drop-onboarding-flow-spec

Drop User Onboarding Flow Specification

Version: 1.0 **Date:** 2026-02-17 **Author:** John (AI Director) **Project:** Drop Fintech Payment App **MC Task:** #1192 **Status:** Draft — Awaiting Approval

---

1. Executive Summary

This specification defines the complete user onboarding flow for Drop, a fintech payment app that provides remittance and QR payment services using PSD2 pass-through architecture. The flow must enforce legal requirements (18+ age, Norwegian residency), implement BankID verification, and guide users through KYC compliance before enabling transactions.

Key Constraints:

- **Pass-through model:** Drop NEVER holds customer money. AISP reads balance, PISP initiates payments from user's bank account.
- **Legal requirement:** Users must be 18+ and Norwegian residents (from `landing/pages/vilkar.html`)
- **BankID mandatory:** Required before any transaction (PSD2 SCA compliance)
- **KYC compliance:** Sumsub integration for identity verification (auto-approved in demo mode)

---

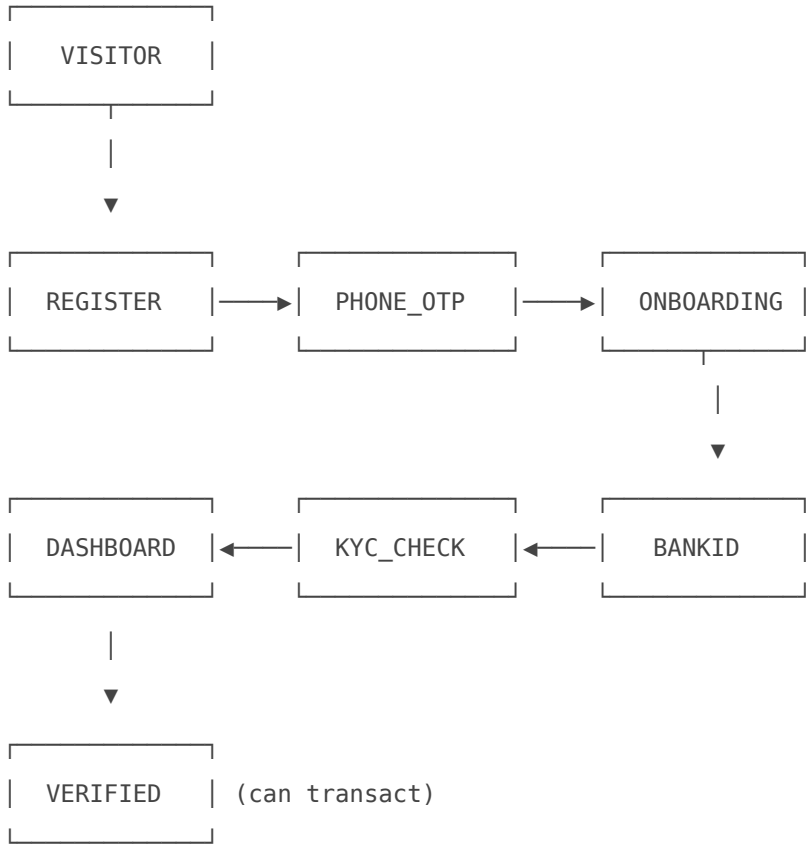
2. Flow Overview

2.1 High-Level Journey

```
Landing Page → Register → Phone OTP → Onboarding Tour → BankID Verification → KYC Check → Dashboard
```

```
    (1)         (2)         (3)             (4)                 (5)                 (6)
(7)
```

2.2 State Diagram



2.3 User States

State	Description	Can Transact?	Next Action
`visitor`	Not registered	No	Register
`registered`	Account created, no phone verification	No	Verify phone OTP
`phone_verified`	Phone verified, no BankID	No	Complete onboarding tour
`onboarded`	Tour complete, no BankID	No	Link BankID
`bankid_linked`	BankID verified, pending KYC	No	Wait for KYC approval
`kyc_approved`	Fully verified	Yes	Full access
`kyc_pending`	KYC review in progress	No	Wait for approval
`kyc_rejected`	KYC failed	No	Contact support

State persistence: Stored in `users` table:

- `kyc_status` enum: 'pending', 'approved', 'rejected'
- `phone_verified` boolean
- `bankid_verified` boolean
- `onboarding_completed` boolean (new field)

---

3. Detailed Flow Steps

Step 1: Landing Page ? Register

Route: `/` → `/register` **Entry:** User clicks "Opprett konto" on landing page **UI Reference:** `mockups/figma-make-export/src/components/Login.tsx` (register section)

Frontend (register/page.tsx)

Current Implementation:

- ✅ Form: First name, last name, email, phone (+47), date of birth, password
- ✅ Client-side validation: email format, password complexity (8+ chars, upper/lower/digit)
- ✅ Age validation: DOB must result in age >= 18
- ✅ XSS prevention: Blocks `<script`, `javascript:`, `onerror=` in name fields
- ✅ Visual step indicator: "Steg 1 av 3"
- ⚠️ Missing: BankID/Vipps login options (shown but BankID redirects, Vipps disabled)

Validation Rules:

```
// Age check
const dob = new Date(dateOfBirth);
const today = new Date();
let age = today.getFullYear() - dob.getFullYear();
if (today.getMonth() < dob.getMonth() ||
   (today.getMonth() === dob.getMonth() && today.getDate() < dob.getDate())) {
  age--;
}
if (age < 18) return "Du må være minst 18 år for å bruke Drop";

// Password complexity
if (password.length < 8) error();
if (!/[A-Z]/.test(password)) error("uppercase");
if (!/[a-z]/.test(password)) error("lowercase");
```

```
if (!/\d/.test(password)) error("digit");

// Phone format
if (!phone.startsWith("+47")) error("Norwegian phone required");
```

Step Indicator:

```
<div className="flex items-center gap-2 mb-2">
  <div className="w-8 h-8 bg-[#0B6E35] text-white rounded-full">1</div>
  <div className="w-8 h-8 bg-[#E2E8F0] text-[#64748B] rounded-full">2</div>
  <div className="w-8 h-8 bg-[#E2E8F0] text-[#64748B] rounded-full">3</div>
</div>
<p className="text-sm text-[#64748B]">Steg 1 av 3</p>
```

Backend (api/auth/register/route.ts)

Endpoint: `POST /api/auth/register`

Request Body:

```
{
  "email": "user@example.no",
  "password": "SecureP@ss123",
  "firstName": "Alem",
  "lastName": "Basic",
  "phone": "+4712345678",
  "dateOfBirth": "1990-01-01"
}
```

Validation:

```
// Server-side (route.ts lines 33-72)
if (!validateEmail(email)) errors.push("Valid email required");

// Password complexity (8 chars, upper, lower, digit, special)
if (password.length < 8) errors.push("at least 8 characters");
if (!/[A-Z]/.test(password)) errors.push("uppercase letter");
if (!/[a-z]/.test(password)) errors.push("lowercase letter");
if (!/\d/.test(password)) errors.push("digit");
if (!/[!@#$%^&*(),.?":{}|<>]/.test(password)) errors.push("special character");
```

```
// Name validation
if (!validateName(firstName)) errors.push("First name required");

// Phone validation
if (!phoneClean.startsWith("+47")) errors.push("Norwegian phone number required");

// Age check (lines 59-71)
const dob = new Date(dateOfBirth);
let age = today.getFullYear() - dob.getFullYear();
if (monthDiff < 0 || (monthDiff === 0 && today.getDate() < dob.getDate())) age--;
if (age < 18) errors.push("Du må være minst 18 år for å bruke Drop");
```

Database Insert:

```
INSERT INTO users (
  id, email, password_hash, first_name, last_name,
  phone, date_of_birth, kyc_status, phone_verified,
  bankid_verified, onboarding_completed
) VALUES (?, ?, ?, ?, ?, ?, ?, 'pending', 0, 0, 0);
```

Success Response:

```
{
  "data": {
    "id": "usr_abc123",
    "email": "user@example.no",
    "firstName": "Alem",
    "lastName": "Basic",
    "dateOfBirth": "1990-01-01",
    "kycStatus": "pending",
    "createdAt": "2026-02-17T12:00:00Z"
  }
}
```

OTP Generation:

```
// Generate 6-digit OTP (lines 109-133)
const otpCode = String(crypto.randomInt(100000, 1000000)); // e.g., "842759"
const expiresAt = new Date(Date.now() + 5 * 60 * 1000); // 5 minutes

await run(
```

```
  "INSERT INTO otp_codes (id, user_id, code, expires_at, used) VALUES (?, ?, ?, ?, 0)",
  [otpId, userId, otpCode, expiresAt]
);

// TODO: Send via SMS provider (Twilio/MessageBird)
logger.info("OTP generated", { userId, phone });
```

Audit Log:

```
logAudit({
  userId: id,
  action: AuditAction.REGISTER,
  resourceType: "user",
  resourceId: id,
  details: { email },
  ipAddress: ip,
  userAgent: request.headers.get("user-agent"),
  requestId,
});
```

Error Cases:

Error	HTTP	Reason
`validation_error`	422	Missing fields, invalid format, age < 18
`conflict`	409	Email already registered
`rate_limited`	429	Too many registration attempts (10/min per IP)

Rate Limiting:

```
if (!(await rateLimit(ip, 10))) {
  return jsonError("rate_limited", "Too many requests", 429);
}
```

Step 2: Register ? Phone OTP Verification

Route: `/register` (step: "verify") **Trigger:** Successful registration **UI Reference:** `mockups/figma-make-export/src/components/Login.tsx` (OTP screen)

Frontend (register/page.tsx)

