Frontend Architecture

Pages

Drop Frontend — Pages

> All pages are in `src/drop-app/src/app/` using Next.js App Router file-based routing.

Page Index

Route	File	Type	Auth Required	BottomNav
`/`	`page.tsx`	Server	No	No
`/login`	`login/page.tsx`	Client	No	No
`/register`	`register/page.tsx`	Client	No	No
`/dashboard`	`dashboard/page.tsx`	Client	Yes	Yes
`/accounts`	`accounts/page.tsx`	Client	Yes	Yes
`/transactions`	`transactions/page.tsx`	Client	Yes	Yes
`/scan`	`scan/page.tsx`	Client	Yes	Yes
`/send`	`send/page.tsx`	Client	Yes	No
`/profile`	`profile/page.tsx`	Client	Yes	Yes
`/profile/personal`	`profile/personal/page.tsx`	Client	Yes	No
`/profile/security`	`profile/security/page.tsx`	Client	Yes	No
`/profile/notifications`	`profile/notifications/page.tsx`	Client	Yes	No
`/profile/language`	`profile/language/page.tsx`	Client	Yes	No
`/notifications`	`notifications/page.tsx`	Client	Yes	Yes
`/cards`	`cards/page.tsx`	Client	Yes	No
`/complaints`	`complaints/page.tsx`	Client	No	No

Route	File	Type	Auth Required	BottomNav
`/fees`	`fees/page.tsx`	Client	No	No
`/privacy`	`privacy/page.tsx`	Client	No	No
`/terms`	`terms/page.tsx`	Client	No	No
`/withdrawal`	`withdrawal/page.tsx`	Client	No	No

Page Details

`/` — Home / Marketing Page

- **File:** `app/page.tsx`
- **Type:** Server component (no "use client")
- **Auth:** None
- **Components used:** DropLogoFull, DropAppIcon, Link, Image, custom icons (IconSendMoney, IconQrScan, IconVirtualCard, IconShield, IconFastTransfer, IconCorridors)
- **Data:** Static arrays `features` (3 items) and `stats` (3 items) defined inline
- **Sections:**
  1. Header with DropLogoFull + nav links (Tjenester, Priser, Om oss, Logg inn, Kom i gang)
  2. Hero with headline, subtext, CTA buttons, phone mockup placeholder
  3. Features grid (Send penger, Betal med QR, Virtuelt kort)
  4. Stats bar (0.5% Gebyr, <2t Leveringstid, 30+ Land)
  5. Trust section (BankID verified, Rask overføring, 30+ land)
  6. Merchant CTA section
  7. Footer with ALAI Holding AS credit

`/login` — Login

- **File:** `app/login/page.tsx`
- **Type:** Client component
- **Auth:** None (entry point)
- **Components used:** Image, Link, Button, Mail/Lock/Eye/EyeOff/ArrowRight (lucide)
- **State:** email, password, showPassword, error, loading
- **Data fetching:** POST `/api/auth/login` with `{ email, password }`
- **Validation:** Email regex `^[^\s@]+@[^\s@]+\.[^\s@]+$`, required fields check
- **On success:** `router.push("/dashboard")`
- **Social login buttons:** BankID, Vipps (UI only, not functional)
- **Dev mode:** Shows demo credentials `amir@example.com / demo1234`

`/register` — Registration

- **File:** `app/register/page.tsx`
- **Type:** Client component
- **Auth:** None
- **Components used:** ArrowLeft/ArrowRight/Check/Eye/EyeOff/Phone/Mail/User/Calendar/Lock (lucide), Button
- **State:** step (1-4), form fields, otp, pin, errors
- **Steps:**
  1. **Info** — firstName, lastName, email, phone (+47 prefix), dateOfBirth, password
  2. **Verify** — 6-digit OTP input (MVP: any 6 digits accepted)
  3. **PIN** — 4-digit PIN with custom numpad UI
  4. **Success** — Welcome message, redirect to /dashboard
- **Validation:**
  - Age >= 18 (calculated from dateOfBirth)
  - XSS protection: names reject `< > " ' & ; ( ) { } [ ]`
  - Password: min 8 chars, must contain letters AND numbers
  - Phone: must be 8 digits (Norwegian format)
- **Data fetching:** POST `/api/auth/register`

`/dashboard` — Main Dashboard

- **File:** `app/dashboard/page.tsx`
- **Type:** Client component
- **Auth:** Yes (`useAuth()` with redirect)
- **Components used:** DropLogo, BottomNav, ScrollArea, Bell/LogOut (lucide)
- **State:** transactions, loading
- **Data fetching:** GET `/api/transactions?limit=10`
- **Interfaces:** `Transaction { id, type, status, amount, currency, recipientName, createdAt }`
- **Layout:**
  1. Header: DropLogo + notification bell + logout + avatar initials
  2. Balance card: primary account balance, formatted NOK
  3. Action buttons: Send penger (→ /send), Skann QR (→ /scan)
  4. Recent transactions list in ScrollArea
  5. BottomNav

`/accounts` — Bank Accounts

- **File:** `app/accounts/page.tsx`
- **Type:** Client component
- **Auth:** Yes (`useAuth()`)
- **Components used:** BottomNav, Card, ArrowLeft/Landmark/Plus/ChevronRight (lucide)
- **Data:** Reads `user.bankAccounts` from auth hook (no separate fetch)

- **Layout:**
  1. PSD2/Open Banking info banner (blue)
  2. Account cards: bankName, masked accountNumber, balance, currency, isPrimary badge
  3. Total balance summary
  4. "Legg til bankkonto" button (BankID connection note)

`/transactions` — Transaction History

- **File:** `app/transactions/page.tsx`
- **Type:** Client component
- **Auth:** Yes (`useAuth()`)
- **Components used:** BottomNav, Tabs/TabsList/TabsTrigger, ArrowLeft/Clock (lucide)
- **State:** transactions, filter, loading
- **Data fetching:** GET `/api/transactions?type={filter}&limit=50`
- **Filters:** Alle (all), Overforinger (remittance), QR-betalinger (qr_payment)
- **Grouping:** `groupByDate()` function groups into: I dag, I gar, Denne uken, Eldre
- **Display:** Amount with +/- prefix and color coding (green for received, red for sent)

`/scan` — QR Scanner

- **File:** `app/scan/page.tsx`
- **Type:** Client component
- **Auth:** Yes (`useAuth()`)
- **Components used:** BottomNav, Button, ArrowLeft/Camera/Check/X/Store (lucide)
- **State:** scanState (scanning | payment | paying | success), scannedMerchant, amount, paymentResult
- **Interfaces:** `ScannedMerchant { id, name, category }`, `PaymentResult { id, status, amount, fee, merchant }`
- **Flow:**
  1. **Scanning** — Camera viewfinder UI with scan frame, "Simuler skanning" button (demo)
  2. **Payment** — Shows merchant info, amount input, 1% fee calculation
  3. **Paying** — Loading spinner
  4. **Success** — Confirmation with transaction details
- **Data fetching:** POST `/api/transactions/qr-payment` with `{ merchantId, amount }`
- **Demo merchant:** "Ahmetov Kebab" (id: "merchant_001", category: "Restaurant")

`/send` — Send Money (Remittance)

- **File:** `app/send/page.tsx`
- **Type:** Client component
- **Auth:** Yes (`useAuth()`)

- **Components used:** Button, ArrowLeft/ArrowRight/Check/ChevronDown/Globe/User (lucide)
- **State:** step (1-4), selectedRecipient, amount, recipients, rates, sending, txResult
- **Steps:**
  1. **Select Recipient** — List from GET `/api/recipients`, shows name + country flag
  2. **Enter Amount** — NOK input, real-time conversion with exchange rate, 0.5% fee display
  3. **Review** — Summary of recipient, amount, rate, fee, total
  4. **Success** — Confirmation with reference number
- **Data fetching:**
  - GET `/api/recipients` (on mount)
  - GET `/api/rates` (on mount)
  - POST `/api/transactions/remittance` with `{ recipientId, amountNOK, targetCurrency }`
- **Country flags:** RS (Serbia), BA (Bosnia), TR (Turkey), PK (Pakistan), PL (Poland)
- **Interface:** `TxResult { id, status, amount, fee, rate, recipientName, targetAmount, targetCurrency }`

