single Next.js app handles everything
- **Consequence:** Easy to deploy and maintain. Refactor if scaling needed.

10. Approvals

Role	Name	Date	Approved
Dev Agent	dev (qwen2.5-coder:32b)	2026-02-08	□
Security Agent	security (qwen2.5-coder:32b)	2026-02-08	□
John (AI Director)	John	2026-02-08	□

API Specification

API Specification: Drop

Version: 1.0 **Date:** 2026-02-09 **Author:** dev agent (Ollama) + John (AI Director) **Base URL:** `/api` (Next.js API Routes) **Auth:** JWT in httpOnly cookie (jose library) **Database:** PostgreSQL 16 via Drizzle ORM (ADR-014; `better-sqlite3` removed 2026-03-03)

---

1. Overview

API Style: REST **Format:** JSON **Rate Limiting:** 60 req/min per IP (standard), 10 req/min for auth endpoints **Auth mechanism:** JWT token set as httpOnly, secure, sameSite=strict cookie

---

2. Authentication

POST /api/auth/register

Description: Register new user account

Request:

```
{
  "email": "amir@example.com",
  "password": "min8chars",
  "firstName": "Amir",
  "lastName": "Hadžić",
  "phone": "+4712345678"
}
```

Response 201:

```
{
  "data": {
```

```
    "id": "usr_abc123",
    "email": "amir@example.com",
    "firstName": "Amir",
    "lastName": "Hadžić",
    "kycStatus": "pending",
    "createdAt": "2026-02-09T10:00:00Z"
  }
}
```

Sets httpOnly JWT cookie

Errors: 400 (validation), 409 (email exists)

POST /api/auth/login

Description: Login and receive JWT cookie

Request:

```
{
  "email": "amir@example.com",
  "password": "min8chars"
}
```

Response 200:

```
{
  "data": {
    "id": "usr_abc123",
    "email": "amir@example.com",
    "firstName": "Amir",
    "lastName": "Hadžić",
    "kycStatus": "approved"
  }
}
```

Sets httpOnly JWT cookie (24h expiry). Note: balance is NOT returned here — use /api/bank-accounts (AISP) to read bank balance.

Errors: 401 (wrong credentials), 423 (account locked)

POST /api/auth/logout

Description: Clear JWT cookie **Auth:** Required

Response 200:

```
{ "message": "Logged out" }
```

Clears httpOnly cookie

GET /api/auth/me

Description: Get current user from JWT **Auth:** Required

Response 200:

```
{
  "data": {
    "id": "usr_abc123",
    "email": "amir@example.com",
    "firstName": "Amir",
    "lastName": "Hadžić",
    "kycStatus": "approved",
    "createdAt": "2026-02-09T10:00:00Z"
  }
}
```

---

3. Bank Accounts (AISP — Pass-through)

> **Pass-through model:** Drop never holds customer money. Balance is read from user's real bank account via Open Banking (AISP). Payments are initiated via PISP from user's bank.

GET /api/bank-accounts

Description: Get linked bank accounts and balances via AISP (Open Banking) **Auth:** Required (BankID consent)

Response 200:

```
{
  "data": [
    {
      "id": "ba_1",
      "bankName": "SpareBank 1",
      "accountNumber": "*****1234",
      "balance": 12450.00,
      "currency": "NOK",
      "lastSynced": "2026-02-09T10:00:00Z"
    }
  ]
}
```

Note: Balance is a cached AISP read from the user's actual bank account. Drop does not store or manage this balance.

~~GET /api/users/balance~~ — REMOVED

~~POST /api/users/top-up~~ — REMOVED

> These endpoints were part of the old wallet model and have been removed. In the pass-through model, there is no wallet to check or top up. Use `/api/bank-accounts` to read bank balances via AISP.