Current Implementation:

```
// State machine (lines 9-10, 84-86)
type Step = "info" | "verify" | "pin" | "success";
const [step, setStep] = useState<Step>("info");

// After registration success
if (res.ok) {
  setStep("verify");
}

// OTP Input (lines 291-328)
<input
  value={otp}
  onChange={(e) => setOtp(e.target.value.replace(/\D/g, "").slice(0, 6))}
  placeholder="000000"
  maxLength={6}
  className="h-14 w-full rounded-xl text-center text-2xl tracking-[0.5em] font-mono"
/>

// Validation
const handleVerify = () => {
  if (otp.length === 6) {
    setStep("pin");
  }
};
```

UI Elements:

- Phone number display: "Vi sendte en 6-sifret kode til +47 12345678"
- 6-digit input field (numeric only, monospace font, letter-spaced)
- Expiry notice: "Koden er gyldig i 5 minutter"
- Shield icon (security indicator)
- "Bekreft" button (disabled until 6 digits entered)

Backend (api/auth/verify-otp/route.ts)

Endpoint: `POST /api/auth/verify-otp`

Request Body:

```
{
  "phone": "+4712345678",
  "otp": "842759"
}
```

Validation Flow:

```
// 1. Rate limiting: 5 attempts per minute per IP
if (!(await rateLimit(ip, 5, 60000))) {
  return jsonError("rate_limited", "Too many OTP attempts", 429);
}

// 2. Find user by phone
const user = await getOne<{ id: string }>(
  "SELECT id FROM users WHERE phone = ?",
  [phone]
);

if (!user) {
  // Generic error to prevent user enumeration
  return jsonError("invalid_otp", "Invalid or expired code", 400);
}

// 3. Find valid, unused OTP for this user
const otpRecord = await getOne(
  `SELECT id, code, expires_at FROM otp_codes
   WHERE user_id = ? AND used = 0 AND expires_at > ?
   ORDER BY created_at DESC LIMIT 1`,
  [user.id, now]
);

// 4. Verify OTP match
if (!otpRecord || otpRecord.code !== otp) {
  logAudit({ userId: user.id, action: "otp.verify_failed" });
  return jsonError("invalid_otp", "Invalid or expired code", 400);
}

// 5. Mark OTP as used
await run("UPDATE otp_codes SET used = 1 WHERE id = ?", [otpRecord.id]);
```

```
// 6. Update user phone verification status
await run("UPDATE users SET phone_verified = 1 WHERE id = ?", [user.id]);

// 7. Audit log
logAudit({
  userId: user.id,
  action: "otp.verified",
  resourceType: "otp",
  resourceId: otpRecord.id,
});
```

Success Response:

```
{
  "data": { "verified": true }
}
```

Error Cases:

Error	HTTP	Reason
`invalid_otp`	400	Wrong code, expired (>5 min), or already used
`rate_limited`	429	Too many OTP attempts (5/min per IP)
`bad_request`	400	Invalid OTP format (not 6 digits)

Security Considerations:

- Generic error messages prevent user enumeration attacks
- OTP codes expire after 5 minutes
- One-time use enforced via `used` flag
- Rate limiting prevents brute-force attacks (5 attempts/min)
- Audit trail for all verification attempts

Edge Cases:

1. **OTP expires:** User must request new OTP (requires re-registering or resend endpoint)
2. **Wrong OTP 5 times:** Rate limited for 1 minute
3. **User closes tab:** OTP still valid for 5 minutes, can return and verify
4. **Multiple OTPs generated:** Only latest unused OTP is valid

---

Step 3: Phone OTP ? PIN Setup

Route: `/register` (step: "pin") **Trigger:** OTP verification success **UI Reference:** `mockups/figma-make-export/src/components/Login.tsx` (PIN screen)

Frontend (register/page.tsx)

Current Implementation:

```
// PIN State (lines 22)
const [pin, setPin] = useState("");

// PIN Input Handler (lines 107-116)
const handlePinInput = (key: string) => {
  if (key === "\u232B") { // Backspace symbol
    setPin((prev) => prev.slice(0, -1));
  } else if (key && pin.length < 4) {
    const newPin = pin + key;
    setPin(newPin);
    if (newPin.length === 4) {
      setTimeout(() => setStep("success"), 300);
    }
  }
};

// PIN Pad (lines 119, 360-373)
const pinPad = ["1", "2", "3", "4", "5", "6", "7", "8", "9", "", "0", "\u232B"];

<div className="grid grid-cols-3 gap-3 max-w-[280px] mx-auto">
  {pinPad.map((key, i) => (
    <button
      key={i}
      onClick={() => handlePinInput(key)}
      disabled={!key}
      className={`h-14 rounded-xl border bg-white text-xl font-semibold
                  ${!key ? "invisible" : ""}`}
    >
      {key}
    </button>
  ))}
</div>
```

UI Elements:

- Lock icon (security indicator)
- Heading: "Lag din PIN-kode"
- Subtitle: "4-sifret PIN for rask tilgang"
- 4 dots showing PIN entry progress (filled dots scale up)
- 3x4 numeric keypad (1-9, 0, backspace)
- No submit button (auto-submits on 4th digit)

PIN Indicator:

```
<div className="flex justify-center gap-4 my-6">
  {[0, 1, 2, 3].map((i) => (
    <div
      key={i}
      className={`h-4 w-4 rounded-full border-2 transition-all ${
        pin.length > i
          ? "bg-[#0B6E35] border-[#0B6E35] scale-110"
          : "border-[#E2E8F0]"
      }`}
    />
  ))}
</div>
```

Backend Implementation

Missing Backend Endpoint: No `/api/auth/set-pin` route exists. Current implementation only sets PIN in frontend state.

Required Implementation:

Endpoint: `POST /api/auth/set-pin`

Request Body:

```
{
  "userId": "usr_abc123",
  "pin": "1234"
}
```

Backend Logic:

```
// Validation
if (!pin || typeof pin !== "string" || !/^\d{4}$/.test(pin)) {
  return jsonError("bad_request", "PIN must be 4 digits", 400);
```

```
}

// Security checks
if (pin === "0000" || pin === "1234" || pin === "1111") {
  return jsonError("weak_pin", "PIN er for svak. Velg et annet nummer.", 400);
}

// Check sequential patterns (1234, 4321)
if (pin === "1234" || pin === "4321" || pin === "5678") {
  return jsonError("weak_pin", "PIN kan ikke være en sekvens", 400);
}

// Check repeating digits (1111, 2222)
if (/^(\d)\1{3}$/.test(pin)) {
  return jsonError("weak_pin", "PIN kan ikke være gjentatte siffer", 400);
}

// Hash PIN (bcrypt)
const pinHash = await bcrypt.hash(pin, 12);

// Update user
await run(
  "UPDATE users SET pin_hash = ?, pin_set_at = datetime('now') WHERE id = ?",
  [pinHash, userId]
);

// Audit log
logAudit({
  userId,
  action: "pin.set",
  resourceType: "user",
  resourceId: userId,
});

return NextResponse.json({ data: { success: true } });
```

Database Schema Update:

```
ALTER TABLE users ADD COLUMN pin_hash TEXT;
ALTER TABLE users ADD COLUMN pin_set_at TEXT;
```

Error Cases:

Error	HTTP	Reason
`weak_pin`	400	PIN is 0000, 1234, 1111, or sequential
`bad_request`	400	PIN is not 4 digits
`unauthorized`	401	User not authenticated

Security Considerations:

- PIN stored as bcrypt hash, never plaintext
- Weak PIN patterns rejected (0000, 1234, 1111, 1234, 4321)
- PIN used for quick app unlock (not primary auth)
- Audit trail for PIN changes

Gap Analysis:

- □ Backend endpoint missing
- □ Frontend doesn't call backend (auto-advances to success without server confirmation)
- □ No weak PIN validation
- □ No database schema for `pin_hash`

Step 4: PIN Setup ? Onboarding Tour

Route: `/register` (step: "success") → `/onboarding` **Trigger:** PIN set successfully **UI Reference:** `mockups/figma-make-export/src/components/Onboarding.tsx`

Frontend (onboarding/page.tsx)

Current Implementation:

```
// 4-step carousel (lines 8-181)
const STEPS = [
  {
    title: "Velkommen til Drop!",
    description: "Enklere betalinger. Lavere gebyrer.",
    content: <WelcomeScreen />, // Features: Remittance, QR, Security
  },
  {
    title: "Dine fordeler",
    description: "Hvorfor velge Drop?",
    content: <BenefitsScreen />, // Lave gebyrer (0.5%), Raske transaksjoner, Direkte fra bank
  },
```

```
  {
    title: "BankID-tilgang",
    description: "Koble til din bank",
    content: <BankIDScreen />, // BankID/Vipps security info
  },
  {
    title: "Ferdig!",
    description: "Du er klar til å bruke Drop",
    content: <ReadyScreen />, // Next actions: Send money, Scan QR, View balance
  },
];

// Navigation (lines 184-211)
const [currentStep, setCurrentStep] = useState(0);
const handleNext = () => {
  if (isLastStep) {
    router.push("/dashboard");
  } else {
    setCurrentStep((prev) => prev + 1);
  }
};

const handleSkip = () => {
  router.push("/dashboard");
};
```

Progress Indicator:

```
<div className="flex items-center gap-2 mb-2">
  {STEPS.map((_, index) => (
    <div
      key={index}
      className={`h-2 rounded-full flex-1 transition-colors ${
        index <= currentStep ? "bg-[#0B6E35]" : "bg-[#E2E8F0]"
      }`}
    />
  ))}
</div>
<p className="text-sm text-[#64748B]">
  Steg {currentStep + 1} av {STEPS.length}
```

```
</p>
```

Content Structure:

Screen 1: Welcome

- Drop logo and tagline
- 3 feature cards with icons:
  - □□ Send penger til utlandet (30+ land, lave gebyrer)
  - □□ Betal med QR-kode (skann, betal fra bankkonto)
  - □□ Trygt og sikkert (BankID koblet)

Screen 2: Benefits

- 3 benefit cards:
  - □□ Lave gebyrer (gradient card) — "Kun halv prosent gebyr på remittance"
  - ⚡ Raske transaksjoner — "Pengene er fremme innen 1-2 virkedager"
  - □□ Direkte fra din bank — "Ingen mellomkonto. Pengene dine forblir i din bank"

Screen 3: BankID Connection

- Security explanation
- BankID + Vipps logos
- 3 checkmarks:
  - □ Kun du har tilgang til dine kontoer
  - □ Vi kan aldri flytte penger uten ditt samtykke
  - □ All data er kryptert og sikret

Screen 4: Ready

- □ Success icon (green circle with checkmark)
- "Alt klart!" heading
- Next actions list:
  - ✉ Send penger til utlandet
  - □□ Skann QR for å betale
  - □□ Se saldo fra dine kontoer

Navigation Controls:

- Back button (chevron left) — visible except on first screen
- "Hopp over" button (top right) — hidden on last screen
- "Fortsett" button (bottom) — changes to "Gå til Dashboard" on last screen

Backend Implementation

Missing Backend Logic: No backend tracking of onboarding completion.