`/profile` — User Profile

- **File:** `app/profile/page.tsx`
- **Type:** Client component
- **Auth:** Yes (`useAuth()`)
- **Components used:** BottomNav, ArrowLeft/ChevronRight/LogOut/Settings/Shield/HelpCircle/Bell/CreditCard/Landmark (lucide)
- **Data:** Reads `user` from auth hook
- **Layout:**
  1. User info card with initials avatar (green bg), full name, email
  2. Menu items: Mine kontoer (→ /accounts), Varsler, Innstillinger, Sikkerhet, Hjelp og stotte
  3. Logout button with confirmation
  4. Version: "Drop v0.1.0 · ALAI Holding AS"

`/withdrawal` — Angrerett (Right of Withdrawal)

- **File:** `app/withdrawal/page.tsx`
- **Type:** Client component
- **Auth:** No
- **Components used:** ChevronLeft, RotateCcw, CheckCircle (lucide)
- **State:** submitted, loading
- **Purpose:** Norwegian angrerettloven compliance — 14-day right of withdrawal form
- **Layout:**
  1. Info section explaining angrerett (right to cancel service agreement within 14 days)

2. Warning banner: Angrerett does not apply to completed payment transactions, only to the service agreement itself
3. Form with optional reason dropdown (not_needed, alternative, not_satisfied, other) and comment textarea
4. Submit button (red) with AML retention notice (data kept for 5 years per hvitvaskingsloven)
5. Success screen with confirmation message (14-day processing time)

`/complaints` — Send Complaint

- **File:** `app/complaints/page.tsx`
- **Type:** Client component
- **Auth:** Yes (`useAuth()`)
- **Components used:** MessageSquare, CheckCircle, ChevronLeft, ExternalLink (lucide)
- **State:** submitted, loading, formData (category, subject, description)
- **Data fetching:** POST `/api/complaints` with `{ category, subject, description }`
- **Purpose:** Finansavtaleloven §3-53 compliance — formal complaint submission with 15 business day response requirement
- **Layout:**
  1. Info text: All complaints taken seriously, up to 15 business days processing time
  2. Form with required fields:
     - Category dropdown: transaction, fees, service, privacy, other
     - Subject text input (max 200 chars)
     - Description textarea (max 2000 chars)
  3. Submit button with POST to `/api/complaints`
  4. External complaint authority section: Finansklagenemnda (FinKN) contact info with link to finansklagenemnda.no
  5. Success screen: Complaint received, 15 business day review commitment

---

`/privacy` — Privacy Policy

- **File:** `app/privacy/page.tsx`
- **Type:** Client component
- **Auth:** None
- **Components used:** ChevronLeft, Shield (lucide)
- **Purpose:** GDPR-compliant privacy policy page (Norwegian language)
- **Sections:**
  1. **Behandlingsansvarlig:** ALAI Holding AS as data controller
  2. **Hvilke opplysninger vi samler inn:** Identification (name, email, phone, BankID), Financial data (bank accounts via Open Banking, transaction history), Technical data (IP, device, app version)

3. **Formaal med behandlingen:** Transaction processing (PSD2/PISP), KYC/AML compliance, AML/terrorism prevention, service improvement
4. **Rettslig grunnlag:** Contract (GDPR 6(1)(b)), Legal obligation (6(1)(c)) for AML/KYC, Legitimate interest (6(1)(f)) for service improvement
5. **Dine rettigheter:** Innsyn (access), Retting (rectification), Sletting (erasure with 5-year AML retention), Dataportabilitet (portability)
6. **Oppbevaring:** Minimum 5 years per hvitvaskingsloven, anonymization on account deletion but AML data retained
7. **Kontakt:** personvern@getdrop.no, complaint to Datatilsynet

---

`/terms` — Terms of Service

- **File:** `app/terms/page.tsx`
- **Type:** Client component
- **Auth:** None
- **Components used:** ChevronLeft, FileText (lucide)
- **Purpose:** Legal terms of service (Norwegian language)
- **Sections:**
  1. **Om tjenesten:** Drop as PISP/AISP under PSD2, provided by ALAI Holding AS
  2. **Krav til brukere:** 18+ age, Norwegian residency with BankID, KYC required, Norwegian phone (+47)
  3. **Betalingsmodell:** Drop never holds customer money, pass-through model via Open Banking
  4. **Gebyrer:** All fees shown before transaction confirmation, see fees page for full list
  5. **Ansvar:** Drop responsible for correct payment execution per betalingstjenesteloven, refund rights per law, not liable for bank/recipient delays
  6. **Misbruk og sperring:** Right to block accounts for suspected AML/fraud, mandatory STR reporting to Økokrim/EFE
  7. **Angrerett:** 14-day withdrawal right per angrerettloven (does not apply to completed transactions)
  8. **Tvister:** Norwegian law, Oslo tingrett jurisdiction, complaint to Finansklagenemnda

---

`/fees` — Fee Overview

- **File:** `app/fees/page.tsx`
- **Type:** Client component
- **Auth:** None
- **Components used:** ChevronLeft, Receipt (lucide)
- **Purpose:** Transparent fee disclosure page (Norwegian language)
- **Sections:**
  1. **Overføring til utlandet:**
     - Transaction fee: 1.5% per transfer

    - Currency markup: 0.5% on mid-market rate
    - Typical total: ~2% (compare with banks at 3-7%)
2. **QR-betaling i butikk:**
    - For customer: Free (no charge to payer)
    - For merchant: 0.5% (lower than card terminals)
3. **Kontotjenester:**
    - Account creation: Free
    - Monthly fee: Free
    - Bank account linking (AISP): Free
4. **Viktig informasjon:**
    - Fees can change with 30-day notice
    - Exchange rates updated in real-time from market
    - Always see final amount before confirming

`/profile/personal` — Personal Information

- **File:** `app/profile/personal/page.tsx`
- **Type:** Client component
- **Auth:** Yes (`useAuth()`)
- **Components used:** ChevronLeft, ShieldCheck (lucide), BottomNav
- **State:** Reads `user` from auth hook
- **Layout:**
    1. User avatar with initials (green gradient)
    2. Full name and email display
    3. Read-only form fields: firstName, lastName, email, phone (+47 987 65 432), dateOfBirth (15. mars 1995)
    4. BankID verification badge (green banner with ShieldCheck icon)
- **Note:** All fields are disabled (cannot edit) — verified via BankID

`/profile/security` — Security Settings

- **File:** `app/profile/security/page.tsx`
- **Type:** Client component
- **Auth:** Yes (`useAuth()`)
- **Components used:** ChevronLeft, ChevronRight, Lock, Smartphone, Laptop (lucide), BottomNav
- **Sections:**
    1. **Passord:** Change password button (shows "Sist endret: Aldri")
    2. **To-faktor autentisering:**
        - BankID verification: Active (green badge)
        - Vipps verification: Not activated (gray badge)
    3. **Aktive enheter:**

    - iPhone 15 Pro: Oslo, Norge — Aktiv nå (green dot indicator)
    - MacBook Pro: Oslo, Norge — I går kl. 18:45
  4. Footer: Support contact (support@getdrop.no)
- **Note:** UI only, no actual functionality connected

`/profile/notifications` — Notification Settings

- **File:** `app/profile/notifications/page.tsx`
- **Type:** Client component
- **Auth:** Yes (`useAuth()`)
- **Components used:** ChevronLeft, Bell, Mail (lucide), BottomNav
- **State:** pushEnabled, emailEnabled, settingsLoaded
- **Data fetching:**
  - GET `/api/settings` (on mount)
  - PATCH `/api/settings` with `{ pushEnabled: boolean }` or `{ emailEnabled: boolean }` (on toggle)
- **Layout:**
  1. Push-varsler toggle switch (Bell icon)
  2. E-postvarsler toggle switch (Mail icon)
- **Behavior:** Toggles immediately update state and send PATCH request, revert on failure