---

4. Recipients

GET /api/recipients

Description: List user's saved recipients **Auth:** Required **Query:** `?page=1&limit=20`

Response 200:

```
{
  "data": [
    {
      "id": "rec_1",
      "name": "Mama Jasmina",
      "country": "RS",
      "countryName": "Serbia",
      "currency": "RSD",
      "bankAccount": "*****1234",
      "createdAt": "2026-02-01T10:00:00Z"
    }
  ],
  "pagination": {
    "page": 1,
    "limit": 20,
    "total": 3
  }
}
```

POST /api/recipients

Description: Add new recipient **Auth:** Required

Request:


```
{
  "name": "Mama Jasmina",
  "country": "RS",
  "currency": "RSD",
  "bankAccount": "265100000012345678",
  "bankName": "Banca Intesa"
}
```


Response 201:


```
{
  "data": {
    "id": "rec_new",
    "name": "Mama Jasmina",
```

```
    "country": "RS",
    "currency": "RSD",
    "bankAccount": "*****5678",
    "createdAt": "2026-02-09T10:00:00Z"
  }
}
```

Errors: 400 (validation), 422 (unsupported country)

DELETE /api/recipients/:id

Description: Remove recipient **Auth:** Required **Response 204:** No content

---

5. Exchange Rates

GET /api/rates

Description: Get current exchange rates for all corridors **Auth:** Not required

Response 200:

```
{
  "data": {
    "baseCurrency": "NOK",
    "rates": {
      "RSD": 11.70,
      "BAM": 1.04,
      "PLN": 0.41,
      "PKR": 26.80,
      "TRY": 3.45,
      "EUR": 0.089
    },
    "updatedAt": "2026-02-09T10:00:00Z"
  }
}
```

GET /api/rates/:currency

Description: Get rate for specific currency pair **Auth:** Not required

Response 200:

```
{
  "data": {
    "from": "NOK",
    "to": "RSD",
    "rate": 11.70,
    "fee": 0.005,
    "updatedAt": "2026-02-09T10:00:00Z"
  }
}
```

Errors: 404 (unsupported currency)

---

6. Transactions — Remittance

POST /api/transactions/remittance

Description: Create new remittance transfer **Auth:** Required (KYC must be approved)

Request:

```
{
  "recipientId": "rec_1",
  "amount": 2000.00,
  "currency": "NOK"
}
```

Response 201:

```
{
  "data": {
    "id": "tx_rem_123",
    "type": "remittance",
    "status": "processing",
```

```
    "sendAmount": 2000.00,
    "sendCurrency": "NOK",
    "receiveAmount": 23400.00,
    "receiveCurrency": "RSD",
    "exchangeRate": 11.70,
    "fee": 10.00,
    "feePercent": 0.5,
    "total": 2010.00,
    "recipientName": "Mama Jasmina",
    "recipientCountry": "RS",
    "eta": "1-2 business days",
    "createdAt": "2026-02-09T10:00:00Z"
  }
}
```


Errors:

- 400 — invalid amount (min 100 NOK, max 50000 NOK)
- 402 — insufficient balance
- 403 — KYC not approved
- 404 — recipient not found
- 422 — unsupported corridor

---

7. Transactions — QR Payment

POST /api/transactions/qr-payment

Description: Pay a merchant via QR code **Auth:** Required

Request:

```
{
  "merchantId": "mer_1",
  "amount": 129.00
}
```

Response 201:

```
{
  "data": {
    "id": "tx_qr_456",
    "type": "qr_payment",
    "status": "completed",
    "amount": 129.00,
    "currency": "NOK",
    "fee": 1.29,
    "feePercent": 1.0,
    "merchantName": "Ahmetov Kebab",
    "merchantId": "mer_1",
    "createdAt": "2026-02-09T14:23:00Z"
  }
}
```


Errors:

- 400 — invalid amount (min 1 NOK)
- 402 — insufficient balance
- 404 — merchant not found

---

8. Transactions — List

GET /api/transactions

Description: List user's transactions (both remittance and QR) **Auth:** Required **Query:** `?page=1&limit=20&type=remittance|qr_payment&status=completed|processing|failed`

Response 200:


```
{
  "data": [
    {
      "id": "tx_rem_123",
      "type": "remittance",
      "status": "completed",
      "amount": -2000.00,
      "currency": "NOK",
      "recipientName": "Mama Jasmina",
```

```
      "createdAt": "2026-02-09T10:00:00Z"
    },
    {
      "id": "tx_qr_456",
      "type": "qr_payment",
      "status": "completed",
      "amount": -129.00,
      "currency": "NOK",
      "merchantName": "Ahmetov Kebab",
      "createdAt": "2026-02-09T14:23:00Z"
    }
  ],
  "pagination": {
    "page": 1,
    "limit": 20,
    "total": 47
  }
}
```