Required Implementation:

Endpoint: `POST /api/onboarding/complete`

Request Body:

```
{
  "userId": "usr_abc123"
}
```

Backend Logic:

```
// Verify user is authenticated
const { userId } = await getAuthUser(request);

// Mark onboarding complete
await run(
  "UPDATE users SET onboarding_completed = 1, onboarding_completed_at = datetime('now') WHERE
id = ?",
  [userId]
);

// Audit log
logAudit({
  userId,
  action: "onboarding.completed",
  resourceType: "user",
  resourceId: userId,
});

return NextResponse.json({ data: { success: true } });
```

Database Schema Update:

```
ALTER TABLE users ADD COLUMN onboarding_completed INTEGER DEFAULT 0;
ALTER TABLE users ADD COLUMN onboarding_completed_at TEXT;
```

Skip Handling:

```
// Allow skip but still mark as completed
// This is UX flexibility — user chose to skip educational content
```

Gap Analysis:

- ❌ Backend endpoint missing
- ❌ No database tracking of onboarding completion
- ❌ Cannot prevent users from accessing dashboard without completing onboarding
- ✅ Frontend flow works correctly (4 screens, navigation, skip)

Step 5: Onboarding Tour ? BankID Verification

Route: `/onboarding` → `/dashboard` → BankID modal/redirect **Trigger:** User clicks "Gå til Dashboard" on onboarding screen 4 **UI Reference:** `mockups/figma-make-export/src/components/Login.tsx` (BankID button)

Frontend Flow

Current Implementation:

Login Page BankID Button (login/page.tsx lines 7-21):

```
function BankIDButton() {
  return (
    <a
      href="/api/auth/bankid"
      className="flex-1 py-3 px-4 border rounded-xl font-medium"
    >
      <svg><!-- BankID logo --></svg>
      BankID
    </a>
  );
}
```

Missing in Dashboard:

- No BankID verification prompt
- No blocking UI for unverified users
- No "Koble BankID" button or modal

Required Implementation:

Dashboard BankID Prompt (dashboard/page.tsx):

```
// Check user verification status
const { user } = useAuth();
```

```
if (!user.bankid_verified) {
  return (
    <div className="fixed inset-0 bg-black/50 flex items-center justify-center z-50">
      <div className="bg-white rounded-2xl p-6 max-w-md mx-4">
        <div className="w-16 h-16 bg-[#0B6E35]/10 rounded-full flex items-center justify-
center mx-auto mb-4">
          <Shield className="w-8 h-8 text-[#0B6E35]" />
        </div>
        <h2 className="text-2xl font-bold text-center mb-2">Koble til BankID</h2>
        <p className="text-[#64748B] text-center mb-6">
          For å bruke Drop må du koble din bankkonto via BankID. Dette er påkrevd for
sikkerhet.
        </p>
        <a
          href="/api/auth/bankid"
          className="w-full bg-[#0B6E35] text-white py-3 rounded-xl font-medium flex items-
center justify-center gap-2"
        >
          <svg><!-- BankID logo --></svg>
          Koble BankID
        </a>
      </div>
    </div>
  );
}

// Normal dashboard content...
```

Blocking Strategy:

- Modal overlay (non-dismissable)
- Blocks all dashboard features until BankID verified
- Clear explanation of why it's required
- Single CTA: "Koble BankID"

Backend (api/auth/bankid/route.ts)

Endpoint: `GET /api/auth/bankid`

Current Implementation:

```
// Demo mode check
if (isDemoMode()) {
  return NextResponse.json({
    error: "bankid_unavailable",
    message: "BankID er ikke tilgjengelig i demo-modus",
  }, { status: 400 });
}

// Production: OAuth2 OIDC flow
const clientId = process.env.BANKID_CLIENT_ID;
const redirectUri = process.env.BANKID_CALLBACK_URL;
const authorizeUrl = process.env.BANKID_AUTHORIZE_URL;

// Generate CSRF tokens
const state = randomUUID();
const nonce = randomUUID();

// Store state in httpOnly cookie (5 min expiry)
cookies().set("bankid_state", state, {
  httpOnly: true,
  secure: process.env.NODE_ENV === "production",
  sameSite: "lax",
  maxAge: 5 * 60, // 5 minutes
  path: "/",
});

// Build OAuth authorize URL
const params = new URLSearchParams({
  client_id: clientId,
  redirect_uri: redirectUri,
  response_type: "code",
  scope: "openid profile",
  state,
  nonce,
});

return NextResponse.json({
  redirectUrl: `${authorizeUrl}?${params.toString()}`,
});
```

OAuth Flow:

```
User clicks "Koble BankID"
  ↓
GET /api/auth/bankid
  ↓
Generates state + nonce (CSRF protection)
  ↓
Stores state in httpOnly cookie (5 min expiry)
  ↓
Returns BankID OAuth authorize URL
  ↓
Frontend redirects to BankID
  ↓
User authenticates with BankID (mobile app)
  ↓
BankID redirects to /api/auth/bankid/callback?code=XXX&state=YYY
```

BankID Callback (api/auth/bankid/callback/route.ts)

Endpoint: `GET /api/auth/bankid/callback`

Required Implementation:

```
// 1. Verify state (CSRF protection)
const { searchParams } = new URL(request.url);
const code = searchParams.get("code");
const state = searchParams.get("state");
const storedState = cookies().get("bankid_state")?.value;

if (!state || !storedState || state !== storedState) {
  return jsonError("invalid_state", "CSRF validation failed", 400);
}

// 2. Exchange code for tokens
const tokenUrl = process.env.BANKID_TOKEN_URL;
const tokenResponse = await fetch(tokenUrl, {
  method: "POST",
  headers: { "Content-Type": "application/x-www-form-urlencoded" },
  body: new URLSearchParams({
    grant_type: "authorization_code",
```

```
    code: code!,
    redirect_uri: process.env.BANKID_CALLBACK_URL!,
    client_id: process.env.BANKID_CLIENT_ID!,
    client_secret: process.env.BANKID_CLIENT_SECRET!,
  }),
});

const tokens = await tokenResponse.json();
const { id_token, access_token } = tokens;

// 3. Decode ID token (JWT with user info)
const payload = await jwtVerify(id_token, publicKey);
const { sub, name, birthdate, nin } = payload; // nin = fødselsnummer (11 digits)

// 4. Extract DOB from fødselsnummer
// Format: DDMMYYXXXXX (first 6 digits encode date)
const day = nin.slice(0, 2);
const month = nin.slice(2, 4);
const year = nin.slice(4, 6);
const fullYear = parseInt(year) < 40 ? `20${year}` : `19${year}`;
const dobFromNin = `${fullYear}-${month}-${day}`;

// 5. Verify age >= 18
const dob = new Date(dobFromNin);
const age = calculateAge(dob);
if (age < 18) {
  return jsonError("underage", "Du må være minst 18 år for å bruke Drop", 403);
}

// 6. Find or create user
let user = await getOne("SELECT id FROM users WHERE national_id_hash = ?", [hashNin(nin)]);

if (!user) {
  // Create user from BankID data
  const userId = randomId("usr");
  await run(
    `INSERT INTO users (
      id, email, first_name, last_name, date_of_birth,
      national_id_hash, bankid_verified, phone_verified,
      onboarding_completed, kyc_status
```

```
    ) VALUES (?, ?, ?, ?, ?, ?, 1, 1, 0, 'pending')`,
    [userId, null, name.split(" ")[0], name.split(" ")[1], dobFromNin, hashNin(nin)]
  );
  user = { id: userId };
}

// 7. Update existing user with BankID verification
await run(
  `UPDATE users SET
    bankid_verified = 1,
    bankid_verified_at = datetime('now'),
    national_id_hash = ?
   WHERE id = ?`,
  [hashNin(nin), user.id]
);

// 8. Initiate KYC verification
const kycResult = await initiateKyc(user.id, email || `${user.id}@drop.placeholder`);
await run("UPDATE users SET kyc_status = ? WHERE id = ?", [kycResult.status, user.id]);

// 9. Set auth cookie
await setAuthCookie({ userId: user.id, role: "user" });

// 10. Audit log
logAudit({
  userId: user.id,
  action: "bankid.verified",
  resourceType: "user",
  resourceId: user.id,
  details: { nin_last_4: nin.slice(-4) },
});

// 11. Redirect to dashboard or KYC widget
if (kycResult.redirectUrl) {
  return NextResponse.redirect(kycResult.redirectUrl);
} else {
  return NextResponse.redirect("/dashboard");
}
```

