Financial Institution Portal — Full Frontend Specification (Help2Home)

Audience: Frontend developers building the standalone Financial Institution Portal and the partner-bank SSO/embedded experience. Written in plain, simple English with step-by-step UI requirements, flows, data needs, error states, and acceptance criteria.

High-level summary

This document describes everything the frontend developer needs to build for the Financial Institution Portal (FIP). The FIP is two things:

1. **A standalone portal** used by partner banks/financial institutions to review, approve, disburse, and monitor loans originated through Help2Home.
2. **An embedded SSO experience** the tenant reaches from Help2Home when they choose a partner institution during a loan application. The embedded view must accept pre-filled details from Help2Home via the API.

We support two setup modes:

- Single partner (one institution) — the portal is pre-configured for that partner.
- Multiple partners — when tenant chooses a bank from a list, they are redirected or embedded into that bank's portal instance.

This spec covers all pages, components, states, integrations, and developer notes. Build everything responsive and accessible.

Table of contents

---

1. Authentication & entry points

Entry points

- **Help2Home SSO link** — Tenant clicks **Apply for loan** on a property page -> a modal lists partner institutions -> tenant chooses one -> **Launch partner** redirects or opens embedded iframe (or new tab, depending on bank policy). The iframe/session receives a signed token and pre-filled data. If the tenant did not have an account with that bank before now, they will be required to open new account, by filling the bank opening account digital form, we can use their details to auto-populate to fill some data while complete the form, and will be required to wait (depending on the bank process) until an account/profile is created for them in the bank system, then they can login to their account dashboard in that bank. This will also allow them to use the account details in the future for other things.
- **Direct bank staff login** — Bank staff visit the FIP login page and authenticate with username/password + 2FA. Optionally support SAML/OAuth for enterprise customers.
- **Admin/OAuth** — Bank admins will have separate login and role-based UI controls.

Authentication flows (frontend responsibilities)

- Build a secure login page with fields: email/username, password, OTP input for 2FA. Follow standard password UX (show/hide, strength meter optional).
- Support **SSO from Help2Home**: implement a route `/sso/launch` that accepts a signed JWT query param or POST. On receiving the token, validate with backend (frontend sends token to API endpoint to exchange for session) and create session cookies (backend handles actual session creation). Display a branded splash while handshake occurs.
- Idle timeout modal and automatic logout – show a 2-minute warning and allow user to extend session.

2. Tenant flow (when tenant chooses a financial institution)

2.1 Partner selection modal (on Help2Home side)

- A list of partner institutions with name, logo, interest rate range, max term, and a short note ("fast verification", "requires KYC video").
- When tenant clicks a partner, Help2Home calls the FIP `prelaunch` API with tenant id and application id; FIP returns a signed token and redirect URL.

  Frontend for Help2Home will implement this modal. For FIP, implement a route that can accept the token and show a confirmation.

2.2 Embedded launch (bank side)

- The FIP must accept pre-filled fields (name, email, phone, property id, requested amount, monthly rent, deposit, application id) via the secure token.
- First screen after SSO: **Loan Summary** — show pre-filled details and a clear CTA: "Start verification".

- Tenant proceeds through the institution's verification journey (KYC, documents upload, credit consent). Each step must show progress (e.g. 4/6 steps) and allow return to Help2Home after completion.

3. Bank Portal UI — pages and components

The portal should be organized into clear sections. For each page list components and behaviors.

3.1 Main layout

- Top nav: Bank logo, institution name, user avatar, notifications bell, quick links.
- Left side nav (collapsible): Dashboard, Loan Applications, Accounts, Disbursements, Monitoring, Reports, API & Webhooks, Settings, Support.
- Each top-level page has a filter bar and a main content area.

3.2 Dashboard (Landing page after login)

Purpose: quick overview of portfolio, new applications, overdue amounts.

Widgets:

- KPI cards: Total outstanding, New applications (24h), Approvals today, Delinquencies (%), Disbursed this month.
- Recent applications list (table) with quick actions (View / Approve / Reject / Request docs).
- Chart: portfolio balance over time (frontend can render chart using data API — provide placeholder while loading).
- Alerts: pending webhooks, reconciliation mismatches.