---

`/profile/language` — Language Settings

- **File:** `app/profile/language/page.tsx`
- **Type:** Client component
- **Auth:** Yes (`useAuth()`)
- **Components used:** ChevronLeft, Check (lucide), BottomNav
- **State:** selected, saving
- **Data fetching:**
  - GET `/api/settings` (on mount)
  - PATCH `/api/settings` with `{ language: string }` (on save)
- **Languages:** nb (Norsk Bokmål), en (English), bs (Bosanski), sq (Shqip)
- **Layout:**
  1. Language list with radio selection (green checkmark for selected)
  2. "Lagre" button (green) to save selection
- **Behavior:** Selection updates local state, user must click "Lagre" to persist

---

`/notifications` — Notifications Center

- **File:** `app/notifications/page.tsx`
- **Type:** Client component
- **Auth:** Yes (`useAuth()`)
- **Components used:** BottomNav, ArrowLeft, Bell, ArrowUpRight, ScanLine, Smartphone, TrendingUp (lucide)
- **State:** notifications, fetching
- **Data fetching:**
  - GET `/api/notifications` (on mount)
  - PATCH `/api/notifications` with `{ notificationIds: [ids] }` (mark read, fire-and-forget)
- **Interface:** `Notification { id, type, title, body, read, createdAt }`
- **Layout:**
  1. Header with back button + "Varsler" title
  2. Empty state: Bell icon + "Ingen varsler enna" message
  3. Grouped notifications: I DAG / I GÅR / date groups
  4. Notification cards: icon based on type, title, body, timestamp, unread dot indicator
  5. BottomNav
- **Notification types:** transaction_complete (green), qr_payment (yellow), security (blue), rate_update (yellow), default (gray)
- **Auto-read:** Automatically marks all unread notifications as read on page load
- **Time formatting:** "I dag kl. HH:MM", "I går kl. HH:MM", or "DD.MM.YYYY kl. HH:MM"

`/cards` — Card Management (FUTURE — feature-flagged)

> **Note:** Cards are a FUTURE feature, gated behind feature flags (all default to `false`). Requires a card issuing partner before activation.

- **File:** `app/cards/page.tsx`
- **Type:** Client component
- **Auth:** Yes (`useAuth()`)
- **Feature flags:** `virtualCards` (gate), `physicalCards`, `cardPin`, `spendingLimits` — all default to `false`
- **Components used:** Button, Dialog, various lucide icons (CreditCard, Plus, ArrowLeft, Smartphone, Eye, EyeOff, Lock, X, Check, Copy, RefreshCw)
- **State:** cards, selectedCard, showDetails, showOrderForm, orderForm, loading
- **Data fetching:**
  - GET `/api/cards` (list)
  - POST `/api/cards` (create virtual)
  - POST `/api/cards` with type "physical" + address (order physical)
  - PATCH `/api/cards/{id}` with `{status: "frozen" | "active"}`

    - DELETE `/api/cards/{id}`
- **Card visual:** Green gradient background, masked card number (•••• •••• •••• 4242), expiry, cardholder name

Component Inventory

Component Inventory

> **Project:** {{PROJECT_NAME}} **Version:** {{VERSION}} **Date:** {{DATE}}
> **Author:** {{AUTHOR}} **Status:** Draft | In Review | Approved **Reviewers:** {{REVIEWERS}}

Document History

Version	Date	Author	Changes
0.1	{{DATE}}	{{AUTHOR}}	Initial draft

---

1. Overview

Total components: {{N}} **Library location:** `src/components/` **Storybook:** `{{https://storybook.PROJECT_NAME.example.com}}` **Design source:** `{{Figma file URL}}` **Owner team:** `{{TEAM_NAME}}`

This inventory tracks every reusable component in **{{PROJECT_NAME}}**. It follows Atomic Design categorization: atoms → molecules → organisms → templates → pages.

---

2. Component Hierarchy Diagram

```
graph TB
    subgraph Pages
        PageLogin["LoginPage"]
        PageDashboard["DashboardPage"]
    end
    subgraph Templates
```

```
        TplAuth["AuthLayout"]
        TplApp["AppLayout"]
    end
    subgraph Organisms
        OrgNavbar["Navbar"]
        OrgSidebar["Sidebar"]
        OrgDataTable["DataTable"]
        OrgUserForm["UserForm"]
    end
    subgraph Molecules
        MolFormField["FormField"]
        MolCard["Card"]
        MolSearchBar["SearchBar"]
        MolDropdown["Dropdown"]
        MolPagination["Pagination"]
    end
    subgraph Atoms
        AtomButton["Button"]
        AtomInput["Input"]
        AtomBadge["Badge"]
        AtomSpinner["Spinner"]
        AtomAvatar["Avatar"]
        AtomIcon["Icon"]
    end

    Pages --> Templates
    Templates --> Organisms
    Organisms --> Molecules
    Molecules --> Atoms
```

TODO: Update diagram to reflect actual component tree.

---

3. Atoms (Primitive Components)

3.1 Button

Property	Value

Category	Atom
Status	`{{Done
File path	`src/components/ui/Button/Button.tsx`
Storybook	`{{URL}}`
Design ref	`{{Figma frame URL}}`

Props API:

Prop	Type	Default	Required	Description
`variant`	`'primary' | 'secondary' | 'ghost' | 'danger' | 'link'`	`'primary'`	No	Visual style variant
`size`	`'sm' | 'md' | 'lg'`	`'md'`	No	Button size
`disabled`	`boolean`	`false`	No	Disables interaction
`loading`	`boolean`	`false`	No	Shows spinner, disables click
`leftIcon`	`ReactNode`	`undefined`	No	Icon before label
`rightIcon`	`ReactNode`	`undefined`	No	Icon after label
`onClick`	`(e: MouseEvent) => void`	`undefined`	No	Click handler
`type`	`'button' | 'submit' | 'reset'`	`'button'`	No	HTML button type
`fullWidth`	`boolean`	`false`	No	100% width

Variants & States: primary, secondary, ghost, danger, link × default, hover, focus, active, disabled, loading

Accessibility:

- Renders as `<button>` — never `<div>` or `<span>`
- `loading` state: `aria-busy="true"`, spinner has `aria-label="Loading"`
- `disabled`: `aria-disabled="true"`, not removed from tab order
- Focus ring: 2px offset, brand color

Dependencies: Icon component, Spinner component

3.2 Input

Property	Value
Category	Atom

Property	Value
Status	`{{Status}}`
File path	`src/components/ui/Input/Input.tsx`
Storybook	`{{URL}}`

Props API:

Prop	Type	Default	Required	Description
`type`	`'text' | 'email' | 'password' | 'number' | 'search' | 'tel' | 'url'`	`'text'`	No	Input type
`value`	`string`	—	Yes (controlled)	Input value
`onChange`	`(e: ChangeEvent) => void`	—	Yes	Change handler
`placeholder`	`string`	`''`	No	Placeholder text
`disabled`	`boolean`	`false`	No	Disabled state
`error`	`boolean`	`false`	No	Triggers error visual state
`errorMessage`	`string`	`undefined`	No	Error text (also sets `aria-describedby`)
`leftAdornment`	`ReactNode`	`undefined`	No	Icon or element left of input
`rightAdornment`	`ReactNode`	`undefined`	No	Icon or element right of input
`size`	`'sm' | 'md' | 'lg'`	`'md'`	No	Input height/font size

Variants & States: default, focused, error, disabled, with-left-icon, with-right-icon

Accessibility:

- Must always be paired with `<label>` (use FormField molecule)
- Error: `aria-invalid="true"` + `aria-describedby` pointing to error message id
- Password: toggle visibility button must have `aria-label`

3.3 Badge

Property	Value
Category	Atom
Status	`{{Status}}`

Property	Value
File path	`src/components/ui/Badge/Badge.tsx`

Props API:

Prop	Type	Default	Required	Description
`variant`	`'success' | 'warning' | 'error' | 'info' | 'neutral'`	`'neutral'`	No	Semantic color variant
`size`	`'sm' | 'md'`	`'md'`	No	Badge size
`dot`	`boolean`	`false`	No	Shows colored dot instead of text
`children`	`ReactNode`	—	Yes	Badge content

TODO: Add remaining atom components following the same pattern (Select, Checkbox, Radio, Toggle, Textarea, Avatar, Tooltip, Spinner, Divider).