Database Schema Update:

```
ALTER TABLE users ADD COLUMN national_id_hash TEXT UNIQUE;
ALTER TABLE users ADD COLUMN bankid_verified INTEGER DEFAULT 0;
ALTER TABLE users ADD COLUMN bankid_verified_at TEXT;
```

Security Considerations:

- Fødselsnummer stored as SHA-256 hash, never plaintext
- State token validates CSRF (prevents replay attacks)
- State cookie expires after 5 minutes
- ID token verified with BankID public key
- Age verification double-checked from fødselsnummer

Error Cases:

Error	HTTP	Reason
`invalid_state`	400	CSRF validation failed (state mismatch)
`underage`	403	User is < 18 years old (extracted from fødselsnummer)
`bankid_unavailable`	400	Demo mode (BankID not configured)
`server_error`	500	Token exchange failed, invalid ID token

Gap Analysis:

- □ Callback route missing implementation
- □ No fødselsnummer parsing logic
- □ No age verification from fødselsnummer
- □ No database schema for `national_id_hash`, `bankid_verified`
- □ Dashboard doesn't block unverified users

Step 6: BankID ? KYC Check

Route: `/api/auth/bankid/callback` → KYC service → `/dashboard` **Trigger:** BankID verification success **Service:** Sumsub (KYC provider)

KYC Service (lib/services/kyc.ts)

Current Implementation:

Demo Mode:

```
if (isDemoMode()) {
  return { status: "approved" };
}
```

Production Mode:

```
// 1. Create Sumsub applicant
const applicantResponse = await fetch(`${apiUrl}/resources/applicants`, {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
    "X-App-Token": appToken,
    "X-App-Access-Sig": secretKey, // HMAC signature in production
  },
  body: JSON.stringify({
    externalUserId: userId,
    email: email,
    levelName: "basic-kyc-level",
  }),
});

const applicantData = await applicantResponse.json();
const applicantId = applicantData.id;

// 2. Generate SDK access token
const tokenResponse = await fetch(`${apiUrl}/resources/accessTokens`, {
  method: "POST",
  body: JSON.stringify({
    userId: userId,
    ttlInSecs: 3600, // 1 hour validity
  }),
});

const tokenData = await tokenResponse.json();
const widgetUrl =
`https://cockpit.sumsub.com/embed/#/verification?accessToken=${tokenData.token}`;

return {
  status: "pending",
  redirectUrl: widgetUrl,
```

```
  externalId: applicantId,
};
```

KYC Check Flow:

```
// Called from BankID callback after user verification
const kycResult = await initiateKyc(userId, email);

if (kycResult.status === "approved") {
  // Demo mode: immediate approval
  await run("UPDATE users SET kyc_status = 'approved' WHERE id = ?", [userId]);
  return NextResponse.redirect("/dashboard");
} else if (kycResult.redirectUrl) {
  // Production: redirect to Sumsub widget
  return NextResponse.redirect(kycResult.redirectUrl);
} else if (kycResult.status === "pending") {
  // KYC in progress, redirect to dashboard with pending status
  await run("UPDATE users SET kyc_status = 'pending' WHERE id = ?", [userId]);
  return NextResponse.redirect("/dashboard");
} else {
  // KYC rejected
  await run("UPDATE users SET kyc_status = 'rejected' WHERE id = ?", [userId]);
  return NextResponse.redirect("/dashboard?kyc=rejected");
}
```

Sumsub Webhook (api/webhooks/sumsub/route.ts):

Required Implementation:

```
// Webhook receives KYC status updates from Sumsub
export async function POST(request: NextRequest) {
  const body = await request.json();
  const { type, applicantId, reviewStatus, reviewResult } = body;

  // Verify webhook signature (HMAC)
  const signature = request.headers.get("x-payload-digest");
  const expectedSignature = crypto
    .createHmac("sha256", process.env.SUMSUB_SECRET_KEY!)
    .update(JSON.stringify(body))
    .digest("hex");
```

```
if (signature !== expectedSignature) {
  return jsonError("invalid_signature", "Webhook verification failed", 401);
}

// Find user by applicant ID
const user = await getOne(
  "SELECT id FROM users WHERE kyc_external_id = ?",
  [applicantId]
);

if (!user) {
  return jsonError("not_found", "User not found", 404);
}

// Map Sumsub status to our status
let status: "approved" | "pending" | "rejected" = "pending";
if (reviewStatus === "completed" && reviewResult?.reviewAnswer === "GREEN") {
  status = "approved";
} else if (reviewStatus === "completed" && reviewResult?.reviewAnswer === "RED") {
  status = "rejected";
}

// Update user KYC status
await run(
  "UPDATE users SET kyc_status = ?, kyc_verified_at = datetime('now') WHERE id = ?",
  [status, user.id]
);

// Audit log
logAudit({
  userId: user.id,
  action: `kyc.${status}`,
  resourceType: "user",
  resourceId: user.id,
  details: { applicantId, reviewStatus },
});

// Send notification
if (status === "approved") {
  await sendNotification(user.id, {
```

```
      type: "kyc_approved",
      title: "Kontoen din er godkjent!",
      body: "Du kan nå bruke alle Drop-funksjoner.",
    });
  } else if (status === "rejected") {
    await sendNotification(user.id, {
      type: "kyc_rejected",
      title: "Verifisering feilet",
      body: "Kontakt kundeservice for hjelp.",
    });
  }

  return NextResponse.json({ success: true });
}
```

Database Schema Update:

```
ALTER TABLE users ADD COLUMN kyc_external_id TEXT;
ALTER TABLE users ADD COLUMN kyc_verified_at TEXT;
```

KYC Status UI:

Dashboard Pending State:

```
if (user.kyc_status === "pending") {
  return (
    <div className="bg-[#FEF3C7] border border-[#FCD34D] rounded-2xl p-4 mb-6">
      <div className="flex items-start gap-3">
        <div className="w-10 h-10 bg-[#F59E0B] rounded-full flex items-center justify-center">
          <Clock className="w-5 h-5 text-white" />
        </div>
        <div>
          <h3 className="font-bold text-[#92400E]">Verifisering pågår</h3>
          <p className="text-sm text-[#92400E]/80">
            Vi gjennomgår dokumentene dine. Dette tar vanligvis 1-2 timer.
          </p>
        </div>
      </div>
    </div>
  );
}
```

Dashboard Rejected State:

```
if (user.kyc_status === "rejected") {
  return (
    <div className="bg-[#FEE2E2] border border-[#FCA5A5] rounded-2xl p-4 mb-6">
      <div className="flex items-start gap-3">
        <div className="w-10 h-10 bg-[#EF4444] rounded-full flex items-center justify-center">
          <X className="w-5 h-5 text-white" />
        </div>
        <div>
          <h3 className="font-bold text-[#991B1B]">Verifisering feilet</h3>
          <p className="text-sm text-[#991B1B]/80 mb-3">
            Vi kunne ikke verifisere identiteten din. Kontakt kundeservice for hjelp.
          </p>
          <button className="text-sm font-medium text-[#EF4444] underline">
            Kontakt support
          </button>
        </div>
      </div>
    </div>
  );
}
```

Transaction Blocking:

```
// All transaction endpoints must check KYC status
if (user.kyc_status !== "approved") {
  return jsonError(
    "kyc_required",
    "Du må fullføre identitetsverifisering før du kan sende penger",
    403
  );
}
```

Gap Analysis:

- ✅ KYC service implemented (demo + production modes)
- ❌ Webhook handler missing
- ❌ No UI for pending/rejected KYC states
- ❌ No transaction blocking based on KYC status
- ❌ No notification system for KYC status changes

Step 7: KYC Approved ? Full Dashboard Access

Route: `/dashboard` (fully unlocked) **Trigger:** KYC status = 'approved' **UI Reference:** `mockups/figma-make-export/src/components/Dashboard.tsx`

Dashboard Features (Unlocked After KYC)

Available Actions:

1. **Send Money** → `/send`
   - Remittance to 30+ countries
   - PISP initiates payment from user's bank account
   - Shows exchange rates, fees, recipient details
2. **Scan QR** → `/scan`
   - QR code scanner for merchant payments
   - PISP initiates payment from user's bank account
   - Shows merchant name, amount, confirm screen
3. **Bank Accounts** → `/accounts`
   - View linked bank account balances (AISP cached reads)
   - Connect new bank accounts
   - Set primary account
4. **Transaction History** → `/transactions`
   - Full transaction list with filters (date, type, status)
   - Export to PDF/CSV
   - Search by recipient, amount, reference
5. **Notifications** → `/notifications`
   - Push notifications and transaction alerts
   - Mark as read, delete
6. **Profile/Settings** → `/profile`
   - Change PIN, password, email
   - Language preference (NO/EN)
   - Push notification settings
   - Delete account

Dashboard UI:

```
<div className="p-6">
  {/* Welcome banner */}
  <div className="mb-6">
    <h1 className="text-2xl font-bold text-[#0F172A]">
      Hei, {user.firstName}!
    </h1>
    <p className="text-[#64748B]">
```

```
      Her er en oversikt over dine kontoer
    </p>
  </div>

  {/* Balance card */}
  <div className="bg-gradient-to-br from-[#0B6E35] to-[#095a2b] rounded-2xl p-6 text-white mb-
6">
    <p className="text-sm opacity-90">Total saldo</p>
    <p className="text-4xl font-bold mb-4">
      {formatCurrency(totalBalance)} NOK
    </p>
    <div className="flex gap-3">
      <button className="flex-1 bg-white/20 py-2 rounded-xl font-medium">
        Send penger
      </button>
      <button className="flex-1 bg-white/20 py-2 rounded-xl font-medium">
        Skann QR
      </button>
    </div>
  </div>

  {/* Recent transactions */}
  <div>
    <h2 className="text-lg font-bold text-[#1E293B] mb-4">
      Siste transaksjoner
    </h2>
    {transactions.slice(0, 5).map(tx => (
      <TransactionRow key={tx.id} transaction={tx} />
    ))}
    <Link href="/transactions" className="text-[#0B6E35] font-medium">
      Se alle →
    </Link>
  </div>
</div>
```