Interactions:

- Clicking a KPI or chart segment drills into the relevant report.

3.3 Loan Applications page (primary workflow)

List / Table layout Columns: Application ID, Tenant name, Property, Requested amount, Term, Status, Credit score, Assigned officer, Received date, Actions.

Filters: status (New, In review, Approved, Rejected, Disbursed), date range, assigned officer, branch, min/max amount, property ID.

Row actions: View (opens application panel), Assign, Approve, Reject, Request documents, Start disbursement.

Application detail view (panel/modal/full page) Sections inside:

1. **Summary header** — Application ID, status badge, requested amount, term, predicted monthly repayment, property snapshot (image + address).
2. **Tenant details** — personal data (name, DOB, phone, email), pre-filled flags (verified by Help2Home), KYC status.

3. **Documents** — list of uploaded docs (ID, payslips, tenancy agreement). Each doc: thumbnail, filename, uploaded on, view/download, mark as verified.
4. **Credit & Risk** — credit report snapshot, score, red flags, automated risk recommendation (approve/reject/needs manual). Show source and timestamp.
5. **Repayment schedule** — installment breakdown, first due date, fees.
6. **Action bar** — Approve (with conditions modal), Reject (select reason), Request more docs (pre-filled message template), Assign to officer.

Approve modal

- Allow officer to set approved amount (can be lower), term, interest rate override, and add internal comments.
- If approval triggers disbursement, offer checkbox: "Start disbursement now".

3.4 Account Management page

- Create or link tenant bank accounts.
- Fields: account number, bank code, account holder name, account type, verification status.
- Action: verify account (call payment-network microflow). Show verification progress and result.
- Option to issue virtual card/ATM or flag for checkbook issuance (simulated flow — backend handles issuing).

3.5 Disbursement page

Purpose: review disbursement details and approve transfer to landlord’s account.

- Disbursement queue table: Application ID, Beneficiary (landlord name), Beneficiary account, amount, requested by, status, actions.
- Disbursement detail: beneficiary KYC, payment reference, fees breakdown, transfer method (instant/ACH), and final approve button.
- On approve: show confirmation, trigger webhook to Help2Home, and show receipt.

3.6 Loan Monitoring & Collections

- Master list of active loans with status: current, upcoming due, overdue.
- Drill into loan: repayment history, next due date, contact history, insurance status (if enabled), late-fee calculations.
- Collections tools: send reminder SMS/email (templated), escalate to recovery, freeze account.

3.7 Reports (Analytics)

- Pre-built reports: Portfolio performance, Delinquency breakdown, Origination funnel (applications → approvals → disbursements), Average ticket size, IRR.
- Filters and export (CSV, PDF). Charts should be interactive and allow date-range zoom.

3.8 Reconciliation & API Settings

- Webhook management UI: list of endpoints, secret keys, last delivery status, resend button.
- API credentials: create/regenerate API keys, set scopes (read-only / write), and IP allowlist.
- Logs: show webhook deliveries with payload preview and response code.

3.9 Settings & Roles

- Role-based access control: Admin, Loan Officer, Reconciliation Officer, Collections, Read-only Auditor.
- UI for creating users, assigning roles, and setting permissions.

3.10 Support & Audit Trail

- Support chat/ticket list (internal) where bank staff can talk to Help2Home support.
- Audit trail per application: every action (approve, reject, doc verified) must be shown with timestamp and actor.

4. Data exchange & API / Webhook notes (what frontend needs to call)

Backend provides endpoints. Frontend must implement calls, show loading states, and handle errors.

Minimum API needs (examples):

- `POST /sso/launch` — exchange SSO token for session (payload: sso_token).
- `GET /applications?filters` — fetch list of loan applications.
- `GET /applications/{id}` — fetch full application detail.
- `POST /applications/{id}/approve` — approve with payload (amount, term, rate, comment).
- `POST /applications/{id}/reject` — reject with reason.
- `POST /applications/{id}/assign` — assign to officer.
- `GET /accounts/{tenant_id}` — get linked accounts.
- `POST /disbursements/{id}/approve` — approve disbursement.
- `GET /reports/...` — fetch report data.
- `GET /webhooks` and `POST /webhooks/resend` — manage webhooks.

