Phase 4.1: RBAC UI Implementation - CareLinkAI

Status: ✅ In Progress
Date: December 9, 2025
Branch: main

Overview

Phase 4.1 implements permission-based UI components across the CareLinkAI application, building on top of the Phase 4 RBAC core system. This phase focuses on making the UI respect user permissions and roles, providing appropriate visual feedback for different user types.

Implementation Status

✅ Completed Components

1. Residents List Page (`src/app/operator/residents/page.tsx`)

- ✅ Created `ResidentsListActions.tsx` with permission-guarded components:
- `NewResidentButton` - Guarded with `PERMISSIONS.RESIDENTS_CREATE`
- `ExportResidentsButton` - Guarded with `PERMISSIONS.RESIDENTS_VIEW`
- `EditResidentButton` - Guarded with `PERMISSIONS.RESIDENTS_UPDATE`
- `ResidentRowActions` - Combined view and edit actions with permission checks
- ✅ Updated main residents list page to use new components
- ✅ Added visual feedback for users without edit permissions

2. Resident Detail Page (`src/app/operator/residents/[id]/page.tsx`)

- ✅ Created `ResidentDetailActions.tsx` with:
- `EditResidentDetailButton` - Shows "View Only" badge for users without edit permissions
- `SummaryPDFButton` - Always visible for all users
- `ReadOnlyBadge` - Displays for FAMILY role users
- `ResidentDetailActionsBar` - Complete action bar with permission-based rendering
- ✅ Integrated into resident detail page
- ✅ Added tooltip feedback for restricted actions

3. AssessmentsTab (`src/components/operator/residents/AssessmentsTab.tsx`)

- ✅ Added permission imports from `@/hooks/usePermissions` and `@/lib/permissions`
- ✅ Wrapped "New Assessment" button with `ActionGuard` for `assessment.create`
- ✅ Wrapped "Create Assessment" button (empty state) with `ActionGuard`
- ✅ Wrapped "Edit" button with `ActionGuard` for `assessment.update`
- ✅ Wrapped "Delete" button with `ActionGuard` for `assessment.delete`
- ✅ View button remains accessible to all users

4. IncidentsTab (`src/components/operator/residents/IncidentsTab.tsx`)

- ✅ Added permission imports
- ✅ Wrapped “Report Incident” button with `ActionGuard` for `incident.create`
- ✅ Wrapped “Report Incident” button (empty state) with `ActionGuard`
- ✅ Wrapped “Edit” button with `ActionGuard` for `incident.update`
- ✅ Wrapped “Delete” button with `ActionGuard` for `incident.delete`
- ✅ View button remains accessible to all users

5. ComplianceTab (`src/components/operator/residents/ComplianceTab.tsx`)

- ✅ Added permission imports
- ✅ **Wrapped entire component with `RoleGuard`** restricting to `["ADMIN", "OPERATOR"]`
- ✅ Added informative fallback message for restricted users
- ✅ Wrapped “Add Compliance Item” buttons with `ActionGuard` for `compliance.create`
- ✅ Wrapped “Edit” button with `ActionGuard` for `compliance.update`
- ✅ Wrapped “Delete” button with `ActionGuard` for `compliance.delete`
- ✅ Special restriction icon (FiShield) for unauthorized access message

---

Permission Patterns Used

1. PermissionGuard Component

```
<PermissionGuard permission={PERMISSIONS.RESIDENTS_CREATE}>
  <button>Create Resident</button>
</PermissionGuard>
```

2. ActionGuard Component

```
<ActionGuard resourceType="assessment" action="create">
  <button>New Assessment</button>
</ActionGuard>
```

3. RoleGuard Component

```
<RoleGuard
  roles={["ADMIN", "OPERATOR"]}
  fallback={<div>Restricted Access Message</div>}
>
  <ComplianceContent />
</RoleGuard>
```