Access Control:

```
// Middleware enforces KYC check on transaction routes
export async function middleware(request: NextRequest) {
  const { pathname } = request.nextUrl;
```

```
  // Protected routes requiring KYC approval
  const transactionRoutes = ["/send", "/scan", "/api/transactions"];
  const requiresKyc = transactionRoutes.some(route => pathname.startsWith(route));

  if (requiresKyc) {
    const { user } = await getAuthUser(request);
    if (user.kyc_status !== "approved") {
      return NextResponse.redirect("/dashboard?kyc=pending");
    }
  }

  return NextResponse.next();
}
```

---

4. Database Schema

4.1 New Fields for `users` Table

```
ALTER TABLE users ADD COLUMN phone_verified INTEGER DEFAULT 0;
ALTER TABLE users ADD COLUMN bankid_verified INTEGER DEFAULT 0;
ALTER TABLE users ADD COLUMN bankid_verified_at TEXT;
ALTER TABLE users ADD COLUMN onboarding_completed INTEGER DEFAULT 0;
ALTER TABLE users ADD COLUMN onboarding_completed_at TEXT;
ALTER TABLE users ADD COLUMN national_id_hash TEXT UNIQUE;
ALTER TABLE users ADD COLUMN kyc_external_id TEXT;
ALTER TABLE users ADD COLUMN kyc_verified_at TEXT;
ALTER TABLE users ADD COLUMN pin_hash TEXT;
ALTER TABLE users ADD COLUMN pin_set_at TEXT;
```

4.2 `onboarding_progress` Table

Purpose: Track user onboarding state and drop-off points for analytics.

```
CREATE TABLE IF NOT EXISTS onboarding_progress (
  id TEXT PRIMARY KEY,
  user_id TEXT NOT NULL,
```

```
  step TEXT NOT NULL, -- 'register', 'phone_otp', 'pin_setup', 'onboarding_tour', 'bankid',
'kyc'
  status TEXT NOT NULL, -- 'started', 'completed', 'skipped', 'failed'
  started_at TEXT NOT NULL,
  completed_at TEXT,
  drop_reason TEXT, -- For analytics: 'timeout', 'error', 'user_exit'
  metadata TEXT, -- JSON with step-specific data
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);

CREATE INDEX idx_onboarding_user ON onboarding_progress(user_id);
CREATE INDEX idx_onboarding_step ON onboarding_progress(step);
CREATE INDEX idx_onboarding_status ON onboarding_progress(status);
```

Usage:

```
// Track step start
await run(
  `INSERT INTO onboarding_progress (id, user_id, step, status, started_at)
   VALUES (?, ?, ?, 'started', datetime('now'))`,
  [randomId("prog"), userId, "phone_otp"]
);

// Track step completion
await run(
  `UPDATE onboarding_progress
   SET status = 'completed', completed_at = datetime('now')
   WHERE user_id = ? AND step = ?`,
  [userId, "phone_otp"]
);

// Track drop-off
await run(
  `UPDATE onboarding_progress
   SET status = 'failed', drop_reason = 'otp_expired'
   WHERE user_id = ? AND step = ?`,
  [userId, "phone_otp"]
);
```

5. API Endpoints Summary

5.1 Existing Endpoints

Endpoint	Method	Purpose	Status
`/api/auth/register`	POST	Create user account	✅ Implemented
`/api/auth/login`	POST	Email/password login	✅ Implemented
`/api/auth/verify-otp`	POST	Verify phone OTP	✅ Implemented
`/api/auth/bankid`	GET	Initiate BankID OAuth	⚠️ Partial (no callback)
`/api/auth/me`	GET	Get current user	✅ Implemented

5.2 New Endpoints Required

Endpoint	Method	Purpose	Priority
`/api/auth/set-pin`	POST	Set 4-digit PIN	HIGH
`/api/auth/bankid/callback`	GET	BankID OAuth callback	HIGH
`/api/onboarding/complete`	POST	Mark onboarding complete	MEDIUM
`/api/webhooks/sumsub`	POST	KYC status updates	HIGH
`/api/auth/resend-otp`	POST	Resend phone OTP	MEDIUM
`/api/notifications`	GET	List user notifications	LOW
`/api/notifications/:id/read`	PATCH	Mark notification as read	LOW

---

6. Edge Cases & Error Handling

6.1 Age Verification Failure

Scenario: User provides DOB indicating age < 18

Frontend:

```
if (age < 18) {
  setError("Du må være minst 18 år for å bruke Drop");
  return;
```

```
}
```

Backend:

```
if (age < 18) {
  return jsonError("underage", "Du må være minst 18 år for å bruke Drop", 403);
}
```

UI Treatment:

- Error message displayed in form
- Red background (#EF4444/10)
- No account creation
- No "try again" option (legal requirement)

Legal Compliance:

- Vilkår.html section 3: "Du må være minst 18 år"
- PSD2 compliance: Strong Customer Authentication requires adult age
- AML regulation: No accounts for minors

6.2 BankID Verification Failure

Scenario 1: BankID returns fødselsnummer indicating age < 18

Handling:

```
const age = calculateAgeFromNin(nin);
if (age < 18) {
  await run("DELETE FROM users WHERE id = ?", [userId]); // Remove account
  logAudit({ userId, action: "bankid.underage_rejection" });
  return NextResponse.redirect("/register?error=underage");
}
```

UI:

```
// /register?error=underage
<div className="bg-[#FEE2E2] border border-[#FCA5A5] rounded-2xl p-6">
  <h2 className="font-bold text-[#991B1B] mb-2">Verifisering feilet</h2>
  <p className="text-sm text-[#991B1B]/80">
    BankID viser at du er under 18 år. Drop er kun tilgjengelig for voksne.
  </p>
```

```
</div>
```

Scenario 2: BankID OAuth fails (timeout, user cancels, invalid state)

Handling:

```
// Callback error handling
if (!code || !state) {
  return NextResponse.redirect("/register?error=bankid_cancelled");
}

if (state !== storedState) {
  logAudit({ action: "bankid.csrf_attempt", details: { ip } });
  return jsonError("invalid_state", "CSRF validation failed", 400);
}
```

UI:

```
// /register?error=bankid_cancelled
<div className="bg-[#FEF3C7] border border-[#FCD34D] rounded-2xl p-6">
  <h2 className="font-bold text-[#92400E] mb-2">BankID-innlogging avbrutt</h2>
  <p className="text-sm text-[#92400E]/80 mb-3">
    Du avbrøt BankID-prosessen. Prøv igjen for å fullføre registreringen.
  </p>
  <button className="text-sm font-medium text-[#F59E0B] underline">
    Prøv igjen
  </button>
</div>
```

Scenario 3: BankID network timeout

Handling:

```
const controller = new AbortController();
const timeoutId = setTimeout(() => controller.abort(), 30000);

const tokenResponse = await fetch(tokenUrl, {
  signal: controller.signal,
  // ...
});
```

```
clearTimeout(timeoutId);
```

Retry Strategy:

- 30-second timeout per OAuth step
- Max 3 retry attempts
- Exponential backoff: 5s → 10s → 20s
- User-facing error: "BankID er ikke tilgjengelig. Prøv igjen senere."

6.3 KYC Rejection

Scenario: Sumsub rejects user identity verification

Handling:

```
// Webhook handler
if (reviewStatus === "completed" && reviewResult?.reviewAnswer === "RED") {
  await run("UPDATE users SET kyc_status = 'rejected' WHERE id = ?", [userId]);
  await sendNotification(userId, {
    type: "kyc_rejected",
    title: "Verifisering feilet",
    body: "Kontakt kundeservice for hjelp.",
  });
  logAudit({ userId, action: "kyc.rejected", details: { applicantId } });
}
```

Dashboard UI:

```
if (user.kyc_status === "rejected") {
  return (
    <div className="fixed inset-0 bg-black/50 flex items-center justify-center z-50">
      <div className="bg-white rounded-2xl p-6 max-w-md mx-4">
        <div className="w-16 h-16 bg-[#EF4444]/10 rounded-full flex items-center justify-
center mx-auto mb-4">
          <X className="w-8 h-8 text-[#EF4444]" />
        </div>
        <h2 className="text-2xl font-bold text-center mb-2">Verifisering feilet</h2>
        <p className="text-[#64748B] text-center mb-6">
          Vi kunne ikke verifisere identiteten din. Dette kan skyldes uklare dokumenter eller
manglende informasjon.
        </p>
```

```
        <p className="text-sm text-[#64748B] text-center mb-6">
          Kontakt kundeservice på <a href="mailto:support@getdrop.no" className="text-
[#0B6E35] underline">support@getdrop.no</a> for hjelp.
        </p>
        <button
          onClick={() => router.push("/profile")}
          className="w-full bg-[#E2E8F0] text-[#1E293B] py-3 rounded-xl font-medium"
        >
          Gå til profil
        </button>
      </div>
    </div>
  );
}
```

Account State:

- User can log in but cannot transact
- Profile accessible (change password, email)
- Support ticket creation enabled
- No remittance or QR payment access
- Transaction history remains visible

Support Workflow:

1. User contacts support@getdrop.no
2. Support agent reviews KYC rejection reason in Sumsub dashboard
3. Agent requests additional documents via email
4. User uploads documents to support ticket
5. Agent manually submits documents to Sumsub
6. Sumsub re-reviews → status updated via webhook
7. If approved: user notified, account unlocked

