Residents Module - Part 1: UI/UX Improvements & Advanced Filters

Implementation Summary

Date: December 11, 2025
Commit: `ecb1ccb` - feat: Comprehensive polish Part 1 - UI/UX improvements and advanced filters for residents module
Status: ✅ **COMPLETED**

📦 New Components Created

1. ResidentCard Component

File: `src/components/operator/residents/ResidentCard.tsx`

Features:
- Enhanced card layout with professional design
- Avatar display with gradient fallback for initials
- Status badge (Active, Inactive, Hospitalized, etc.)
- Care level badge with color coding
- Key information grid with icons:
- Age (with calendar icon)
- Room number (with map pin icon)
- Home name (with home icon)
- Primary caregiver (with user icon)
- Assessment and incident count badges
- Admission date display
- Hover effects and smooth transitions
- Mobile-responsive design
- Clickable card linking to resident detail page

Usage:

```
<ResidentCard resident={residentData} />
<ResidentCardGrid residents={residentsArray} />
```

---

2. CareLevelBadge Component

File: `src/components/ui/CareLevelBadge.tsx`

Features:
- Color-coded badges for all care levels:
- 🟢 **Independent** (Green) - Low care needs
- 🔵 **Assisted Living** (Blue) - Moderate care needs

- 🟣 **Memory Care** (Purple) - Specialized care
- 🟠 **Skilled Nursing** (Orange) - High care needs
- Icon support with care-level-specific icons
- Configurable sizes (sm, md, lg)
- Tooltips showing care level description
- Helper functions for reuse:
- `getCareLevelColor()`
- `getCareLevelIcon()`
- `getCareLevelLabel()`

Usage:

```
<CareLevelBadge level="ASSISTED_LIVING" size="md" />
```

---

3. ResidentCardSkeleton Component

File: `src/components/operator/residents/ResidentCardSkeleton.tsx`

Features:
- Animated shimmer effect
- Matches ResidentCard layout exactly
- Multiple variants:
- `ResidentCardSkeleton` - Single card skeleton
- `ResidentListSkeleton` - Multiple card skeletons
- `ResidentTableRowSkeleton` - Single table row skeleton
- `ResidentTableSkeleton` - Multiple table row skeletons
- Configurable count for list/table variants

Usage:

```
<ResidentListSkeleton count={6} />
<ResidentTableSkeleton count={5} />
```

---

4. ResidentFilters Component

File: `src/components/operator/residents/ResidentFilters.tsx`

Features:
- **6+ Advanced Filter Options:**
1. **Care Level** - Filter by Independent, Assisted Living, Memory Care, Skilled Nursing
2. **Status** - Filter by Active, Inactive, Hospitalized, Deceased, Pending, Inquiry, Discharged
3. **Age Range** - Filter by Under 60, 60-70, 70-80, 80-90, 90+
4. **Room Floor** - Filter by Floor 1-5
5. **Admission Date Range** - Filter by Last 30 Days, Last 90 Days, Last Year, More Than a Year
6. **Assigned Caregiver** - Filter by specific caregiver assignment
- Active filter count indicator
- “Clear All Filters” button

- Responsive grid layout (1-6 columns based on screen size)
- Clean, professional UI design

Usage:

```
<ResidentFiltersComponent
  filters={filters}
  onChange={setFilters}
  onClear={handleClearFilters}
  caregivers={caregiversList}
/>
```

---

5. ResidentsListClient Component

File: `src/components/operator/residents/ResidentsListClient.tsx`

Features:
- **Enhanced Search:**
- Search by name, room number, or medical conditions
- Debounced input for performance
- Clear search button
- Results count display
- **8 Sorting Options:**
1. Name (A-Z)
2. Name (Z-A)
3. Newest Admission
4. Oldest Admission
5. Youngest First
6. Oldest First
7. By Care Level
8. By Room Number
- **View Mode Toggle:**
- Grid view (cards)
- Table view (traditional table)
- **Smart Filtering:**
- Client-side filtering for instant results
- Multiple filters active simultaneously
- Handles null/undefined values gracefully
- **Enhanced Empty States:**
- Context-aware messages
- Different states for “no data” vs “no filtered results”
- Actionable buttons
- **Loading States:**
- Skeleton loaders for both grid and table views
- **Mobile-Responsive:**
- Stacked layout on mobile
- Touch-friendly controls
- Overflow scroll for tables

