Login & Authentication Flow

Flow: Login & Authentication

Document: LLD-001 **Version:** 1.0 **Date:** 2026-02-21 **Author:** Frontend Architect (AI Agent) **Status:** Draft **Scope:** End-to-end login flow for web and mobile, including BankID OIDC, session management, demo mode, and error handling

---

1. Overview

Drop uses **BankID OIDC** as the sole production authentication method. Email/password login exists only in demo/dev mode. Authentication produces a JWT stored as an httpOnly cookie (web) or Bearer token (mobile). The login flow includes BankID redirect, loading states, token receipt, session persistence, and comprehensive error handling.

Key facts:

- BankID is mandatory for production (PSD2/SCA compliance)
- Demo mode provides email/password fallback for development
- JWT lifetime: 7d (all clients)
- Session tracking via SHA-256 token hash in `sessions` table
- Age verification (>= 18) enforced during BankID callback

---

2. Web Login Flow (BankID OIDC)

2.1 Sequence Diagram — Web BankID Login

```
sequenceDiagram
    actor User
    participant Browser as Browser<br/>(Next.js Client)
    participant BFF as Next.js BFF<br/>(/api/auth/bankid)
    participant BankID as BankID OIDC<br/>Provider
```

```
    participant DB as SQLite/PostgreSQL

    User->>Browser: Navigate to /login
    Browser->>Browser: Render login page<br/>(BankID + Vipps buttons)
    User->>Browser: Click "BankID" button

    Browser->>BFF: GET /api/auth/bankid
    BFF->>BFF: Rate limit check (10/min per IP)
    BFF->>BFF: Generate state + nonce
    BFF->>BFF: Set bankid_state httpOnly cookie
    BFF-->>Browser: { redirectUrl }

    Browser->>BankID: Redirect to BankID authorize URL<br/>(client_id, redirect_uri, state,
nonce, scope=openid)
    BankID->>User: BankID authentication UI<br/>(code device, app, or biometric)
    User->>BankID: Authenticate with BankID

    BankID-->>Browser: 302 → /api/auth/bankid/callback?code=XXX&state=YYY

    Browser->>BFF: GET /api/auth/bankid/callback?code&state
    BFF->>BFF: Verify state matches bankid_state cookie
    BFF->>BankID: POST /token (exchange code for tokens)
    BankID-->>BFF: { id_token, access_token }
    BFF->>BFF: Verify id_token signature (JWKS)
    BFF->>BFF: Parse pid (national ID, 11 digits)
    BFF->>BFF: Verify age >= 18 from pid birthdate
    BFF->>DB: SELECT user WHERE national_id_hash = SHA-256(pid)

    alt New user
        BFF->>DB: INSERT user (kyc_status=approved, auth_provider=bankid)
        BFF->>DB: INSERT default settings (NOK, nb)
    end

    BFF->>DB: INSERT session (token_hash, expires_at)
    BFF->>BFF: Sign JWT (userId, email, role)
    BFF->>BFF: Set drop_token httpOnly cookie (7d, secure, sameSite=Lax)
    BFF-->>Browser: 302 → /dashboard

    Browser->>Browser: Router navigates to /dashboard
```

```
    Browser->>BFF: GET /api/auth/me (with cookie)
    BFF->>DB: Verify session not revoked
    BFF-->>Browser: { user, bankAccounts, totalBalance }
    Browser->>Browser: Render dashboard with user data
```

2.2 Demo Mode Login (Development Only)

In development (`isDemoMode()` returns true), a demo login endpoint is available. It loads a fixed demo user (`usr_demo1`, email: `demo@example.test`) without requiring credentials:

```
sequenceDiagram
    actor User
    participant Browser as Browser<br/>(/login page)
    participant API as Next.js API<br/>(/api/auth/login)
    participant DB as SQLite

    User->>Browser: Click "Demo Login"

    Browser->>API: POST /v1/auth/demo-login (no credentials)

    API->>API: Check isDemoMode()
    API->>DB: SELECT user WHERE id = 'usr_demo1'
    Note over API: Fixed demo user (demo@example.test)

    alt Demo mode active
        API->>DB: INSERT session
        API->>API: Sign JWT, set httpOnly cookie
        API-->>Browser: 200 { user, token }
        Browser->>Browser: router.push("/dashboard")
    else Demo mode disabled
        API-->>Browser: 404 { error: "not_found" }
    end
```