---

4. Molecules (Composite Components)

4.1 FormField

Property	Value
Category	Molecule
Status	`{{Status}}`
File path	`src/components/ui/FormField/FormField.tsx`
Composes	Input, Label, ErrorMessage, HelpText

Props API:

Prop	Type	Default	Required	Description
`label`	`string`	—	Yes	Visible label text
`htmlFor`	`string`	—	Yes	Links label to input id
`error`	`string`	`undefined`	No	Error message text
`helpText`	`string`	`undefined`	No	Helper text below input
`required`	`boolean`	`false`	No	Shows required indicator

Prop	Type	Default	Required	Description
`children`	`ReactNode`	—	Yes	Input component

Accessibility: Label always associated with input via `htmlFor`/`id` pair.

4.2 Card

Property	Value
Category	Molecule
Status	`{{Status}}`
File path	`src/components/ui/Card/Card.tsx`

Props API:

Prop	Type	Default	Required	Description
`variant`	`'default' | 'bordered' | 'elevated'`	`'default'`	No	Visual style
`padding`	`'sm' | 'md' | 'lg' | 'none'`	`'md'`	No	Internal padding
`as`	`ElementType`	`'div'`	No	Polymorphic render element
`children`	`ReactNode`	—	Yes	Card content

Sub-components: `Card.Header`, `Card.Body`, `Card.Footer`

TODO: Add remaining molecule components: Modal, Dropdown, Table, Pagination, Toast, SearchBar, Breadcrumb, Tabs, Accordion, DatePicker.

---

5. Organisms (Complex Components)

5.1 DataTable

Property	Value
Category	Organism
Status	`{{Status}}`
File path	`src/components/features/DataTable/DataTable.tsx`

Property	Value
Dependencies	Table (molecule), Pagination, SearchBar, Spinner, Badge

Features:

- Sortable columns (client + server-side)
- Pagination (configurable page sizes)
- Row selection (single + multi)
- Column visibility toggle
- Export action slot
- Loading skeleton state
- Empty state slot
- Row action slot (per-row dropdown)

TODO: Document full props API for DataTable.

5.2 Navigation / Sidebar

Property	Value
Category	Organism
Status	`{{Status}}`
File path	`src/components/layouts/Sidebar/Sidebar.tsx`

TODO: Add remaining organism components.

---

6. Shared Hooks / Composables Inventory

Hook	File	Purpose	Used By
`useDebounce`	`src/hooks/useDebounce.ts`	Debounce value changes	SearchBar, Input
`useLocalStorage`	`src/hooks/useLocalStorage.ts`	Persistent local state	Theme, preferences
`useMediaQuery`	`src/hooks/useMediaQuery.ts`	Responsive breakpoint checks	Layout components
`useClickOutside`	`src/hooks/useClickOutside.ts`	Close on outside click	Dropdown, Modal
`useFocusTrap`	`src/hooks/useFocusTrap.ts`	Trap focus inside element	Modal, Drawer

Hook	File	Purpose	Used By
`useToast`	`src/hooks/useToast.ts`	Trigger toast notifications	Global
`usePermission`	`src/hooks/usePermission.ts`	Check user permissions	Auth-gated components
`{{HOOK_NAME}}`	`{{PATH}}`	`{{PURPOSE}}`	`{{CONSUMERS}}`

7. Third-Party Component Usage

Package	Version	Components Used	Wrapping Strategy
`{{@radix-ui/react-dialog}}`	`{{1.x}}`	Modal base	Wrapped in `src/components/ui/Modal` — custom styling
`{{@radix-ui/react-select}}`	`{{2.x}}`	Select base	Wrapped in `src/components/ui/Select`
`{{react-hook-form}}`	`{{7.x}}`	Form state	Used directly + FormField wrapper
`{{recharts}}`	`{{2.x}}`	Charts	Wrapped in `src/components/charts/`

Policy: Never use third-party components directly in feature code — always wrap in local component to control API surface and allow future swap.

8. Component Deprecation Process

```
Active → Deprecated (soft) → Deprecated (hard) → Removed
```

Stage	Action Required
Deprecated (soft)	Add `@deprecated` JSDoc comment, console warning in dev, migration guide in Storybook
Deprecated (hard)	TypeScript `@deprecated` annotation triggers IDE warning, added to removal milestone
Removed	Delete component, update CHANGELOG, run codemod if available

Deprecation notice minimum period: 2 sprint cycles before hard removal.

TODO: Link to CHANGELOG.md for tracked deprecations.

Approval

Role	Name	Date	Signature
Author			
Frontend Lead			
Design System Owner			

Design System Documentation

Design System Documentation

> **Project:** Drop — Fintech Payment App **Version:** 0.1.0 **Date:** 2026-02-23
> **Author:** John (AI Director, ALAI) **Status:** In Review **Reviewers:** Alem Bašić (CEO)

Document History

Version	Date	Author	Changes
0.1	2026-02-23	John	Initial draft from source code + brand guide analysis

---

1. Design Principles

Principle	Description
Clarity first	Every UI element must communicate its purpose without explanation. Amounts are large and legible. Actions are labeled.
Scandinavian minimal	Generous whitespace, no visual noise. Forest green + white — nothing extra.
Trust through transparency	Fees visible before every transaction. No hidden costs. PSD2 disclosures shown explicitly.
Mobile-first	The app is a mobile PWA. `max-w-sm` containers, `pb-24` for BottomNav clearance, touch targets ≥ 44×44px.
Accessible by default	WCAG AA compliance is a baseline requirement, not an optional enhancement.
Light mode only (Phase 1)	Background is always off-white (#FAFCF8) or white. Dark mode is Phase 2.

---

2. Color System

2.1 Primitive Palette (Raw Values)

```
/* Brand primitives — defined in brand/colors.css and globals.css */

/* Green scale */
--color-green-primary:   #0B6E35;   /* Forest Green — primary brand */
--color-green-dark:      #095C2C;   /* Hover/pressed state */
--color-green-light:     #E8F5E9;   /* Light green tint, badges */

/* Gold scale */
--color-gold-primary:    #D4A017;   /* Gold accent, logo arrow, premium elements */
--color-gold-light:      #FFF8E1;   /* Gold tint */

/* Neutral scale */
--color-neutral-0:       #FFFFFF;   /* White — cards, elevated surfaces */
--color-neutral-50:      #FAFCF8;   /* Off-white — page background */
--color-neutral-100:     #F9FAFB;   /* Input field backgrounds */
--color-neutral-200:     #F3F4F6;   /* Section backgrounds, shadcn secondary */
--color-neutral-300:     #E5E7EB;   /* Input borders, card borders, dividers */
--color-neutral-400:     #D1D5DB;   /* Horizontal rule dividers */
--color-neutral-500:     #9CA3AF;   /* Muted text, inactive nav items */
--color-neutral-600:     #6B7280;   /* Secondary text, placeholders */
--color-neutral-700:     #374151;   /* Dark text on light backgrounds */
--color-neutral-900:     #1A1A1A;   /* Near-black — headings, primary text */
--color-neutral-1000:    #000000;

/* Semantic */
--color-success:         #10B981;   /* Emerald — success, positive amounts */
--color-error:           #EF4444;   /* Red — error states, negative amounts */
--color-warning:         #D97706;   /* Warning — pending transactions */
--color-info:            #2563EB;   /* Info — PSD2 banners */
```

2.2 Semantic Tokens (Light Mode — from globals.css :root)

```
:root {
  /* shadcn/ui CSS variable tokens */
  --background:           #FAFCF8;
  --foreground:           #1A1A1A;
  --primary:              #0B6E35;
  --primary-foreground:   #FFFFFF;
  --secondary:            #F3F4F6;
  --secondary-foreground: #1A1A1A;
  --accent:               #F3F4F6;
  --accent-foreground:    #1A1A1A;
  --muted:                #F3F4F6;
  --muted-foreground:     #6B7280;
  --destructive:          #EF4444;
  --border:               #E5E7EB;
  --input:                #E5E7EB;
  --ring:                 #0B6E35;
  --radius:               0.75rem;

  /* Drop custom semantic tokens */
  --color-drop-primary:   #0B6E35;
  --color-drop-secondary: #D4A017;
  --color-drop-accent:    #10B981;
  --color-drop-dark:      #1A1A1A;
  --color-drop-light:     #FAFCF8;
  --color-drop-error:     #EF4444;
}
```