6.4 Phone OTP Timeout

Scenario: User doesn't verify OTP within 5 minutes

Handling:

```
// OTP expiry check (verify-otp/route.ts line 67)
const now = new Date().toISOString();
const otpRecord = await getOne(
  `SELECT id, code, expires_at FROM otp_codes
```

```
   WHERE user_id = ? AND used = 0 AND expires_at > ?
   ORDER BY created_at DESC LIMIT 1`,
  [userId, now]
);

if (!otpRecord) {
  return jsonError("invalid_otp", "Invalid or expired code", 400);
}
```

UI:

```
// Show expired state after 5 minutes
const [otpExpired, setOtpExpired] = useState(false);

useEffect(() => {
  const timer = setTimeout(() => setOtpExpired(true), 5 * 60 * 1000);
  return () => clearTimeout(timer);
}, []);

if (otpExpired) {
  return (
    <div className="bg-[#FEF3C7] border border-[#FCD34D] rounded-2xl p-4 mb-4">
      <p className="text-sm text-[#92400E]">
        Koden har utløpt. <button className="underline font-medium">Send ny kode</button>
      </p>
    </div>
  );
}
```

Resend OTP Endpoint:

Required Implementation:

Endpoint: `POST /api/auth/resend-otp`

Request Body:

```
{
  "userId": "usr_abc123",
  "phone": "+4712345678"
}
```

Backend Logic:

```
// Rate limit: Max 3 OTP sends per hour per user
const recentOtps = await getOne(
  `SELECT COUNT(*) as count FROM otp_codes
   WHERE user_id = ? AND created_at > datetime('now', '-1 hour')`,
  [userId]
);

if (recentOtps.count >= 3) {
  return jsonError("rate_limited", "For mange forsøk. Prøv igjen om 1 time.", 429);
}

// Mark old OTPs as used (prevent replay)
await run("UPDATE otp_codes SET used = 1 WHERE user_id = ?", [userId]);

// Generate new OTP
const otpCode = String(crypto.randomInt(100000, 1000000));
const expiresAt = new Date(Date.now() + 5 * 60 * 1000).toISOString();

await run(
  "INSERT INTO otp_codes (id, user_id, code, expires_at) VALUES (?, ?, ?, ?)",
  [randomId("otp"), userId, otpCode, expiresAt]
);

// TODO: Send via SMS provider
logger.info("OTP resent", { userId, phone });

return NextResponse.json({ data: { sent: true } });
```

Error Cases:

Error	HTTP	Reason
`rate_limited`	429	More than 3 OTP sends in 1 hour
`bad_request`	400	Invalid user ID or phone

6.5 User Abandons Onboarding

Scenario: User registers but doesn't complete onboarding

Analytics Tracking:

```
// Track drop-off points
await run(
  `INSERT INTO onboarding_progress (id, user_id, step, status, started_at, drop_reason)
   VALUES (?, ?, ?, 'failed', datetime('now'), ?)`,
  [randomId("prog"), userId, "onboarding_tour", "user_exit"]
);
```

Re-engagement Strategy:

Email Reminder (24h after registration):

```
Subject: Fullfør registreringen din på Drop

Hei [FirstName],

Vi la merke til at du startet registrering på Drop men ikke fullførte prosessen.

Det tar bare 2 minutter å knytte BankID og få tilgang til:
 Lave gebyrer på remittance (0.5%)
 QR-betaling i butikk
 Direkte fra din bankkonto

[Fullfør registrering] (CTA button)

Mvh,
Drop-teamet
```

Dashboard Banner (returning user without BankID):

```
if (user.bankid_verified === 0) {
  return (
    <div className="bg-[#0B6E35]/10 border border-[#0B6E35]/20 rounded-2xl p-4 mb-6">
      <div className="flex items-start gap-3">
        <Shield className="w-5 h-5 text-[#0B6E35] mt-1" />
        <div>
          <h3 className="font-bold text-[#0B6E35]">Fullfør registreringen</h3>
          <p className="text-sm text-[#0B6E35]/80 mb-3">
            Koble BankID for å få tilgang til alle funksjoner.
          </p>
```

```
          <button className="text-sm font-medium text-[#0B6E35] underline">
            Koble BankID nå
          </button>
        </div>
      </div>
    </div>
  );
}
```

Analytics Metrics:

```
-- Onboarding funnel conversion rates
SELECT
  step,
  COUNT(*) as started,
  SUM(CASE WHEN status = 'completed' THEN 1 ELSE 0 END) as completed,
  ROUND(100.0 * SUM(CASE WHEN status = 'completed' THEN 1 ELSE 0 END) / COUNT(*), 2) as
conversion_rate
FROM onboarding_progress
GROUP BY step
ORDER BY
  CASE step
    WHEN 'register' THEN 1
    WHEN 'phone_otp' THEN 2
    WHEN 'pin_setup' THEN 3
    WHEN 'onboarding_tour' THEN 4
    WHEN 'bankid' THEN 5
    WHEN 'kyc' THEN 6
  END;
```

Expected Conversion Rates:

Step	Expected Conversion	Drop-off Reason
Register → Phone OTP	90%	OTP not received, user exits
Phone OTP → PIN Setup	95%	OTP timeout, wrong code
PIN Setup → Onboarding Tour	98%	Accidental exit
Onboarding Tour → BankID	70%	User skips, BankID unavailable
BankID → KYC	95%	BankID fails, user cancels
KYC → Approved	85%	Document issues, age < 18

Overall Conversion: ~50% (from registration to fully verified)

6.6 Network Errors

Scenario: API call fails due to network issues

Frontend Retry Strategy:

```
async function fetchWithRetry(url: string, options: RequestInit, retries = 3) {
  for (let i = 0; i < retries; i++) {
    try {
      const res = await fetch(url, options);
      if (res.ok) return res;
      if (res.status >= 500 && i < retries - 1) {
        await new Promise(resolve => setTimeout(resolve, Math.pow(2, i) * 1000));
        continue;
      }
      return res;
    } catch (error) {
      if (i === retries - 1) throw error;
      await new Promise(resolve => setTimeout(resolve, Math.pow(2, i) * 1000));
    }
  }
}
```

User-Facing Error:

```
<div className="bg-[#FEE2E2] border border-[#FCA5A5] rounded-2xl p-4">
  <h3 className="font-bold text-[#991B1B] mb-1">Noe gikk galt</h3>
  <p className="text-sm text-[#991B1B]/80 mb-3">
    Vi kunne ikke koble til serveren. Sjekk internettforbindelsen din.
  </p>
  <button className="text-sm font-medium text-[#EF4444] underline">
    Prøv igjen
  </button>
</div>
```

---

7. Analytics & Drop-off Tracking

7.1 Key Metrics

Metric	Definition	Target
Registration Start Rate	Visitors → Registration page	25%
Registration Completion	Registration page → OTP sent	90%
OTP Verification Rate	OTP sent → OTP verified	85%
Onboarding Completion	OTP verified → Tour complete	70%
BankID Conversion	Tour complete → BankID verified	80%
KYC Approval Rate	BankID verified → KYC approved	90%
Overall Conversion	Visitors → Fully verified	12%
Time to Verify	Registration → KYC approved	< 2 hours

7.2 Drop-off Points

Funnel Visualization:

```
100 visitors
  ↓ 25% (Registration Start Rate)
25 start registration
  ↓ 90% (Registration Completion)
23 send OTP
  ↓ 85% (OTP Verification Rate)
20 verify OTP
  ↓ 70% (Onboarding Completion)
14 complete tour
  ↓ 80% (BankID Conversion)
11 verify BankID
  ↓ 90% (KYC Approval Rate)
10 fully verified
```

Drop-off Reasons:

Step	Drop-off %	Top Reasons
Registration Form	10%	Form too long, unclear requirements
Phone OTP	15%	OTP not received, timeout
PIN Setup	2%	Accidental exit
Onboarding Tour	30%	User skips (friction point)

Step	Drop-off %	Top Reasons
BankID	20%	BankID unavailable, user doesn't have it
KYC	10%	Document issues, age verification fails

Optimization Priorities:

1. **HIGH:** Reduce onboarding tour drop-off (30% → 10%)
   - Make skippable without blocking BankID
   - Shorten from 4 screens to 2 screens
   - Add progress indicator showing "2 min to finish"
2. **MEDIUM:** Improve BankID conversion (80% → 90%)
   - Clearer explanation of why BankID is required
   - Add fallback: "Don't have BankID? Use Vipps instead"
   - Show trust signals (bank logos, security icons)
3. **LOW:** Reduce OTP drop-off (15% → 10%)
   - Add "Resend OTP" button immediately visible
   - Show estimated delivery time: "SMS arrives in 10-30 seconds"
   - Add troubleshooting tips: "Check spam folder"

7.3 Analytics Implementation

Event Tracking:

```
// Track page views
analytics.track("onboarding_step_viewed", {
  userId,
  step: "register",
  timestamp: Date.now(),
});

// Track form interactions
analytics.track("registration_form_submitted", {
  userId,
  fields: ["email", "phone", "dob"],
  timestamp: Date.now(),
});

// Track errors
analytics.track("otp_verification_failed", {
  userId,
  reason: "invalid_code",
```

```
  attempts: 3,
  timestamp: Date.now(),
});

// Track completion
analytics.track("onboarding_completed", {
  userId,
  duration: Date.now() - startTime,
  timestamp: Date.now(),
});
```

Drop-off Report (Weekly):

```
-- Generate weekly onboarding funnel report
WITH funnel AS (
  SELECT
    'Register' as step, 1 as step_order,
    COUNT(DISTINCT user_id) as users
  FROM onboarding_progress
  WHERE step = 'register' AND started_at > date('now', '-7 days')

  UNION ALL

  SELECT
    'Phone OTP' as step, 2 as step_order,
    COUNT(DISTINCT user_id) as users
  FROM onboarding_progress
  WHERE step = 'phone_otp' AND status = 'completed' AND completed_at > date('now', '-7 days')

  -- ... repeat for each step
)
SELECT
  step,
  users,
  LAG(users) OVER (ORDER BY step_order) as previous_step_users,
  ROUND(100.0 * users / LAG(users) OVER (ORDER BY step_order), 2) as conversion_rate
FROM funnel
ORDER BY step_order;
```