---

3. Mobile Login Flow (BankID OIDC)

3.1 Sequence Diagram — Mobile BankID Login

```
sequenceDiagram
    actor User
    participant App as Expo App
    participant WebBrowser as expo-web-browser
    participant API as Hono API<br/>(/v1/auth)
    participant BankID as BankID OIDC
    participant DB as Database

    User->>App: Open app, tap "Logg inn"
    App->>API: GET /v1/auth/bankid/initiate?platform=mobile
    API->>API: Rate limit check
    API->>API: Generate state + nonce
    API-->>App: { redirectUrl, state }

    App->>WebBrowser: Open BankID URL<br/>(expo-web-browser)
    WebBrowser->>BankID: BankID authorize URL
    BankID->>User: BankID authentication
    User->>BankID: Authenticate

    BankID-->>WebBrowser: Redirect to drop://auth/callback?code&state
    WebBrowser-->>App: Deep link intercept

    App->>API: POST /v1/auth/bankid/callback<br/>{ code, state, platform: "mobile" }
    API->>BankID: Exchange code for tokens
    BankID-->>API: { id_token }
    API->>API: Verify id_token (JWKS)
    API->>API: Parse pid, verify age >= 18
    API->>DB: Find or create user

    alt New user
        API->>DB: INSERT user (kyc_status=approved)
    end

    API->>DB: INSERT session
    API->>API: Sign JWT (7d expiry)
    API-->>App: { token, data: { user } }

    App->>App: Store token in AsyncStorage
    App->>App: Navigate to (tabs) dashboard
```

```
    Note over App: Future: biometric unlock<br/>(Face ID / Touch ID)
```

4. Authentication State Diagram

```
stateDiagram-v2
    [*] --> Unauthenticated: App launch

    Unauthenticated --> BankIDRedirect: Click "BankID"
    Unauthenticated --> DemoLogin: Enter credentials (dev mode)

    BankIDRedirect --> BankIDAuthenticating: Browser opens BankID
    BankIDAuthenticating --> CallbackProcessing: BankID returns code
    BankIDAuthenticating --> BankIDError: Auth failed/cancelled/timeout

    CallbackProcessing --> Authenticated: Valid token + session created
    CallbackProcessing --> AgeRejected: User under 18
    CallbackProcessing --> BankIDError: Token verification failed

    DemoLogin --> Authenticated: Valid credentials
    DemoLogin --> LoginError: Invalid credentials

    Authenticated --> SessionActive: JWT valid + session not revoked
    SessionActive --> TokenExpired: JWT expired (7d all platforms)
    SessionActive --> SessionRevoked: Logout or admin revoke
    SessionActive --> Authenticated: Token refresh

    TokenExpired --> Unauthenticated: Redirect to /login
    SessionRevoked --> Unauthenticated: Clear cookie/token
    AgeRejected --> Unauthenticated: Show age error
    BankIDError --> Unauthenticated: Show error + retry
    LoginError --> Unauthenticated: Show error message
```

5. Error States

5.1 Error State Table

Error	Cause	User-Facing Message (Norwegian)	Recovery Action
BankID Unavailable	BankID service down	"BankID er midlertidig utilgjengelig. Prøv igjen senere."	Retry button, show status page link
BankID Timeout	User took too long (>5min)	"BankID-sesjonen utløp. Vennligst prøv igjen."	Auto-redirect back to login page
BankID Cancelled	User cancelled authentication	"Innlogging avbrutt. Trykk 'BankID' for å prøve igjen."	Show login page with BankID button
State Mismatch	CSRF attack or stale session	"Noe gikk galt. Vennligst prøv å logge inn på nytt."	Clear state cookie, redirect to /login
Token Verification Failed	Invalid/tampered id_token	"Autentisering mislyktes. Prøv igjen."	Redirect to /login
Age Under 18	User is younger than 18	"Du må være minst 18 år for å bruke Drop."	No retry — age requirement is firm
Rate Limited	Too many login attempts	"For mange forsøk. Vent litt og prøv igjen."	Wait and retry (10/min limit)
Invalid Credentials (Demo)	Wrong email or password	"Feil e-post eller passord."	Re-enter credentials
Session Expired	JWT expired	"Sesjonen din har utløpt. Logg inn igjen."	Redirect to /login
Session Revoked	Logout from another device	"Du har blitt logget ut."	Re-login via BankID
Network Error	No connectivity	"Ingen nettverkstilkobling. Sjekk internett."	Retry when connectivity restored