GET /api/transactions/:id

Description: Get transaction detail **Auth:** Required

Response 200:


```
{
  "data": {
    "id": "tx_rem_123",
    "type": "remittance",
    "status": "completed",
    "sendAmount": 2000.00,
    "sendCurrency": "NOK",
    "receiveAmount": 23400.00,
    "receiveCurrency": "RSD",
    "exchangeRate": 11.70,
    "fee": 10.00,
    "total": 2010.00,
    "recipientName": "Mama Jasmina",
    "recipientCountry": "RS",
```

```
    "createdAt": "2026-02-09T10:00:00Z",
    "completedAt": "2026-02-10T14:30:00Z"
  }
}
```

---

9. Merchants

POST /api/merchants/register

Description: Register as a merchant **Auth:** Required

Request:

```
{
  "businessName": "Ahmetov Kebab",
  "orgNumber": "923456789",
  "address": "Grønland 12, Oslo",
  "bankAccount": "1234.56.78901"
}
```

Response 201:

```
{
  "data": {
    "id": "mer_1",
    "businessName": "Ahmetov Kebab",
    "orgNumber": "923456789",
    "qrCode": "drop://pay/mer_1",
    "status": "active",
    "feeRate": 0.01,
    "createdAt": "2026-02-09T10:00:00Z"
  }
}
```

Errors: 400 (validation), 409 (org number exists)

GET /api/merchants/dashboard

Description: Get merchant stats **Auth:** Required (merchant role) **Query:** `?period=today|week|month`

Response 200:


```
{
  "data": {
    "period": "today",
    "revenue": 4350.00,
    "transactionCount": 12,
    "fees": 43.50,
    "netRevenue": 4306.50,
    "nextPayout": 4306.50,
    "payoutTime": "17:00"
  }
}
```

GET /api/merchants/transactions

Description: List merchant's received payments **Auth:** Required (merchant role) **Query:** `?page=1&limit=20&date=2026-02-09`

Response 200:


```
{
  "data": [
    {
      "id": "tx_qr_456",
      "customerName": "Amir K.",
      "amount": 129.00,
      "fee": 1.29,
      "net": 127.71,
      "status": "completed",
      "time": "14:23"
    }
  ],
  "pagination": {
    "page": 1,
    "limit": 20,
    "total": 12
```

```
  }
}
```

GET /api/merchants/qr

Description: Get merchant's QR code data **Auth:** Required (merchant role)

Response 200:

```
{
  "data": {
    "merchantId": "mer_1",
    "businessName": "Ahmetov Kebab",
    "qrValue": "drop://pay/mer_1",
    "address": "Grønland 12, Oslo"
  }
}
```

---

10. Database Schema (PostgreSQL 16 — 19 tables)

Source: `src/shared/db/schema.ts` (Drizzle ORM schema — PostgreSQL 16, all environments; `better-sqlite3` removed per ADR-014)

Core Tables (12)