Frontend must:

- Use authenticated HTTP headers (Authorization: Bearer) provided by backend.
- Gracefully show partial data when some endpoints fail (and show retry CTA).
- Display timestamps in ISO 8601 and user's local timezone.

5. UI states, validation and error handling

Loading states

- Show skeletons for tables, cards, and forms while awaiting API responses.

Empty states

- Provide friendly empty state illustrations and CTAs (e.g., "No new applications. Create a test application" for staging).

Validation rules (frontend):

- Numeric fields: enforce min/max, show inline errors.
- Required fields: show which ones are required with clear messages.
- File uploads: limit types (pdf, jpg, png), max size (e.g., 10 MB), and show preview/thumb.

Network & API errors

- Show toast or inline error with clear action (Retry / Contact support).
- On critical failures (session expired), redirect to login with message.

6. Notifications, messaging & audit trail

Notifications

- System-wide notifications icon with unread count.
- Types: info (webhook failed), action needed (new application), system alert (reconciliation mismatch).
- Clicking a notification navigates to the relevant page.

Messages to tenants (via Help2Home)

- Frontend must provide templated messages when requesting docs or sending approval/rejection notices. Messages should appear in Help2Home message history via webhook.

Audit trail

- All actions must create logs visible in application detail. Show who did what and when.

7. Security & privacy front-end checklist

- Use secure cookies and ensure `SameSite` and `Secure` flags are set (backend responsibility but frontend must avoid insecure storage of tokens).
- Never store tokens in localStorage. Use session cookies or secure storage recommended by backend.
- Sanitize and escape any user-input displayed (avoid injection risk).
- Mask sensitive numbers (show last 4 digits of account numbers by default). Provide a "reveal" button that logs the reveal action in the audit trail.
- For file uploads, show file scanning status (if backend performs virus scan). Do not assume uploads are safe.

8. Accessibility & responsive behavior

- All interactive elements must be keyboard accessible (tab order, aria-labels).
- Provide meaningful alt text on images and logos.
- Color contrast must meet WCAG AA.
- Pages must be responsive: wide multi-column layout on desktop; stacked single-column on tablet/phone.

9. Testing, staging, and developer acceptance criteria

Unit & integration tests

- Components: login page, application table, application detail, approve modal, document viewer.
- API error handling: simulate 401, 500, and timeouts.

Acceptance criteria (example)

- SSO: tenant clicks partner from Help2Home and lands in FIP with pre-filled data visible within 3 seconds.
- Approve flow: officer can approve and trigger disbursement; webhook delivered to Help2Home with 200 OK status recorded.
- Disbursement: approving a disbursement marks it as "In transit" and shows a receipt with reference ID.

10. Appendix — Example screens & micro-interactions

Loan Application Detail micro-interactions

- When a document is marked verified, show a small green animation and append "verified by {user} at {time}" to the audit trail.
- If credit score is older than 7 days, show a warning banner: "Credit report is older than 7 days — re-run report." with a "Re-run" CTA.

Disbursement confirmation

- On successful disbursement approve: show a modal that contains transfer reference, expected settlement time, and a button "Send receipt to Help2Home now".

Onboarding tip

- First-time users see a short guided walkthrough overlay that points to Dashboard KPIs, Loan Applications, and Webhook settings.

Final developer notes

- Keep UI components modular and reusable — `Table`, `FilterBar`, `DocumentViewer`, `KpiCard`, `ApprovalModal`, `Toast`.
- Use optimistic UI updates sparingly and only where rollback is easy.
- Coordinate with backend teams to confirm API contract (field names, status enums). If the frontend has assumptions, document them in the API integration notes.

If you want, I can now:

- create React component stubs for the top-level pages (single-file components styled with Tailwind), or
- make a concise checklist for QA to validate the portal.

Tell me which one to generate next and I will add it to the canvas.