5.2 Error Handling by Platform

Platform	Error Display	Navigation
Web	Inline error message on login page, red text below form	`router.push("/login")` on session errors
Mobile	Alert dialog or inline error text	`router.replace("/")` (welcome screen) on session errors

---

6. Session Management

6.1 Token Storage

Platform	Storage	Token Name	Flags

Web	httpOnly cookie	`drop_token`	httpOnly, secure, sameSite=Lax
Mobile	AsyncStorage	Bearer token	In-memory variable + persistent storage

6.2 Session Lifecycle

Event	Action	Database
Login success	Create session record	`INSERT INTO sessions (id, user_id, token_hash, expires_at)`
Each request	Verify session valid	`SELECT * FROM sessions WHERE token_hash = ? AND revoked = 0 AND expires_at > now()`
Token refresh	New session, revoke old	`INSERT new session`, `UPDATE old session SET revoked = 1`
Logout	Revoke all sessions	`UPDATE sessions SET revoked = 1 WHERE user_id = ?`
Admin action	Revoke specific session	`UPDATE sessions SET revoked = 1 WHERE id = ?`

6.3 Token Refresh

`POST /v1/auth/refresh` refreshes the user's session:

1. Reads `drop_token` cookie / Bearer token
2. Verifies current JWT and session validity
3. Revokes old session (`UPDATE sessions SET revoked = 1`)
4. Creates new session + JWT
5. Sets new `drop_token` cookie (Max-Age=604800, HttpOnly, SameSite=Lax)
6. Returns new user data

6.4 Deprecated Endpoints

The following endpoints return `410 Gone` and exist only for backward compatibility:

Endpoint	Status	Reason
`POST /v1/auth/login`	410 Gone	Replaced by BankID OIDC flow
`POST /v1/auth/register`	410 Gone	User creation is automatic on first BankID login
`POST /v1/auth/verify-otp`	410 Gone	OTP flow removed with BankID migration

6.5 useAuth Hook (Web)

The `useAuth()` hook on the web client:

1. Calls `GET /api/auth/me` on mount
2. If 401 → redirects to `/login`
3. Returns `{ user, loading }` to the page component
4. Every authenticated page wraps content with `if (loading) return <Skeleton />`

---

7. UI Components Involved

7.1 Web Login Page (`/login`)

Component	Source	Purpose
`Image`	next/image	Drop logo display
`Link`	next/link	Navigation to /register, /dashboard
`Button`	shadcn/ui	Submit button, social login buttons
`Mail`	lucide-react	Email input icon
`Lock`	lucide-react	Password input icon
`Eye` / `EyeOff`	lucide-react	Password visibility toggle
`ArrowRight`	lucide-react	Submit button icon

7.2 Mobile Login Screen (`login.js`)

Element	Implementation	Purpose
Email input	`<TextInput>`	Email entry
Password input	`<TextInput secureTextEntry>`	Password entry
Login button	`<TouchableOpacity>`	Trigger `api.login()`
Register link	`router.push("/register")`	Navigate to registration

---

8. Accessibility Considerations (WCAG 2.1 AA)

Requirement	Implementation
Form labels	All inputs have associated `<label>` elements
Error announcements	Error messages use `role="alert"` for screen readers
Focus management	After error, focus returns to the first invalid field
Color contrast	Error red (#EF4444) on white background meets 4.5:1 ratio
Keyboard navigation	Tab order follows visual order: email → password → submit → social buttons
Password visibility	Toggle button has aria-label "Vis passord" / "Skjul passord"
Loading state	Submit button shows loading spinner and is disabled during request

---

9. Cross-References

- **BankID OIDC integration details:** See Authentication
- **API endpoints:** `GET /v1/auth/bankid/initiate`, `POST /v1/auth/bankid/callback`, `POST /v1/auth/demo-login`, `POST /v1/auth/refresh`, `GET /api/auth/me` — See API Reference
- **Session database schema:** `sessions` table — See Database Schema
- **Component overview:** See component-overview.md
- **Figma login screen:** `mockups/figma-make-export/src/app/screens/Login.tsx`
- **Web login page:** `src/drop-app/src/app/login/page.tsx` — See PAGES.md
- **Registration flow:** See flow-registration-onboarding.md

Revision #7
Created 2026-02-21 05:58:50 UTC by John
Updated 2026-06-28 20:01:33 UTC by John