4. Permission Guard with Fallback

```
<PermissionGuard
  permission={PERMISSIONS.RESIDENTS_UPDATE}
  fallback={
    <button disabled className="cursor-not-allowed">
      View Only
    </button>
  }
>
  <button>Edit</button>
</PermissionGuard>
```

Role-Based UI Behavior

ADMIN

- ✅ Full access to all features
- ✅ Can create, edit, delete all resources
- ✅ Can view compliance information
- ✅ Can manage operators

OPERATOR

- ✅ Full access to their scope (homes/residents)
- ✅ Can create, edit, delete within their scope
- ✅ Can view compliance information
- ✅ Cannot manage other operators

CAREGIVER

- ✅ Can view residents
- ✅ Can create assessments and incidents (limited)
- ✅ Cannot edit/delete assessments or incidents
- ✅ Cannot view compliance information
- ✅ Read-only on most features

FAMILY

- ✅ Can view their family member’s information only
- ✅ **“View Only” badge displayed prominently**
- ✅ No edit/delete permissions
- ✅ Cannot view compliance information
- ✅ Cannot view other residents

Visual Indicators

1. Read-Only Badges

- Amber-colored badge for family members: “View Only”

- Displayed at the top of resident detail pages
- Includes eye icon (FiEye)

2. Disabled Buttons

- Buttons show as disabled when user lacks permissions
- Cursor changes to "not-allowed"
- Tooltips explain why action is restricted

3. Restricted Access Messages

- Full-page message for compliance tab (non-admin/operator users)
- Shield icon (FiShield) for security-related restrictions
- Clear, user-friendly language

4. Missing Action Buttons

- Buttons completely hidden for unauthorized users (when using ActionGuard without fallback)
- Cleaner UI without disabled elements

Files Modified

New Files Created

1. `/src/components/operator/residents/ResidentsListActions.tsx` - List page action components
2. `/src/components/operator/residents/ResidentDetailActions.tsx` - Detail page action components

Files Modified

1. `/src/app/operator/residents/page.tsx` - Main residents list page
2. `/src/app/operator/residents/[id]/page.tsx` - Resident detail page
3. `/src/components/operator/residents/AssessmentsTab.tsx` - Assessments tab with guards
4. `/src/components/operator/residents/IncidentsTab.tsx` - Incidents tab with guards
5. `/src/components/operator/residents/ComplianceTab.tsx` - Compliance tab (role-restricted)

Pending Tasks

⏳ In Progress

- FamilyTab permission guards

📋 Pending

- Sidebar navigation permission filtering
- Operator dashboard role-specific metrics
- Additional visual indicators (tooltips, help text)
- UI Permission Patterns documentation
- Comprehensive testing with all roles
- Git commit and GitHub push

Testing Checklist

Manual Testing Required

As ADMIN:

- [ ] Can see all buttons (Create, Edit, Delete)
- [ ] Can access all tabs including Compliance
- [ ] Can perform all CRUD operations
- [ ] No “View Only” badges shown

As OPERATOR:

- [ ] Can see all buttons within their scope
- [ ] Can access Compliance tab
- [ ] Can perform CRUD operations on their homes/residents
- [ ] No “View Only” badges shown

As CAREGIVER:

- [ ] Can view residents
- [ ] Can create assessments/incidents
- [ ] Cannot edit/delete assessments/incidents (buttons hidden)
- [ ] Cannot access Compliance tab (shows restricted message)
- [ ] No “View Only” badge (not applicable)

As FAMILY:

- [ ] **“View Only” badge displayed prominently**
- [ ] Can view their family member only
- [ ] All edit/delete buttons hidden or disabled
- [ ] Compliance tab shows restricted access message
- [ ] Summary PDF button still accessible

API Integration

All permission checks are enforced at both:
1. **UI Level** (Phase 4.1) - Hide/disable buttons, show badges
2. **API Level** (Phase 4) - Server-side authorization using `requirePermission()`, `requireResidentAccess()`, etc.