2.3 Semantic Tokens (Dark Mode)

```
/* Phase 2 — Dark mode TBD */
[data-theme="dark"] {
  /* TBD — requires design review */
  /* Drop is light-mode-first. Dark mode planned for Phase 2. */
}
```

2.4 Brand Gradient

```
.bg-gradient-brand {
  background: linear-gradient(135deg, #0B6E35 0%, #D4A017 100%);
}
```

2.5 Contrast Ratios (WCAG)

Pair	Text Color	Background	Ratio	WCAG AA (4.5:1)	WCAG AAA (7:1)
Primary text on page bg	`#1A1A1A`	`#FAFCF8`	~18.6:1	Pass	Pass
Secondary text on page bg	`#6B7280`	`#FAFCF8`	~4.7:1	Pass	Fail
Primary button text	`#FFFFFF`	`#0B6E35`	~5.0:1	Pass	Fail
Success text	`#10B981`	`#FFFFFF`	~3.0:1	Fail (large text only)	Fail
Error text	`#EF4444`	`#FFFFFF`	~3.9:1	Fail (large text only)	Fail
Muted text	`#9CA3AF`	`#FFFFFF`	~2.9:1	Fail	Fail

Action items: Success (#10B981), error (#EF4444), and muted (#9CA3AF) colors fail WCAG AA for small text. These are used for semantic indicators (amount colors, hints) — always paired with a non-color indicator. Full audit in `accessibility-audit.md`.

---

3. Typography

3.1 Font Families

Token	Value	Usage
`--font-fraunces`	`Fraunces, Georgia, "Times New Roman", serif`	Display/headings, logo wordmark, brand text
`--font-dm-sans`	`"DM Sans", -apple-system, BlinkMacSystemFont, "Segoe UI", sans-serif`	Body text, UI labels, inputs (default body font)
`--font-geist-mono`	`"Geist Mono", "JetBrains Mono", "Fira Code", "Courier New", monospace`	Code, monospace elements

Why these fonts:

- **Fraunces** — Variable serif with "wonky" optical size. Gives character and warmth. Differentiates Drop from every other fintech app (which all use sans-serif). Signals: "we are different, we are human."
- **DM Sans** — Clean geometric sans-serif. Readable at all screen sizes. Friendly without being playful. Good number legibility (critical for fintech).

3.2 Type Scale

Token	Size	Weight	Line Height	Letter Spacing	Usage
Display	56px	700 (Fraunces)	1.1	-0.02em	Hero headlines (landing page)
H1	40px	600 (Fraunces)	1.2	-0.01em	Page titles
H2	32px	600 (Fraunces)	1.3	-0.005em	Section headers
H3	24px	500 (Fraunces)	1.3	normal	Sub-sections
H4	20px	600 (DM Sans)	1.4	normal	Card titles
Body Large	18px	400 (DM Sans)	1.7	normal	Lead paragraphs
Body	16px	400 (DM Sans)	1.6	normal	Default text
Body Small	14px	400 (DM Sans)	1.5	normal	Secondary content, captions
Label	14px	500 (DM Sans)	1.4	0.01em	Form labels, UI labels
Caption	12px	500 (DM Sans)	1.4	0.02em	Timestamps, meta, hints

3.3 In-App Tailwind Typography Patterns

Context	Tailwind Class	Example
Logo wordmark	`font-[family-name:var(--font-fraunces)] text-3xl font-bold`	"drop" on login
Page heading	`text-xl font-semibold text-[#1A1A1A]`	Dashboard greeting
Section heading	`text-lg font-semibold text-[#1A1A1A]`	"Siste transaksjoner"
Form label	`text-sm font-medium text-[#1A1A1A]`	"E-post"
Body text	Inherited DM Sans	General content
Muted text	`text-sm text-[#6B7280]`	Descriptions, timestamps
Small text	`text-xs text-[#9CA3AF]`	Footers, hints
Balance (large)	`text-2xl font-bold text-[#1A1A1A]`	Balance display
Amount (list)	`text-sm font-semibold` + color	Transaction amounts

4. Spacing & Layout

4.1 Spacing Scale (4px Base Unit — Tailwind defaults)

Tailwind	Value	Usage
`p-1`	4px	Micro gaps
`p-2`	8px	Tight inline spacing
`p-3`	12px	Compact elements
`p-4`	16px	Default content spacing
`p-6`	24px	Card padding, section padding
`p-8`	32px	Large section gaps
`px-6`	24px	Page horizontal padding (standard)
`pb-24`	96px	Bottom padding (BottomNav clearance)

4.2 Page Structure Patterns

```
/* Standard authenticated page */
min-h-screen bg-[#FAFCF8]      /* Off-white background */
px-6 pb-24 pt-6               /* Padding: horizontal 24px, bottom 96px (nav), top 24px */

/* Login / onboarding (centered) */
min-h-screen bg-[#EEEEEE]      /* Slightly darker bg for auth pages */
max-w-sm mx-auto               /* 384px max width, centered */

/* Card pattern (standard) */
rounded-2xl bg-white p-6 shadow-sm

/* Card pattern (compact) */
rounded-xl bg-white p-4 shadow-sm
```

4.3 Responsive Breakpoints (Tailwind defaults)

Name	Min Width	Usage
Default	0px	Mobile — primary design target
`sm`	640px	Larger phones
`md`	768px	Tablets — landing page 2-column grid
`lg`	1024px	Desktop — landing page 3-column grid

Note: The app is mobile-first. `max-w-sm` containers keep content readable on wider screens.

4.4 Bottom Nav

The BottomNav is `fixed bottom-0 left-0 right-0` with `h-16 border-t bg-white`. All pages that include BottomNav must add `pb-24` to their content container.

---

5. Component Library

5.1 Custom Drop Components

Component	File	Status	Description
BottomNav	`components/bottom-nav.tsx`	Done	Fixed 5-tab navigation
DropLogo	`components/drop-logo.tsx`	Done	Forward-D SVG mark
DropWordmark	`components/drop-logo.tsx`	Done	"drop" in Fraunces
DropLogoFull	`components/drop-logo.tsx`	Done	Logo mark + wordmark
DropAppIcon	`components/drop-logo.tsx`	Done	App icon — rounded green square
CookieConsent	`components/cookie-consent.tsx`	Done	GDPR consent banner + modal
PrePaymentDisclosure	`components/pre-payment-disclosure.tsx`	Done	PSD2 pre-payment modal
PWARegister	`components/pwa-register.tsx`	Done	Service Worker registration
drop-icons	`components/drop-icons.tsx`	Done	9 custom fintech icons

5.2 shadcn/ui Primitive Components (Atoms)

Component	File	Radix Primitive	Status

Alert	`ui/alert.tsx`	— (div-based)	Done
Avatar	`ui/avatar.tsx`	@radix-ui/react-avatar	Done
Badge	`ui/badge.tsx`	— (cva variants)	Done
Button	`ui/button.tsx`	@radix-ui/react-slot	Done
Card	`ui/card.tsx`	— (div-based)	Done
Dialog	`ui/dialog.tsx`	@radix-ui/react-dialog	Done
Input	`ui/input.tsx`	— (input element)	Done
ScrollArea	`ui/scroll-area.tsx`	@radix-ui/react-scroll-area	Done
Select	`ui/select.tsx`	@radix-ui/react-select	Done
Separator	`ui/separator.tsx`	@radix-ui/react-separator	Done
Sheet	`ui/sheet.tsx`	@radix-ui/react-dialog	Done
Skeleton	`ui/skeleton.tsx`	— (pulse animation)	Done
Sonner	`ui/sonner.tsx`	sonner toast library	Done
Tabs	`ui/tabs.tsx`	@radix-ui/react-tabs	Done

5.3 Layout Components

Component	Description
Page wrapper	`min-h-screen bg-[#FAFCF8] px-6 pb-24 pt-6`
Auth wrapper	`min-h-screen flex items-center justify-center bg-[#EEEEEE]`
Card	`rounded-2xl bg-white p-6 shadow-sm`
Scroll area	`ScrollArea` from shadcn/ui for transaction lists