> **Note:** The SQL below is a historical snapshot from the original spec (SQLite syntax). The authoritative schema is `src/shared/db/schema.ts` (Drizzle ORM, PostgreSQL 16). Use `make db-push` to apply schema changes.

```
-- Users (NO balance field — pass-through model)
CREATE TABLE users (
  id TEXT PRIMARY KEY,
  email TEXT UNIQUE NOT NULL,
```

```
  password_hash TEXT NOT NULL,
  first_name TEXT NOT NULL,
  last_name TEXT NOT NULL,
  phone TEXT,
  date_of_birth TEXT,
  kyc_status TEXT DEFAULT 'pending' CHECK(kyc_status IN ('pending','approved','rejected')),
  role TEXT DEFAULT 'user' CHECK(role IN ('user','merchant')),
  risk_level TEXT DEFAULT 'low' CHECK(risk_level IN ('low','medium','high')),
  pep_status TEXT DEFAULT 'not_checked' CHECK(pep_status IN
('not_checked','clear','match','pending_review')),
  sanctions_cleared INTEGER DEFAULT 0,
  kyc_method TEXT CHECK(kyc_method IN ('bankid','document','simplified')),
  kyc_verified_at TEXT,
  national_id_hash TEXT,
  deleted_at TEXT,
  created_at TEXT DEFAULT (datetime('now'))
);

-- Bank accounts (balance is cached AISP read, NOT held by Drop)
CREATE TABLE bank_accounts (
  id TEXT PRIMARY KEY,
  user_id TEXT NOT NULL REFERENCES users(id),
  bank_name TEXT NOT NULL,
  account_number TEXT NOT NULL,
  iban TEXT,
  balance REAL DEFAULT 0,  -- Cached AISP-read balance (read-only in production)
  balance_synced_at TEXT,  -- When balance was last synced from bank via AISP
  currency TEXT DEFAULT 'NOK',
  is_primary INTEGER DEFAULT 0,
  connected_at TEXT DEFAULT (datetime('now'))
);

CREATE TABLE recipients (
  id TEXT PRIMARY KEY,
  user_id TEXT NOT NULL REFERENCES users(id),
  name TEXT NOT NULL,
  country TEXT NOT NULL,
  currency TEXT NOT NULL,
  bank_account TEXT NOT NULL,
```

```
  bank_name TEXT,
  created_at TEXT DEFAULT (datetime('now'))
);

CREATE TABLE merchants (
  id TEXT PRIMARY KEY,
  user_id TEXT NOT NULL REFERENCES users(id),
  business_name TEXT NOT NULL,
  org_number TEXT UNIQUE NOT NULL,
  address TEXT,
  bank_account TEXT NOT NULL,
  fee_rate REAL DEFAULT 0.01,
  status TEXT DEFAULT 'active',
  created_at TEXT DEFAULT (datetime('now'))
);

CREATE TABLE transactions (
  id TEXT PRIMARY KEY,
  user_id TEXT NOT NULL REFERENCES users(id),
  type TEXT NOT NULL CHECK(type IN ('remittance','qr_payment')),
  status TEXT DEFAULT 'processing' CHECK(status IN ('processing','completed','failed')),
  amount REAL NOT NULL,
  currency TEXT DEFAULT 'NOK',
  fee REAL DEFAULT 0,
  recipient_id TEXT REFERENCES recipients(id),
  merchant_id TEXT REFERENCES merchants(id),
  send_amount REAL,
  send_currency TEXT,
  receive_amount REAL,
  receive_currency TEXT,
  exchange_rate REAL,
  purpose_code TEXT,
  created_at TEXT DEFAULT (datetime('now')),
  completed_at TEXT
);

CREATE TABLE exchange_rates (
  id INTEGER PRIMARY KEY AUTOINCREMENT,  -- SERIAL for PostgreSQL
  from_currency TEXT DEFAULT 'NOK',
```

```
  to_currency TEXT NOT NULL,
  rate REAL NOT NULL,
  updated_at TEXT DEFAULT (datetime('now'))
);

CREATE TABLE cards (
  id TEXT PRIMARY KEY,
  user_id TEXT NOT NULL REFERENCES users(id),
  type TEXT DEFAULT 'virtual' CHECK(type IN ('virtual','physical')),
  last_four TEXT NOT NULL,
  token_ref TEXT,
  expiry TEXT NOT NULL,
  status TEXT DEFAULT 'active' CHECK(status IN ('active','frozen','cancelled')),
  shipping_address TEXT,
  pin_hash TEXT,
  created_at TEXT DEFAULT (datetime('now'))
);

CREATE TABLE sessions (
  id TEXT PRIMARY KEY,
  user_id TEXT NOT NULL REFERENCES users(id),
  token_hash TEXT NOT NULL,
  created_at TEXT DEFAULT (datetime('now')),
  expires_at TEXT NOT NULL,
  revoked INTEGER DEFAULT 0
);

CREATE TABLE notifications (
  id TEXT PRIMARY KEY,
  user_id TEXT NOT NULL REFERENCES users(id),
  type TEXT NOT NULL,
  title TEXT NOT NULL,
  body TEXT NOT NULL,
  read INTEGER DEFAULT 0,
  created_at TEXT DEFAULT (datetime('now'))
);

CREATE TABLE settings (
  user_id TEXT PRIMARY KEY REFERENCES users(id),
```

```
  currency TEXT DEFAULT 'NOK',
  language TEXT DEFAULT 'nb',
  push_enabled INTEGER DEFAULT 1,
  email_enabled INTEGER DEFAULT 1,
  updated_at TEXT DEFAULT (datetime('now'))
);

CREATE TABLE spending_limits (
  id TEXT PRIMARY KEY,
  user_id TEXT NOT NULL REFERENCES users(id),
  card_id TEXT REFERENCES cards(id),
  limit_type TEXT NOT NULL,
  amount REAL NOT NULL,
  currency TEXT DEFAULT 'NOK',
  created_at TEXT DEFAULT (datetime('now'))
);

CREATE TABLE rate_limits (
  key TEXT PRIMARY KEY,
  count INTEGER NOT NULL,
  reset_at INTEGER NOT NULL
);
```