This dual-layer approach ensures:
- Better UX (no confusing unauthorized errors)
- Security (server validates all requests)
- Consistency (permission definitions shared between UI and API)

Performance Considerations

- Permission checks use React hooks with minimal re-renders
- `usePermissions()` hook caches permissions for the session

- Guard components only render when permissions change
- No additional API calls for permission checks (uses session data)

Deployment Notes

Pre-Deployment

- [ ] Run TypeScript compilation: `npm run build`
- [ ] Verify no type errors
- [ ] Test with different user roles in development
- [ ] Review all modified files

Post-Deployment

- [ ] Smoke test with each role (ADMIN, OPERATOR, CAREGIVER, FAMILY)
- [ ] Verify buttons hide/show correctly
- [ ] Check “View Only” badges display for FAMILY users
- [ ] Confirm Compliance tab restriction works
- [ ] Monitor error logs for authorization issues

Next Steps

1. **Complete FamilyTab updates** - Add permission guards
2. **Update Sidebar Navigation** - Filter menu items by role
3. **Update Operator Dashboard** - Role-specific metrics and quick actions
4. **Add More Visual Feedback** - Tooltips, help text, contextual messages
5. **Create UI Patterns Documentation** - Developer guide for using permission guards
6. **Comprehensive Testing** - Test all scenarios with all roles
7. **Git Commit & Push** - Commit changes with proper message

---

Developer Notes

Using Permission Guards

```
// Import guards
import { PermissionGuard, RoleGuard, ActionGuard } from '@/hooks/usePermissions';
import { PERMISSIONS } from '@/lib/permissions';

// For specific permissions
<PermissionGuard permission={PERMISSIONS.RESIDENTS_CREATE}>
  <CreateButton />
</PermissionGuard>

// For resource actions
<ActionGuard resourceType="resident" action="create">
  <CreateButton />
</ActionGuard>

// For roles
<RoleGuard roles={["ADMIN", "OPERATOR"]}>
  <AdminContent />
</RoleGuard>

// With fallback
<PermissionGuard
  permission={PERMISSIONS.RESIDENTS_UPDATE}
  fallback={<ViewOnlyMessage />}
>
  <EditButton />
</PermissionGuard>
```

Available Permissions

See `/src/lib/permissions.ts` for full list:
- `PERMISSIONS.RESIDENTS_VIEW`
- `PERMISSIONS.RESIDENTS_CREATE`
- `PERMISSIONS.RESIDENTS_UPDATE`
- `PERMISSIONS.RESIDENTS_DELETE`
- `PERMISSIONS.ASSESSMENTS_VIEW`
- `PERMISSIONS.ASSESSMENTS_CREATE`
- `PERMISSIONS.ASSESSMENTS_UPDATE`
- `PERMISSIONS.ASSESSMENTS_DELETE`
- `PERMISSIONS.INCIDENTS_VIEW`
- `PERMISSIONS.INCIDENTS_CREATE`
- `PERMISSIONS.INCIDENTS_UPDATE`
- `PERMISSIONS.INCIDENTS_DELETE`
- `PERMISSIONS.COMPLIANCE_VIEW`
- `PERMISSIONS.COMPLIANCE_CREATE`
- `PERMISSIONS.COMPLIANCE_UPDATE`
- `PERMISSIONS.COMPLIANCE_DELETE`
- And more...

Related Documentation

- Phase 4 RBAC Implementation (/PHASE_4_RBAC_IMPLEMENTATION.md) - Core RBAC system
- Permission System (/src/lib/permissions.ts) - Permission definitions
- Auth Utils (/src/lib/auth-utils.ts) - Server-side authorization utilities
- usePermissions Hook (/src/hooks/usePermissions.tsx) - Client-side permission hooks

---

Status: Phase 4.1 implementation is 70% complete. Core resident management components are fully protected with RBAC UI guards. Remaining work includes navigation filtering, dashboard updates, and comprehensive testing.