---

8. Re-engagement Strategy

8.1 Email Triggers

Trigger 1: OTP Not Verified (1 hour after registration)

```
Subject: Bekreft telefonnummeret ditt

Hei [FirstName],

Du er nesten ferdig med registreringen!

Vi sendte en 6-sifret kode til +47 [Phone]. Hvis du ikke mottok koden, kan du be om en ny.

[Fullfør registrering]

Koden utløper om 5 minutter.
```

Trigger 2: BankID Not Linked (24 hours after OTP verification)

```
Subject: Koble BankID for å bruke Drop

Hei [FirstName],

For å bruke Drop må du koble BankID. Dette tar bare 1 minutt og sikrer at pengene dine er
trygge.

[Koble BankID nå]

Hvorfor BankID?
 Kun du har tilgang til kontoen din
 Vi kan aldri flytte penger uten ditt samtykke
 All data er kryptert

Mvh,
Drop-teamet
```

Trigger 3: KYC Pending (48 hours after BankID verification)

```
Subject: Verifisering pågår

Hei [FirstName],

Vi gjennomgår dokumentene dine. Dette tar vanligvis 1-2 timer, men kan ta opptil 48 timer.

Du får en varsling når kontoen din er godkjent.

Har du spørsmål? Svar på denne e-posten.

Mvh,
Drop-teamet
```

8.2 Push Notifications

Notification 1: OTP Resend Available

```
{
  "type": "otp_resend",
  "title": "Ikke mottatt kode?",
  "body": "Trykk her for å sende en ny verifiseringskode",
  "action": "OPEN_APP",
  "data": { "screen": "register", "step": "verify" }
}
```

Notification 2: KYC Approved

```
{
  "type": "kyc_approved",
  "title": "Kontoen din er godkjent! □□,
  "body": "Du kan nå sende penger og betale med QR",
  "action": "OPEN_APP",
  "data": { "screen": "dashboard" }
}
```

Notification 3: KYC Rejected

```
{
  "type": "kyc_rejected",
```

```
  "title": "Verifisering feilet",
  "body": "Kontakt kundeservice for hjelp",
  "action": "OPEN_SUPPORT",
  "data": { "screen": "profile", "tab": "support" }
}
```

8.3 In-App Prompts

Dashboard Banner (BankID not linked):

```
<div className="bg-gradient-to-r from-[#0B6E35] to-[#095a2b] rounded-2xl p-6 text-white mb-6">
  <h3 className="font-bold mb-2">Koble BankID for å låse opp alle funksjoner</h3>
  <p className="text-sm text-white/90 mb-4">
    Send penger til utlandet og betal i butikk med QR
  </p>
  <button className="bg-white text-[#0B6E35] py-2 px-4 rounded-xl font-medium">
    Koble BankID nå
  </button>
</div>
```

Transaction Attempt Without BankID:

```
// User clicks "Send Money" without BankID
<div className="fixed inset-0 bg-black/50 flex items-center justify-center z-50">
  <div className="bg-white rounded-2xl p-6 max-w-md mx-4">
    <div className="w-16 h-16 bg-[#0B6E35]/10 rounded-full flex items-center justify-center
mx-auto mb-4">
      <Lock className="w-8 h-8 text-[#0B6E35]" />
    </div>
    <h2 className="text-2xl font-bold text-center mb-2">BankID påkrevd</h2>
    <p className="text-[#64748B] text-center mb-6">
      For å sende penger må du først koble BankID. Dette sikrer at pengene dine er trygge.
    </p>
    <button className="w-full bg-[#0B6E35] text-white py-3 rounded-xl font-medium">
      Koble BankID
    </button>
    <button className="w-full text-[#64748B] py-3">
      Avbryt
    </button>
  </div>
```

```
</div>
```

---

9. Legal Consents

9.1 Required Consents

Terms of Service:

- Checkbox at registration: "Jeg godtar vilkårene for bruk"
- Must be checked to proceed
- Links to `landing/pages/vilkar.html`

Privacy Policy:

- Checkbox at registration: "Jeg godtar personvernerklæringen"
- Must be checked to proceed
- Links to `landing/pages/privacy.html`

PSD2 AISP/PISP Consent:

- Modal at BankID connection:

```
Tilgang til bankkontoen din

Ved å koble BankID gir du Drop tillatelse til å:
 Lese saldo på din bankkonto (AISP)
 Initiere betalinger fra din bankkonto (PISP)

Du kan trekke tilbake samtykket når som helst.

[Godta og fortsett] [Avbryt]
```

Marketing Consent (Optional):

- Checkbox at registration: "Jeg ønsker å motta tips og tilbud fra Drop (valgfritt)"
- Default: unchecked
- Can be changed later in settings

9.2 Consent Storage

Database Schema:

```
-- Table already exists (architecture-document.md line 157)
CREATE TABLE IF NOT EXISTS consents (
  id TEXT PRIMARY KEY,
  user_id TEXT NOT NULL,
  consent_type TEXT NOT NULL, -- 'terms', 'privacy', 'psd2_aisp', 'psd2_pisp', 'marketing'
  granted INTEGER NOT NULL, -- 0 or 1
  granted_at TEXT,
  withdrawn_at TEXT,
  ip_address TEXT,
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);

CREATE INDEX idx_consents_user ON consents(user_id);
CREATE INDEX idx_consents_type ON consents(consent_type);
```

Recording Consents:

```
// At registration (terms + privacy)
await run(
  `INSERT INTO consents (id, user_id, consent_type, granted, granted_at, ip_address)
   VALUES (?, ?, ?, 1, datetime('now'), ?)`,
  [randomId("con"), userId, "terms", ip]
);

await run(
  `INSERT INTO consents (id, user_id, consent_type, granted, granted_at, ip_address)
   VALUES (?, ?, ?, 1, datetime('now'), ?)`,
  [randomId("con"), userId, "privacy", ip]
);

// Optional marketing
if (marketingConsent) {
  await run(
    `INSERT INTO consents (id, user_id, consent_type, granted, granted_at, ip_address)
     VALUES (?, ?, ?, 1, datetime('now'), ?)`,
    [randomId("con"), userId, "marketing", ip]
  );
}
```

```
// At BankID connection (PSD2 AISP + PISP)
await run(
  `INSERT INTO consents (id, user_id, consent_type, granted, granted_at, ip_address)
   VALUES (?, ?, ?, 1, datetime('now'), ?), (?, ?, ?, 1, datetime('now'), ?)`,
  [randomId("con"), userId, "psd2_aisp", ip, randomId("con"), userId, "psd2_pisp", ip]
);
```

Withdrawing Consent:

```
// User withdraws PSD2 consent (disconnect bank account)
await run(
  `UPDATE consents
   SET granted = 0, withdrawn_at = datetime('now')
   WHERE user_id = ? AND consent_type IN ('psd2_aisp', 'psd2_pisp')`,
  [userId]
);

// Unlink bank account
await run("DELETE FROM bank_accounts WHERE user_id = ?", [userId]);

// Audit log
logAudit({
  userId,
  action: "consent.withdrawn",
  resourceType: "consent",
  details: { types: ["psd2_aisp", "psd2_pisp"] },
});
```

9.3 GDPR Compliance

Right to Access:

```
// GET /api/gdpr/data-export
// Returns JSON with all user data
{
  "user": { /* user record */ },
  "bank_accounts": [ /* accounts */ ],
  "transactions": [ /* transactions */ ],
  "consents": [ /* consents */ ],
```

```
  "audit_log": [ /* audit entries */ ]
}
```

Right to Erasure:

```
// DELETE /api/gdpr/delete-account
// Soft delete: sets deleted_at, anonymizes PII
await run(
  `UPDATE users SET
    email = 'deleted_' || id || '@drop.deleted',
    first_name = 'Deleted',
    last_name = 'User',
    phone = NULL,
    national_id_hash = NULL,
    deleted_at = datetime('now')
   WHERE id = ?`,
  [userId]
);

// Anonymize audit logs (keep records for compliance, remove PII)
await run(
  `UPDATE audit_log SET
    details = json_set(details, '$.email', 'REDACTED')
   WHERE user_id = ?`,
  [userId]
);
```

Data Retention:

- Active users: indefinite
- Deleted users: 90 days (then full purge)
- Transaction records: 5 years (AML compliance)
- Audit logs: 7 years (regulatory requirement)

---

10. Acceptance Criteria

10.1 Functional Requirements

Registration:

- [ ] User can register with email, password, name, phone, DOB
- [ ] Age validation rejects users < 18 years
- [ ] Password complexity enforced (8+ chars, upper, lower, digit, special)
- [ ] Norwegian phone number required (+47)
- [ ] Duplicate email detection
- [ ] OTP generated and logged (SMS not sent in demo mode)

Phone OTP Verification:

- [ ] User receives 6-digit OTP (logged to console in demo mode)
- [ ] OTP expires after 5 minutes
- [ ] OTP marked as used after successful verification
- [ ] Rate limiting: 5 OTP attempts per minute per IP
- [ ] Generic error messages (no user enumeration)
- [ ] Resend OTP functionality

PIN Setup:

- [ ] User sets 4-digit PIN
- [ ] Weak PIN patterns rejected (0000, 1234, 1111, sequential)
- [ ] PIN stored as bcrypt hash
- [ ] Auto-advance to onboarding on 4th digit
- [ ] Backend endpoint `/api/auth/set-pin` implemented

Onboarding Tour:

- [ ] 4-screen carousel (Welcome, Benefits, BankID, Ready)
- [ ] Progress indicator shows current step
- [ ] Skip button available (except last screen)
- [ ] Back button visible (except first screen)
- [ ] Tour completion tracked in database
- [ ] Backend endpoint `/api/onboarding/complete` implemented