---

6. Iconography Guidelines

Item	Standard
Primary library	`lucide-react`
Delivery	Inline SVG via React component
Sizes	`h-4 w-4` (inline/small), `h-5 w-5` (nav/buttons), `h-6 w-6` (feature icons)
Stroke width	1.5px at 24px (Lucide default)
Color	`currentColor` — inherits from parent text color

Item	Standard
Custom icons	`components/drop-icons.tsx` — 9 domain-specific icons
Social auth	Inline SVG — BankID (green rounded rect, "ID"), Vipps (orange circle, "V")

Custom Drop Icons:

Export	Description
`IconSendMoney`	Arrow going up-right from horizontal line
`IconQrScan`	QR code frame with scan corners
`IconVirtualCard`	Credit card outline
`IconShield`	Shield with checkmark
`IconFastTransfer`	Lightning bolt
`IconCorridors`	Globe with meridians
`IconWallet`	Wallet outline (unused — no wallet in pass-through model)
`IconHistory`	Clock with arrow
`IconTopUp`	Plus inside circle (unused — no top-up in pass-through model)

Accessibility rule: Icons conveying meaning must have `aria-label`. Decorative icons: `aria-hidden="true"`.

---

7. Motion & Animation Standards

7.1 Duration Tokens

Usage	Value
Button hover	`transition-colors` (CSS default ~150ms)
Focus states	`transition-colors` on border/ring
Loading states	Text replacement ("Logger inn...", "Sender...") or Skeleton
Scanner frame	`animate-pulse` on scan page corners
Complex animations	None in Phase 1 — simplicity-first

7.2 Easing Tokens

Token	Value	Usage
`--ease-default`	`cubic-bezier(0.4, 0, 0.2, 1)`	General UI transitions
`--ease-enter`	`cubic-bezier(0, 0, 0.2, 1)`	Elements entering
`--ease-exit`	`cubic-bezier(0.4, 0, 1, 1)`	Elements leaving

7.3 Reduced Motion

```
@media (prefers-reduced-motion: reduce) {
  *, *::before, *::after {
    animation-duration: 0.01ms !important;
    transition-duration: 0.01ms !important;
  }
}
```

Rule: Every animated component MUST respect `prefers-reduced-motion`.

---

8. Accessibility Requirements Per Component

Requirement	Buttons	Inputs	Modals	BottomNav	Transaction List
Keyboard accessible	Required	Required	Required	Required	Required
Focus visible	`ring-[#0B6E35]`	`focus:border-[#0B6E35]`	Focus trap	Required	Required
ARIA role	`button`	`textbox`	`dialog`	`nav`	`list` / `listitem`
Screen reader label	Visible text or `aria-label`	`<label>` associated	`aria-labelledby`	`aria-label="Navigasjon"`	—
Error state	Text message	`text-[#EF4444]`	—	—	—
Touch target	`h-12` (48px)	`h-11` (44px)	—	`h-16` (64px)	`min-h-[44px]`

---

9. Design Token Format

9.1 CSS Custom Properties (Source of Truth)

```
/* src/app/globals.css */
:root {
  --color-drop-primary:   #0B6E35;
  --color-drop-secondary: #D4A017;
  --color-drop-accent:    #10B981;
  --color-drop-dark:      #1A1A1A;
  --color-drop-light:     #FAFCF8;
  --color-drop-error:     #EF4444;
  /* shadcn/ui tokens... */
}
```

9.2 Tailwind Config Extension

```
// tailwind.config.ts
export default {
  theme: {
    extend: {
      colors: {
        drop: {
          primary:   '#0B6E35',
          secondary: '#D4A017',
          accent:    '#10B981',
          dark:      '#1A1A1A',
          light:     '#FAFCF8',
          error:     '#EF4444',
        }
      },
      fontFamily: {
        fraunces: ['var(--font-fraunces)', 'Georgia', 'serif'],
        sans:     ['var(--font-dm-sans)', 'system-ui', 'sans-serif'],
        mono:     ['var(--font-geist-mono)', 'monospace'],
      }
    }
  }
}
```

9.3 JavaScript/TypeScript Constants (for charting/canvas)

```
// For use in non-CSS contexts (future charts, canvas)
export const dropColors = {
  primary:   '#0B6E35',
  secondary: '#D4A017',
  accent:    '#10B981',
  dark:      '#1A1A1A',
  light:     '#FAFCF8',
  error:     '#EF4444',
} as const;
```

Token update process: Figma → update `globals.css` CSS variables → update Tailwind config → PR review.

---

10. Component Code Patterns

Primary Button

```
<Button className="h-12 w-full rounded-xl bg-[#0B6E35] text-sm font-semibold text-white
shadow-sm hover:bg-[#095C2C]">
  Send penger
</Button>
```

Outline Button

```
<button className="flex h-12 flex-1 items-center justify-center gap-2 rounded-xl border
border-[#E5E7EB] bg-white text-sm font-medium text-[#1A1A1A] hover:bg-[#F9FAFB]">
  Avbryt
</button>
```

Text Input with Left Icon

```
<div className="relative">
  <Mail className="absolute left-3 top-1/2 h-4 w-4 -translate-y-1/2 text-[#6B7280]" />
  <input
    className="h-11 w-full rounded-lg border border-[#E5E7EB] bg-[#F9FAFB] pl-10 pr-3 text-sm outline-none focus:border-[#0B6E35] focus:ring-1 focus:ring-[#0B6E35]"
    type="email"
    placeholder="din@epost.no"
  />
</div>
```

Error Message

```
<p className="rounded-md bg-[#EF4444]/10 p-2 text-sm text-[#EF4444]">
  {error}
</p>
```

Primary Badge

```
<span className="rounded-full bg-[#0B6E35]/10 px-2 py-0.5 text-xs font-medium text-[#0B6E35]">
  Primær konto
</span>
```

Avatar (Initials)

```
<div className="flex h-10 w-10 items-center justify-center rounded-full bg-[#0B6E35] text-sm font-bold text-white">
  {initials}
</div>
```

Transaction Amount (Positive / Negative)

```
// Positive (received)
<span className="text-sm font-semibold text-[#10B981]">+1 250,00 NOK</span>

// Negative (sent)
<span className="text-sm font-semibold text-[#EF4444]">-500,00 NOK</span>
```

Approval

Role	Name	Date	Signature
Author	John (AI Director)	2026-02-23	
Lead Designer			
Frontend Lead			
Accessibility Reviewer			

State Management Architecture

State Management Architecture

> **Project:** Drop — Fintech Payment App **Version:** 0.1.0 **Date:** 2026-02-23
> **Author:** John (AI Director, ALAI) **Status:** In Review **Reviewers:** Alem Bašić (CEO)

Document History

Version	Date	Author	Changes
0.1	2026-02-23	John	Initial draft from source code analysis

---

1. State Architecture Overview

Drop uses a deliberately simple, lightweight state management approach. There are no global state libraries (Redux, Zustand, Jotai, etc.). All state is local component state.

Layer	Mechanism	Scope
Auth / user state	`useAuth()` custom hook	Per-component (fetches `/api/auth/me` on each mount)
Feature flags	`useFeatureFlag()` hook	Per-component (reads `NEXT_PUBLIC_FF_*` env vars)
Page / API data	`useState` + `useEffect` fetch	Component-local
Form state	`useState` per field	Component-local
UI state (modals, tabs, steps)	`useState`	Component-local
Navigation	Next.js `useRouter` / `usePathname`	Framework-provided
Auth token (web)	httpOnly cookie	Server-set, never in JS
Auth token (mobile)	Bearer token in-memory (`let token`) + `AsyncStorage`	Module-level in `lib/api.js`

Guiding principle: Keep it simple. No global state = no accidental shared state bugs. API data is fetched fresh on each page mount. User authentication is the only cross-cutting concern, handled by the `useAuth()` hook.