Compliance Tables (7) — Added 2026-02-16

```
CREATE TABLE audit_log (
  id TEXT PRIMARY KEY,
  timestamp TEXT DEFAULT (datetime('now')),
  user_id TEXT REFERENCES users(id),
  action TEXT NOT NULL,
  resource_type TEXT,
  resource_id TEXT,
  details TEXT,
  ip_address TEXT,
  user_agent TEXT
);

CREATE TABLE aml_alerts (
  id TEXT PRIMARY KEY,
```

```
  user_id TEXT NOT NULL REFERENCES users(id),
  alert_type TEXT NOT NULL,
  severity TEXT NOT NULL CHECK(severity IN ('low','medium','high','critical')),
  transaction_id TEXT REFERENCES transactions(id),
  details TEXT,
  status TEXT DEFAULT 'open' CHECK(status IN
('open','investigating','resolved','escalated','filed')),
  reviewed_by TEXT,
  reviewed_at TEXT,
  created_at TEXT DEFAULT (datetime('now'))
);

CREATE TABLE str_reports (
  id TEXT PRIMARY KEY,
  user_id TEXT NOT NULL REFERENCES users(id),
  alert_id TEXT REFERENCES aml_alerts(id),
  report_type TEXT NOT NULL,
  status TEXT DEFAULT 'draft' CHECK(status IN ('draft','submitted','acknowledged')),
  filed_at TEXT,
  reference_number TEXT,
  details TEXT,
  created_at TEXT DEFAULT (datetime('now'))
);

CREATE TABLE screening_results (
  id TEXT PRIMARY KEY,
  user_id TEXT NOT NULL REFERENCES users(id),
  screening_type TEXT NOT NULL CHECK(screening_type IN ('pep','sanctions','adverse_media')),
  provider TEXT,
  result TEXT NOT NULL CHECK(result IN ('clear','match','potential_match','error')),
  match_details TEXT,
  screened_at TEXT DEFAULT (datetime('now'))
);

CREATE TABLE consents (
  id TEXT PRIMARY KEY,
  user_id TEXT NOT NULL REFERENCES users(id),
  consent_type TEXT NOT NULL,
  granted INTEGER NOT NULL DEFAULT 1,
```

```
  granted_at TEXT DEFAULT (datetime('now')),
  withdrawn_at TEXT,
  ip_address TEXT
);

CREATE TABLE data_access_requests (
  id TEXT PRIMARY KEY,
  user_id TEXT NOT NULL REFERENCES users(id),
  request_type TEXT NOT NULL CHECK(request_type IN
('export','erasure','rectification','restriction')),
  status TEXT DEFAULT 'pending' CHECK(status IN
('pending','processing','completed','rejected')),
  requested_at TEXT DEFAULT (datetime('now')),
  completed_at TEXT,
  download_url TEXT,
  notes TEXT
);

CREATE TABLE complaints (
  id TEXT PRIMARY KEY,
  user_id TEXT NOT NULL REFERENCES users(id),
  category TEXT NOT NULL,
  subject TEXT NOT NULL,
  description TEXT NOT NULL,
  status TEXT DEFAULT 'received' CHECK(status IN
('received','investigating','resolved','escalated')),
  resolution TEXT,
  created_at TEXT DEFAULT (datetime('now')),
  resolved_at TEXT
);
```