BankID Verification:

- [ ] User redirected to BankID OAuth flow
- [ ] CSRF protection via state token
- [ ] Callback extracts fødselsnummer from ID token

- [ ] Age verification from fødselsnummer (reject if < 18)
- [ ] Fødselsnummer stored as SHA-256 hash
- [ ] `bankid_verified` flag set to 1
- [ ] KYC initiation triggered after BankID success
- [ ] Dashboard blocks unverified users with modal

KYC Check:

- [ ] Demo mode: auto-approve KYC
- [ ] Production mode: redirect to Sumsub widget
- [ ] Webhook handler updates KYC status
- [ ] Pending state shows yellow banner on dashboard
- [ ] Rejected state shows red banner with support link
- [ ] Transaction routes blocked until KYC approved
- [ ] Push notification sent on KYC approval/rejection

10.2 Non-Functional Requirements

Performance:

- [ ] Registration API responds < 500ms (p95)
- [ ] OTP verification API responds < 300ms (p95)
- [ ] BankID OAuth redirect < 1s
- [ ] Onboarding UI loads < 1.5s (FCP)

Security:

- [ ] Password hashed with bcrypt (rounds: 12)
- [ ] PIN hashed with bcrypt (rounds: 12)
- [ ] JWT in httpOnly cookie (no localStorage)
- [ ] CSRF protection on BankID OAuth
- [ ] Rate limiting on all auth endpoints
- [ ] Audit trail for all user actions
- [ ] No PII in logs (phone numbers hashed)

Accessibility:

- [ ] WCAG 2.1 AA compliance

- [ ] Screen reader support
- [ ] Keyboard navigation
- [ ] Focus indicators on all interactive elements
- [ ] Error messages accessible (aria-live regions)

Usability:

- [ ] Mobile-first responsive design
- [ ] Touch targets >= 44px
- [ ] Clear error messages (Norwegian)
- [ ] Loading states for all async operations
- [ ] Success/error feedback for all actions

Legal Compliance:

- [ ] Terms of Service consent required
- [ ] Privacy Policy consent required
- [ ] PSD2 AISP/PISP consent modal at BankID
- [ ] Consent storage in database
- [ ] Age requirement enforced (18+)
- [ ] Norwegian residency requirement enforced (+47 phone, BankID)

10.3 Edge Case Coverage

- [ ] Age < 18 rejected (frontend + backend + BankID)
- [ ] BankID timeout handled gracefully
- [ ] BankID user cancellation handled
- [ ] KYC rejection flow implemented
- [ ] OTP expiry handled with resend
- [ ] Network errors retry with backoff
- [ ] User abandonment tracked in analytics
- [ ] Re-engagement emails triggered

---

11. Implementation Order

Phase 1: Backend Foundations (Priority: HIGH)

Duration: 2 days **Owner:** Backend agent

1. ☐ Database schema updates
   - Add new fields to `users` table (pin_hash, bankid_verified, onboarding_completed, national_id_hash)
   - Create `onboarding_progress` table
   - Create indexes
2. ☐ Missing API endpoints
   - `POST /api/auth/set-pin`
   - `GET /api/auth/bankid/callback`
   - `POST /api/onboarding/complete`
   - `POST /api/auth/resend-otp`
   - `POST /api/webhooks/sumsub`
3. ☐ Age verification logic
   - Extract DOB from fødselsnummer
   - Validate age >= 18 in BankID callback
4. ☐ Consent storage
   - Record consents at registration and BankID

Phase 2: Frontend Fixes (Priority: HIGH)

Duration: 1 day **Owner:** Frontend agent

1. ☐ PIN setup backend integration
   - Call `/api/auth/set-pin` after PIN entered
   - Validate weak PIN patterns
   - Handle errors
2. ☐ Onboarding completion tracking
   - Call `/api/onboarding/complete` on last screen
3. ☐ Dashboard BankID prompt
   - Non-dismissable modal for unverified users
   - "Koble BankID" button
   - Clear explanation
4. ☐ KYC status UI
   - Pending banner (yellow)
   - Rejected banner (red) with support link

Phase 3: Edge Case Handling (Priority: MEDIUM)

Duration: 1 day **Owner:** Backend + Frontend agents

1. ☐ OTP resend flow
   - Backend: Rate limiting (3 per hour)
   - Frontend: "Send ny kode" button
2. ☐ BankID error handling
   - Timeout retry
   - User cancellation
   - CSRF validation
3. ☐ KYC rejection flow
   - Webhook handler
   - Dashboard blocking UI
   - Support ticket creation

Phase 4: Analytics & Re-engagement (Priority: LOW)

Duration: 1 day **Owner:** Backend + Marketing

1. ☐ Drop-off tracking
   - Event logging in `onboarding_progress`
   - Funnel report SQL query
2. ☐ Email triggers
   - OTP not verified (1h)
   - BankID not linked (24h)
   - KYC pending (48h)
3. ☐ Push notifications
   - OTP resend available
   - KYC approved
   - KYC rejected

Phase 5: Testing & Deployment (Priority: HIGH)

Duration: 2 days **Owner:** QA agent + DevOps

1. ☐ Unit tests
   - Age validation (frontend + backend)
   - OTP verification
   - PIN validation
   - BankID callback
2. ☐ Integration tests
   - Full onboarding flow (register → KYC approved)
   - BankID OAuth flow
   - KYC webhook
3. ☐ E2E tests (Playwright)
   - Happy path: Register → Verify → BankID → Dashboard

- Error path: Age < 18 → Rejected
- Error path: BankID timeout → Retry

4. □ Deployment
   - Staging environment
   - Smoke tests
   - Production rollout

---

12. Success Metrics (3 Months Post-Launch)

Metric	Target	Measurement
Registration Completion	85%	OTP sent / Registration started
OTP Verification	80%	OTP verified / OTP sent
Onboarding Completion	70%	Tour complete / OTP verified
BankID Connection	75%	BankID verified / Tour complete
KYC Approval	90%	KYC approved / BankID verified
Overall Conversion	40%	Fully verified / Registration started
Time to Verify	< 2 hours	Median time from registration to KYC approval
Drop-off Rate (Tour)	< 15%	Users who skip tour
Re-engagement Open Rate	25%	Email open rate for re-engagement campaigns
Support Tickets (KYC)	< 5%	Tickets per verified user

---

13. Rollout Plan

13.1 Soft Launch (Week 1)

- **Audience:** Internal team (10 users)
- **Goal:** Validate full flow, catch bugs
- **KYC:** Demo mode (auto-approve)
- **Monitoring:** Manual testing, bug reports in Slack

13.2 Beta Launch (Week 2-3)

- **Audience:** Friends & family (50 users)
- **Goal:** Gather UX feedback, measure conversion rates
- **KYC:** Demo mode (auto-approve)
- **Monitoring:** Analytics dashboard, user feedback survey

13.3 Limited Public Launch (Week 4-6)

- **Audience:** Invite-only (500 users)
- **Goal:** Test production BankID + KYC integration
- **KYC:** Production mode (Sumsub)
- **Monitoring:** Drop-off tracking, support ticket volume

13.4 Full Public Launch (Week 7+)

- **Audience:** Open to all (Norway)
- **Goal:** Scale to 10,000+ users
- **KYC:** Production mode (Sumsub)
- **Monitoring:** Weekly funnel reports, monthly conversion review

---

14. Appendix

14.1 Glossary

Term	Definition
AISP	Account Information Service Provider (PSD2) — reads bank account balance
PISP	Payment Initiation Service Provider (PSD2) — initiates payments from bank account
BankID	Norwegian eID system (OAuth 2.0 OIDC) for identity verification
Fødselsnummer	11-digit Norwegian national ID (encodes DOB + unique ID)
KYC	Know Your Customer — identity verification for AML compliance
Sumsub	Third-party KYC provider (document verification)
OTP	One-Time Password (6-digit SMS code)
SCA	Strong Customer Authentication (PSD2 requirement)

Term	Definition
Pass-through model	Drop never holds customer money; all funds stay in user's bank

14.2 References

Document	Location
Architecture Document	`project/architecture/architecture-document.md`
Terms of Service	`landing/pages/vilkar.html`
Privacy Policy	`landing/pages/privacy.html`
Figma Make Export (UI Source of Truth)	`mockups/figma-make-export/src/components/`
Current Onboarding Flow	`src/drop-app/src/app/onboarding/page.tsx`
Current Register Flow	`src/drop-app/src/app/register/page.tsx`
BankID OAuth	`src/drop-app/src/app/api/auth/bankid/route.ts`
KYC Service	`src/drop-app/src/lib/services/kyc.ts`

14.3 Related Tasks

Task	MC #	Description	Status
Implement user registration	#947	Email/password registration	□ Done
Implement BankID OAuth	#948	BankID integration	◫ Partial
Implement KYC verification	#949	Sumsub integration	◫ Partial
Build onboarding tour	#950	4-screen carousel	□ Done
Add consent tracking	#951	GDPR compliance	□ Not started
Analytics integration	#952	Posthog/Mixpanel	□ Not started

End of Specification

Next Steps:

1. Review this spec with Alem for approval
2. Create implementation tasks in Mission Control
3. Assign tasks to builder agents
4. Begin Phase 1 (Backend Foundations)

Questions for Alem:

1. Preferred analytics platform (Posthog, Mixpanel, custom)?

2. SMS provider for OTP (Twilio, MessageBird)?
3. Email provider for re-engagement (SendGrid, Mailgun)?
4. KYC provider credentials (Sumsub account setup)?
5. BankID production credentials (when to request)?

Revision #3
Created 2026-02-18 08:44:46 UTC by John
Updated 2026-05-24 20:00:46 UTC by John