---

2. Data Flow Diagram

```
flowchart TD
    User["User Interaction"] --> Component["Component"]

    Component -->|"Auth check (every mount)"| AuthHook["useAuth() Hook\nGET /api/auth/me"]
    Component -->|"Data fetch (useEffect)"| LocalState["useState + useEffect\nLocal component state"]
    Component -->|"Form input"| FormState["useState per field\nComponent-local"]
    Component -->|"Navigate"| Router["Next.js Router\nuseRouter / usePathname"]
    Component -->|"Feature check"| FlagHook["useFeatureFlag()\nReads NEXT_PUBLIC_FF_*"]

    AuthHook -->|"cookie auth"| BFF["BFF API Routes\n/api/auth/me"]
    LocalState -->|"fetch with credentials"| BFF
    BFF -->|"User object + data"| Component

    FlagHook -->|"env var read"| EnvVars["process.env\nNEXT_PUBLIC_FF_*"]
```

---

3. State Categories

3.1 Auth State — `useAuth()` Hook

File: `src/lib/use-auth.ts`

Interface:

```
function useAuth(redirectIfUnauthenticated?: boolean): {
  user: User | null;
  loading: boolean;
  logout: () => Promise<void>;
  refreshUser: () => Promise<void>;
}
```

```
// Default: redirectIfUnauthenticated = true
```

User Model:

```
interface User {
  id: string;
  email: string;
  firstName: string;
  lastName: string;
  totalBalance: number;
  bankAccounts: BankAccount[];
  kycStatus: string;
}

interface BankAccount {
  id: string;
  bankName: string;
  accountNumber: string;
  balance: number;
  currency: string;
  isPrimary: boolean;
}
```

Behavior:

1. On mount: fetches `GET /api/auth/me` with `credentials: "include"` (httpOnly cookie auth)
2. If 401 and `redirectIfUnauthenticated` is true: redirects to `/login`
3. `logout()`: calls `POST /api/auth/logout`, clears cookie server-side, redirects to `/login`
4. `refreshUser()`: re-fetches `/api/auth/me` to update user state

Auth flow:

```
Login page → POST /api/auth/login → httpOnly cookie set → router.push("/dashboard")
Dashboard → useAuth() → GET /api/auth/me → User object
Logout → POST /api/auth/logout → cookie cleared → redirect /login
```

Usage patterns:

```
// Standard protected page (redirects if unauthenticated)
const { user, loading } = useAuth();
if (loading) return <Skeleton />;
// user is guaranteed non-null after loading
```

```
// Page that checks auth without redirect
const { user } = useAuth(false);
```

Pages using this hook (data source):

- `accounts/page.tsx` — reads `user.bankAccounts` (no separate fetch)
- `profile/page.tsx` — reads `user.firstName`, `user.lastName`, `user.email`

3.2 Feature Flags — `useFeatureFlag()` Hook

File: `src/lib/feature-flags.ts`

Available Flags:

Flag Name	Default	Used In
`virtualCards`	`false`	cards page (gate — shows "not available" if false)
`physicalCards`	`false`	cards page (order physical card option)
`cardDetails`	`false`	cards page (show full card details)
`cardFreeze`	`false`	cards page (freeze/unfreeze)
`cardPin`	`false`	cards page (change PIN)
`spendingLimits`	`false`	cards page (spending limits)
`notifications`	`true`	notification features
`merchantDashboard`	`true`	merchant page (gate)

Environment Variable Pattern:

```
NEXT_PUBLIC_FF_VIRTUAL_CARDS=true
NEXT_PUBLIC_FF_PHYSICAL_CARDS=false
NEXT_PUBLIC_FF_NOTIFICATIONS=true
NEXT_PUBLIC_FF_MERCHANT_DASHBOARD=true
```

Convention: `NEXT_PUBLIC_FF_` + `SCREAMING_SNAKE_CASE` version of flag name.

API:

```
// Server-side (API routes / middleware)
isEnabled(flagName: string): boolean
```

```
getAllFlags(): Record<string, boolean>
featureGate(flagName: string): middleware  // Returns 404 if flag disabled

// Client-side (React hooks)
useFeatureFlag(flagName: string): boolean
useFeatureFlags(): Record<string, boolean>
```

Usage pattern:

```
// Page-level gate
const cardsEnabled = useFeatureFlag("virtualCards");
if (!cardsEnabled) return <div>Feature ikke tilgjengelig</div>;

// Conditional rendering
const physicalEnabled = useFeatureFlag("physicalCards");
{physicalEnabled && <OrderPhysicalCard />}
```

3.3 Page Data — `useState` + `useEffect` Fetch (Pattern 1)

Most pages fetch data in a `useEffect` on mount. No SWR, React Query, or other caching library is used.

```
const [data, setData] = useState([]);
const [loading, setLoading] = useState(true);

useEffect(() => {
  fetch("/api/endpoint", { credentials: "include" })
    .then(res => res.json())
    .then(json => setData(json.data))
    .catch(() => {})  // Silent error — empty state
    .finally(() => setLoading(false));
}, []);
```

Pages using this pattern:

- `dashboard/page.tsx` — fetches `GET /api/transactions?limit=10`
- `transactions/page.tsx` — fetches `GET /api/transactions?type={filter}&limit=50`
- `send/page.tsx` — fetches `GET /api/recipients` and `GET /api/rates`
- `notifications/page.tsx` — fetches `GET /api/notifications`

- `profile/notifications/page.tsx` — fetches `GET /api/settings`
- `profile/language/page.tsx` — fetches `GET /api/settings`
- `merchant/page.tsx` — fetches `/api/merchants/dashboard`, `/api/merchants/transactions`, `/api/merchants/qr`
- `cards/page.tsx` — fetches `GET /api/cards` (feature-flagged)

3.4 Form State — `useState` Per Field (Pattern 3)

Form pages use async handlers that POST data and handle success/error states. No form library.

```
const [email, setEmail] = useState("");
const [password, setPassword] = useState("");
const [error, setError] = useState("");
const [loading, setLoading] = useState(false);

const handleSubmit = async () => {
  setLoading(true);
  setError("");
  const res = await fetch("/api/auth/login", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    credentials: "include",
    body: JSON.stringify({ email, password }),
  });
  if (res.ok) {
    router.push("/dashboard");
  } else {
    const data = await res.json();
    setError(data.message || "Noe gikk galt");
  }
  setLoading(false);
};
```

Pages using this pattern:

- `login/page.tsx` — POST `/api/auth/login`
- `register/page.tsx` — POST `/api/auth/register` (4-step: info, OTP, PIN, success)
- `send/page.tsx` — POST `/api/transactions/remittance`
- `scan/page.tsx` — POST `/api/transactions/qr-payment`

- `complaints/page.tsx` — POST `/api/complaints`
- `withdrawal/page.tsx` — POST (withdrawal request)
- `cards/page.tsx` — POST/PATCH/DELETE `/api/cards/*` (feature-flagged)

3.5 Filter-Driven Refetch (Pattern 4)

Transactions and notification pages refetch when filter changes via `useEffect` dependency.

```
const [filter, setFilter] = useState("all");

useEffect(() => {
  fetch(`/api/transactions?type=${filter}&limit=50`, { credentials: "include" })
    .then(res => res.json())
    .then(json => setData(json.transactions))
    .catch(() => {})
    .finally(() => setLoading(false));
}, [filter]);
```

Pages using this pattern:

- `transactions/page.tsx` — filters: "all", "remittance", "qr_payment"
- `notifications/page.tsx` — auto-marks read via PATCH on mount

---

4. API Routes (BFF Layer)

All API routes are under `src/app/api/`. The frontend calls these endpoints via the Next.js BFF (which translates httpOnly cookie auth to Bearer token for the Hono backend).