Indexes

```
-- Core indexes
CREATE INDEX idx_transactions_user ON transactions(user_id);
CREATE INDEX idx_transactions_merchant ON transactions(merchant_id);
CREATE INDEX idx_recipients_user ON recipients(user_id);
CREATE INDEX idx_merchants_org ON merchants(org_number);
CREATE INDEX idx_bank_accounts_user ON bank_accounts(user_id);
```

```
CREATE INDEX idx_cards_user ON cards(user_id);
CREATE INDEX idx_sessions_user ON sessions(user_id);
CREATE INDEX idx_sessions_token ON sessions(token_hash);
CREATE INDEX idx_notifications_user ON notifications(user_id);
CREATE INDEX idx_spending_limits_user ON spending_limits(user_id);
CREATE INDEX idx_spending_limits_card ON spending_limits(card_id);

-- Compliance indexes
CREATE INDEX idx_audit_log_user ON audit_log(user_id);
CREATE INDEX idx_audit_log_timestamp ON audit_log(timestamp);
CREATE INDEX idx_audit_log_action ON audit_log(action);
CREATE INDEX idx_aml_alerts_user ON aml_alerts(user_id);
CREATE INDEX idx_aml_alerts_status ON aml_alerts(status);
CREATE INDEX idx_screening_user ON screening_results(user_id);
CREATE INDEX idx_consents_user ON consents(user_id);
CREATE INDEX idx_data_requests_user ON data_access_requests(user_id);
CREATE INDEX idx_complaints_user ON complaints(user_id);
CREATE INDEX idx_complaints_status ON complaints(status);
```

Note: Schema uses PostgreSQL 16 in all environments (development, CI, staging, production) via Drizzle ORM. The old dual-mode SQLite/PostgreSQL driver has been removed (ADR-014).

---

11. Common Error Format

All errors follow this format:

```
{
  "error": "error_code",
  "message": "Human readable message",
  "details": []
}
```

Code	Error	Description
400	bad_request	Malformed request body
401	unauthorized	Missing or expired JWT
402	insufficient_balance	Not enough NOK in bank account (PISP will fail)
403	kyc_required	KYC must be approved

Code	Error	Description
404	not_found	Resource not found
409	conflict	Duplicate (email, org number)
422	validation_error	Field validation failed
429	rate_limited	Too many requests
500	internal_error	Server error

---

12. Rate Limits

Endpoint Group	Limit	Window
Auth (login/register)	10/min	Per IP
Transactions (create)	30/min	Per user
Read endpoints	60/min	Per user
Exchange rates	120/min	Per IP

---

13. API Route File Structure (Next.js)

```
src/app/api/
├── auth/
│   ├── register/route.ts
│   ├── login/route.ts
│   ├── logout/route.ts
│   └── me/route.ts
├── bank-accounts/
│   └── route.ts          (GET linked accounts via AISP)
├── recipients/
│   ├── route.ts          (GET list, POST create)
│   └── [id]/route.ts     (DELETE)
├── transactions/
│   ├── route.ts          (GET list)
│   ├── [id]/route.ts     (GET detail)
│   ├── remittance/route.ts (POST)
│   └── qr-payment/route.ts (POST)
├── merchants/
```

```
│   ├── register/route.ts
│   ├── dashboard/route.ts
│   ├── transactions/route.ts
│   └── qr/route.ts
├── rates/
│   ├── route.ts          (GET all)
│   └── [currency]/route.ts (GET specific)
└── lib/
    ├── db.ts             (PostgreSQL/Drizzle connection)
    ├── auth.ts           (JWT verify/sign)
    └── middleware.ts     (auth middleware, rate limit)
```

Generated: 2026-02-09 by dev agent (Ollama) + John (orchestration) Status: Ready for implementation (Sprint 1)