Usage:

```
<ResidentsListClient
  initialResidents={residents}
  caregivers={caregiversList}
/>
```

🛠️ Utility Functions

resident-utils.ts

File: `src/lib/resident-utils.ts`

Functions:

1. `calculateAge(dateOfBirth)` - Calculate age in years from date of birth
2. `getRoomFloor(roomNumber)` - Extract floor number from room number (e.g., 201 → 2)
3. `getDaysSinceAdmission(admissionDate)` - Calculate days since admission
4. `formatDate(date, formatStr)` - User-friendly date formatting
5. `isWithinDays(date, days)` - Check if date is within a certain number of days
6. `getAgeRange(age)` - Get age range category for a given age
7. `isAgeInRange(age, range)` - Filter age by range
8. `getAdmissionDateRange(admissionDate)` - Get admission date range category
9. `isAdmissionInRange(admissionDate, range)` - Filter by admission date range
10. `getInitials(firstName, lastName)` - Get resident initials
11. `getFullName(firstName, lastName)` - Get full name

🎨 Enhanced Existing Components

1. Enhanced Tab Design

File: `src/app/operator/residents/[id]/page.tsx`

Changes:
- Added icons to each tab with proper coloring
- Added count badges for tabs with data:
- Assessments tab shows count
- Incidents tab shows count
- Family tab shows count
- Improved active tab styling
- Better mobile responsiveness with overflow scroll
- Enhanced hover states
- Whitespace nowrap for tab labels

Visual Improvements:

```
// Before:
<Link>Assessments</Link>

// After:
<Link>
  <Icon className="w-4 h-4" />
  <span>Assessments</span>
  {count > 0 && <Badge>{count}</Badge>}
</Link>
```

---

2. Enhanced Empty State Component

File: `src/components/ui/empty-state.tsx`

Changes:
- Added support for `onClick` handler in addition to `href`
- Allows for both navigation and custom actions
- Better flexibility for different use cases

Usage:

```
// With href:
<EmptyState action={{ label: "Add Resident", href: "/residents/new" }} />

// With onClick:
<EmptyState action={{ label: "Clear Filters", onClick: handleClear }} />
```

✅ Success Criteria Met

✓ Visual Design Improvements

- ✅ Enhanced resident cards with professional layout
- ✅ Better status badges with colors and icons
- ✅ Care level badges with appropriate colors
- ✅ Improved typography and spacing
- ✅ Better visual hierarchy
- ✅ Age display and calculation
- ✅ Room number display
- ✅ Primary caregiver display

✓ Loading States

- ✅ Skeleton loaders for cards
- ✅ Skeleton loaders for table rows
- ✅ Better loading feedback
- ✅ Animated shimmer effects

✓ Empty States

- ✅ Helpful messages when no data

- ✅ Actionable suggestions
- ✅ Better visual design
- ✅ Context-aware states

✓ Mobile Responsiveness

- ✅ Responsive grid layouts (1/2/3 columns)
- ✅ Mobile-friendly filters (vertical stacking)
- ✅ Touch-friendly elements
- ✅ Overflow scroll for tabs

✓ Advanced Filters

- ✅ Filter by care level (4 options)
- ✅ Filter by status (7 options)
- ✅ Filter by age range (5 options)
- ✅ Filter by room floor (5 options)
- ✅ Filter by admission date range (4 options)
- ✅ Filter by assigned caregiver
- ✅ Multiple filters active at once
- ✅ Active filter count indicator
- ✅ Clear all filters button

✓ Sorting Options

- ✅ Sort by name (A-Z, Z-A)
- ✅ Sort by admission date (newest/oldest)
- ✅ Sort by age (youngest/oldest)
- ✅ Sort by care level
- ✅ Sort by room number
- ✅ **Total: 8 sorting options**

✓ Enhanced Search

- ✅ Search by name
- ✅ Search by room number
- ✅ Search by medical conditions
- ✅ Clear search button
- ✅ Search results count

✓ Tab Design

- ✅ Better tab styling with icons
- ✅ Active tab indicators
- ✅ Tab counts (Assessments, Incidents, Family)
- ✅ Mobile-responsive overflow scroll

✓ Status Indicators

- ✅ Better visual indicators for status
- ✅ Care level badges with colors and icons
- ✅ Status badges with icons
- ✅ Age display with icon