Endpoint	Method	Called From
`/api/auth/login`	POST	login page
`/api/auth/logout`	POST	useAuth hook, profile page
`/api/auth/me`	GET	useAuth hook (every protected page mount)
`/api/auth/register`	POST	register page
`/api/transactions`	GET	dashboard, transactions pages
`/api/transactions/remittance`	POST	send page
`/api/transactions/qr-payment`	POST	scan page

Endpoint	Method	Called From
`/api/recipients`	GET	send page
`/api/rates`	GET	send page
`/api/notifications`	GET	notifications page
`/api/notifications`	PATCH	notifications page (mark read)
`/api/settings`	GET	profile/notifications, profile/language
`/api/settings`	PATCH	profile/notifications, profile/language
`/api/complaints`	POST	complaints page
`/api/consents`	POST	cookie-consent component
`/api/cards`	GET/POST	cards page (feature-flagged)
`/api/cards/{id}`	PATCH	cards page (freeze/unfreeze, feature-flagged)
`/api/cards/{id}`	DELETE	cards page (delete card, feature-flagged)
`/api/merchants/dashboard`	GET	merchant page
`/api/merchants/transactions`	GET	merchant page
`/api/merchants/qr`	GET	merchant page

---

5. Mobile State Management

File: `src/drop-mobile/lib/api.js`

The mobile app uses a different auth mechanism — Bearer token instead of httpOnly cookie.

```
// Token stored at module level (in-memory)
let token = null;

export const setToken = (t) => { token = t; };
export const getToken = () => token;

// All requests include auth header
const request = async (endpoint, options = {}) => {
  const headers = {
    "Content-Type": "application/json",
    ...(token && { Authorization: `Bearer ${token}` }),
    ...options.headers,
```

```
  };
  // ...
};
```

Token persistence: On login, token stored in `AsyncStorage` for 7-day lifetime. On app launch, token loaded from `AsyncStorage` and set via `setToken()`.

Mobile state patterns:

- All screens use `useState` (same as web)
- Auth data read from `api.getMe()` on dashboard mount
- Pull-to-refresh: `RefreshControl` on `FlatList` components
- No global state library — same philosophy as web

---

6. Persistent State

Data	Storage	Notes
Auth session (web)	httpOnly cookie	Server-set, 7-day lifetime, never in JS
Auth token (mobile)	Bearer token, `AsyncStorage`	7-day lifetime, device-bound
Cookie preferences	`localStorage`	CookieConsent component
Language preference	Server-side (`/api/settings`)	Synced on login
Push notification prefs	Server-side (`/api/settings`)	Synced on page load

RULE: Auth tokens NEVER in localStorage. httpOnly cookies on web, `AsyncStorage` (module-level) on mobile.

---

7. State Debugging

Tool	Usage	Enabled In
React DevTools	Component state tree inspection	Dev only
`__DEV__` flag	Enables demo credentials on login	Dev only
Network tab	Inspect `/api/auth/me` response	Dev only
Console logs	API client logs (mobile)	Dev only

No state management devtools — not needed without a global store.

---

8. Performance Considerations

No Unnecessary Re-renders

- Each `useEffect` has explicit dependency arrays
- `useState` updates are batched by React 19
- No derived state that needs memoization in current implementation

Fresh Data on Navigation

- `useAuth()` re-fetches `/api/auth/me` on every page mount — ensures fresh user data
- Page data re-fetches on every mount — no stale cache issues
- Trade-off: more network requests, but always fresh data for financial app accuracy

Future Optimization Path (if needed)

1. Add `SWR` or `TanStack Query` for caching with stale-while-revalidate
2. Persist query cache across navigation (currently re-fetches on each visit)
3. Optimistic updates for settings toggles (partially implemented in notification settings)

Approval

Role	Name	Date	Signature
Author	John (AI Director)	2026-02-23	
Frontend Lead			
Tech Lead			

Landing Pages

Drop Frontend — Landing Pages

> Covers the static marketing site (`landing/`) and the standalone web prototype (`src/drop-web/`).

Marketing Landing Page

File: `landing/index.html` (~646 lines)

Tech Stack

- Pure HTML/CSS/JS (no framework)
- Fonts: Fraunces + DM Sans (Google Fonts CDN)
- CSS custom properties matching brand system
- Responsive (mobile-first)

Page Sections

#	Section	Description
1	**Nav**	DropLogoFull SVG + links (Tjenester, Priser, Om oss) + Logg inn / Kom i gang buttons
2	**Hero**	Headline "Enklere betalinger for alle i Norge", subtext, 2 CTA buttons, phone mockup placeholder
3	**Trust Bar**	3 stats: "0.5% gebyr", "<2 timer leveringstid", "30+ land"

#	Section	Description
4	**Features**	3 cards: Send penger (globe icon), QR-betaling (QR icon), Virtuelt kort (card icon)
5	**Calculator**	Transfer calculator with amount input, country dropdown, live conversion display
6	**How It Works**	4 steps: Last ned → Koble bank → Velg mottaker → Send
7	**Social Proof / Early Access**	Waitlist signup form with email input
8	**CTA**	Final call-to-action with app store buttons (placeholder)
9	**Footer**	Logo, company info (ALAI Holding AS, Org.nr 932 516 136), nav links to sub-pages

CSS Animations

```
@keyframes fadeUp    /* Scroll-triggered fade in from below */
@keyframes float     /* Gentle floating on phone mockup */
@keyframes shimmer   /* Shimmer effect on CTA buttons */
@keyframes pulse     /* Subtle pulse on trust badges */
```

Colors Used

```
--drop-green: #0B6E35;
--drop-gold: #D4A017;
--drop-dark: #1A1A1A;
--drop-light: #FAFCF8;
--drop-gray: #6B7280;
--drop-border: #E5E7EB;
```

Responsive Breakpoints

- Mobile: default (single column)
- Tablet: `@media (min-width: 768px)` — 2-column grids
- Desktop: `@media (min-width: 1024px)` — 3-column grids, wider hero

Sub-Pages

Directory: `landing/pages/`

12 static HTML pages linked from the footer:

File	Route (relative)	Content
`cookies.html`	/pages/cookies	Cookie policy
`karriere.html`	/pages/karriere	Careers page
`kontakt.html`	/pages/kontakt	Contact form / info
`lisenser.html`	/pages/lisenser	Licenses and regulatory info
`om-drop.html`	/pages/om-drop	About Drop
`personvern.html`	/pages/personvern	Privacy policy
`presse.html`	/pages/presse	Press / media kit
`priser.html`	/pages/priser	Pricing
`qr-betaling.html`	/pages/qr-betaling	QR payment feature page
`send-penger.html`	/pages/send-penger	Send money feature page
`sikkerhet.html`	/pages/sikkerhet	Security overview
`vilkar.html`	/pages/vilkar	Terms and conditions

Waitlist Script

File: `landing/pages/waitlist.js`

- Handles email collection for early access signup
- Connected to the waitlist form in the landing page hero/social proof section

---

Standalone Web Prototype (`drop-web`)

File: `src/drop-web/index.html` (~1305 lines)

Overview

This is an **older/alternative prototype** with a different design system. It predates the main Next.js app and serves as a standalone demo.

Key Differences from Main App

Aspect	Main App (`drop-app`)	Web Prototype (`drop-web`)
Framework	Next.js (React)	Vanilla JS (single file)
Theme	Light (#FAFCF8 bg)	Dark (#0D1117 bg)
Primary green	`#0B6E35` (Forest Green)	`#00C853` (Bright Green)
Fonts	Fraunces + DM Sans	Outfit + Plus Jakarta Sans
API port	localhost:3000	localhost:3001
Routing	File-based (App Router)	JS function `showScreen()`

Screens

Screen	Description
Welcome	Dark bg, "drop." wordmark, "Penger uten grenser" tagline, login/register buttons
Login	Email + password form, BankID/Vipps buttons, validation
App (Dashboard)	4 tabs: Hjem, Send, QR, Profil
Hjem tab	Balance card (dark green gradient), recent transactions
Send tab	Recipient selection, amount input, country/currency picker
QR tab	Scanner placeholder with merchant list
Profil tab	User info, contacts, settings, logout

Architecture

- Single HTML file with embedded CSS and JS
- `showScreen(name)` function handles "routing" by showing/hiding divs
- `switchTab(name)` for tab navigation within the app screen
- API calls to `localhost:3001/api` with Bearer token auth
- Functions: `doLogin()`, `doRegister()`, `doSend()`, `doQRPay()`, `renderTransactions()`

Status

This prototype appears to be an **earlier iteration** or **demo showcase**. The main `drop-app` (Next.js) is the production codebase. The `drop-web` prototype uses different colors, fonts, and a dark theme that does not match the current brand guide.