- ✅ Room number badge

✓ Design Consistency

- ✅ Reused components from Caregivers module
- ✅ Maintained consistent design language
- ✅ Used same color scheme and spacing
- ✅ Ensured accessibility

✓ Code Quality

- ✅ TypeScript properly typed
- ✅ ESLint validation passed
- ✅ Next.js Image optimization
- ✅ Follows existing code patterns
- ✅ Comprehensive comments
- ✅ Proper imports and exports

📊 Statistics

- **New Files Created:** 6
- **Files Modified:** 2
- **Lines of Code Added:** 1,215
- **Lines of Code Modified:** 17
- **Components Created:** 5
- **Utility Functions:** 11
- **Filter Options:** 6+
- **Sorting Options:** 8
- **Loading States:** 4 variants

🚀 Integration Guide

Using the New Components

1. In a Server Component Page:

```
import { ResidentsListClient } from '@/components/operator/residents/ResidentsListCli-
ent';

export default async function Page() {
  const residents = await fetchResidents();
  const caregivers = await fetchCaregivers();

  return (
    <ResidentsListClient
      initialResidents={residents}
      caregivers={caregivers}
    />
  );
}
```

2. Using Individual Cards:

```
import { ResidentCard, ResidentCardGrid } from '@/components/operator/residents/Resid-
entCard';

// Single card:
<ResidentCard resident={resident} />

// Grid of cards:
<ResidentCardGrid residents={residentsArray} />
```

3. Using Filters:

```
import { ResidentFiltersComponent, DEFAULT_FILTERS } from '@/components/operator/res-
idents/ResidentFilters';

const [filters, setFilters] = useState(DEFAULT_FILTERS);

<ResidentFiltersComponent
  filters={filters}
  onChange={setFilters}
  onClear={() => setFilters(DEFAULT_FILTERS)}
  caregivers={caregiversList}
/>
```

4. Using Loading States:

```
import { ResidentListSkeleton, ResidentTableSkeleton } from '@/components/operator/
residents/ResidentCardSkeleton';

{isLoading ? (
  <ResidentListSkeleton count={6} />
) : (
  <ResidentCardGrid residents={residents} />
)}
```

5. Using Care Level Badges:

```
import { CareLevelBadge } from '@/components/ui/CareLevelBadge';

<CareLevelBadge level="ASSISTED_LIVING" size="md" showIcon={true} />
```

🔄 Next Steps

Immediate:

1. **Test in Browser:**
   - Verify all filters work correctly
   - Test sorting functionality
   - Confirm search works as expected
   - Check mobile responsiveness
   - Validate loading states

2. **Push to GitHub:**
   - The commit is ready: `ecb1ccb`
   - Push manually or set up GitHub token authentication
   - Command: `git push origin main`

Future Enhancements (Part 2):

- Analytics dashboard for residents module
- Export functionality (CSV, PDF)
- Document upload for residents
- Quick actions menu
- Batch operations
- Advanced reporting

🐛 Known Issues & Notes

Git Push:

- ⚠️ **Action Required:** Manual push needed due to GitHub token authentication issue
- The commit has been created successfully locally
- Changes are ready to be pushed to `origin/main`

Testing:

- ✅ All components pass ESLint validation
- ✅ TypeScript compilation successful
- ⚠️ Browser-based functional testing pending (to be done by user)

📝 Commit Information

Commit Hash: `ecb1ccb`
Branch: `main`
Files Changed: 8
Insertions: 1,215
Deletions: 17

Commit Message:

```
feat: Comprehensive polish Part 1 - UI/UX improvements and advanced filters for resid-
ents module

✨ New Components:
- ResidentCard, ResidentCardSkeleton, CareLevelBadge
- ResidentFilters, ResidentsListClient

🎨 UI/UX Enhancements:
- Enhanced cards, tabs, badges
- Better layout and spacing
- Improved mobile responsiveness

🔍 Advanced Filtering & Sorting:
- 6+ filter options, 8 sorting options
- Active filter count, Clear filters button

🛠️ Utility Functions:
- resident-utils.ts with 11 helper functions

✅ Code Quality:
- TypeScript, ESLint validated
- Next.js Image optimization
```

👥 Credits

Implementation: DeepAgent (Abacus.AI)
Project: CareLinkAI
Repository: profyt7/carelinkai
Deployment: https://carelinkai.onrender.com

End of Summary