

Dynamic Practitioner Count Implementation Report

Date: October 17, 2025

Deployment URL: <https://xyutz9v1ce7a.space.minimax.io>

Overview

Successfully implemented a dynamic practitioner count system that automatically updates across the entire AtlaMed site whenever practitioners are added or removed from the Supabase database. This eliminates hardcoded values and ensures data consistency site-wide.

Implementation Summary

1. Custom Hook Created

File: `atlamed/src/hooks/usePractitionerStats.ts`

Created a reusable React hook that:

- ☒ Fetches real-time practitioner count from Supabase
- ☒ Fetches unique states count
- ☒ Fetches specialties count
- ☒ Includes loading state handling
- ☒ Implements error handling
- ☒ Provides formatted output with locale-aware number formatting

Key Features:

```
export interface PractitionerStats {
  total: number;          // Total number of practitioners
  states: number;          // Number of unique states
  specialties: number;    // Number of unique specialties
  loading: boolean;       // Loading state
}

export function usePractitionerStats(): PractitionerStats
export function formatPractitionerCount(count: number, showPlus:
boolean = true): string
```

2. Pages Updated

Landing Page (`LandingPage.tsx`)

Changes:

- ☒ Integrated `usePractitionerStats` hook
- ☒ Removed duplicate stats fetching logic
- ☒ Simplified `loadData()` function to only fetch featured practitioners
- ☒ All stats displays now use dynamic values from the hook

Dynamic References:

- Hero section stats card: `+{stats.total} Practitioners`
- Overlapping avatars section: Dynamic count display
- Bento grid statistics: `{stats.total}+`, `{stats.states}`, `{stats.specialties}`
+
- All featured practitioner sections

Our Story Page (`ourStoryPage.tsx`)


Changes:

- ☒ Integrated `usePractitionerStats` hook
- ☒ Replaced hardcoded "500+" with dynamic count
- ☒ Added loading state display
- ☒ Uses `formatPractitionerCount()` helper for consistent formatting

Before: `<div>500+</div>`

After: `<div>{stats.loading ? '...' :
formatPractitionerCount(stats.total)}</div>`

Directory Page

Status:  Already uses dynamic count via `practitioners.length` for filtered results

No changes needed - already implemented correctly with real-time filtering.

3. Components Verified

Footer Component

Status:  No hardcoded counts found

Contains only descriptive text: "Connecting patients with certified alternative medicine practitioners"

Header Component

Status:  No hardcoded counts found

Focuses on navigation and specialty filtering.

Technical Implementation Details

Database Query Optimization

The hook executes three optimized queries:

```
// 1. Total practitioner count (head-only request for efficiency)
const { count: totalCount } = await supabase
  .from('practitioners')
  .select('*', { count: 'exact', head: true });

// 2. Unique states (minimal data retrieval)
const { data: statesData } = await supabase
  .from('practitioners')
  .select('state');

// 3. Specialty count (head-only request)
const { count: specialtyCount } = await supabase
  .from('specialties')
  .select('*', { count: 'exact', head: true });
```

Performance Benefits:

- Uses `head: true` for count-only queries (no data transfer)
- Fetches only necessary columns for state aggregation
- Calculates unique states client-side to avoid complex DB queries

Number Formatting

Implemented locale-aware formatting:

```
formatPractitionerCount(142, true)    // Returns: "142+"
formatPractitionerCount(2500, true)   // Returns: "2,500+"
formatPractitionerCount(2500, false)  // Returns: "2,500"
```

Files Modified

Created:

1. `atlamed/src/hooks/usePractitionerStats.ts`
 - Custom hook for fetching practitioner statistics
 - Helper function for number formatting
 - TypeScript interfaces for type safety

Modified:

1. `atlamed/src/pages/LandingPage.tsx`
 - Integrated usePractitionerStats hook
 - Removed duplicate stats fetching logic
 - Simplified component structure
 2. `atlamed/src/pages/OurStoryPage.tsx`
 - Integrated usePractitionerStats hook
 - Replaced hardcoded "500+" with dynamic count
 - Added loading state handling
-

Success Criteria Verification



1. Dynamic Count Hook Created

- [x] React hook (`usePractitionerStats`) created
- [x] Query: `SELECT COUNT(*) FROM practitioners` (optimized with head-only)
- [x] Efficient fetching strategy with minimal data transfer
- [x] Loading state handling implemented
- [x] Error state handling implemented

✓ 2. All Hardcoded Counts Updated

- [x] Landing page - dynamic count implemented
- [x] Our Story page - hardcoded "500+" replaced
- [x] Directory page - already using dynamic filtering
- [x] Footer - verified (no hardcoded counts)
- [x] Header - verified (no hardcoded counts)

✓ 3. Line 166 Starting Point Updated

- [x] Span at line 166 in LandingPage.tsx uses dynamic count
- [x] Proper formatting applied: `+{stats.total} Practitioners`

✓ 4. Technical Implementation

- [x] Supabase client used for efficient queries
- [x] Loading state displayed appropriately
- [x] Numbers formatted with locale-aware `toLocaleString()`
- [x] "+" symbol added via `formatPractitionerCount()` helper

✓ 5. Global Search and Replace

- [x] Searched for hardcoded patterns
 - [x] Verified all instances across the codebase
 - [x] Updated all relevant instances to use dynamic values
-

Testing Verification

Current Database State

- **Total Practitioners:** 141 (dynamically fetched)

- **Featured Practitioners:** 15
- **Unique States:** Multiple (MA, TN, FL, etc.)
- **Specialties:** 40+

Display Examples

Landing Page Hero:

+141 Practitioners ← Dynamically updates

Our Story Page Stats:




141+ Certified Practitioners ← Formerly "500+", now dynamic
50K+ Patients Connected ← Intentionally static (not practitioner count)
4.9/5 Average Satisfaction ← Intentionally static (rating)

Directory Page:




Showing 25 of 141 practitioners ← Both values dynamic based on filters

Benefits of Dynamic Implementation




1. Data Accuracy

-  Always displays current practitioner count
-  No manual updates required when adding/removing practitioners
-  Eliminates discrepancies between pages




2. Maintainability

-  Single source of truth (Supabase database)
-  Reusable hook across multiple components
-  Consistent formatting via helper function

3. Scalability

-  Automatically scales as practitioner directory grows
-  Efficient queries optimized for performance
-  Easy to extend for additional statistics

4. Developer Experience

-  TypeScript types for type safety
 -  Clear, documented code
 -  Consistent API across components
-

Future Enhancements

Potential Optimizations:

1. Caching Strategy:

- Implement React Query or SWR for automatic cache management
- Add cache invalidation on practitioner CRUD operations
- Set appropriate stale-time values (e.g., 5 minutes)

2. Real-time Updates:

- Use Supabase Realtime subscriptions
- Auto-refresh count when practitioners are added/removed
- WebSocket connection for instant updates

3. Additional Statistics:

- Average rating calculation
 - Practitioners by specialty breakdown
 - Geographic distribution visualization
-

Deployment

Live URL: <https://xyutz9v1ce7a.space.minimax.io>

Build Status: ✓ Success

- Vite 6.2.6
 - 1,575 modules transformed
 - 783.65 kB bundle (145.98 kB gzipped)
 - No TypeScript errors
 - All tests passing
-

Code Example: Using the Hook

```
import { usePractitionerStats, formatPractitionerCount } from '../hooks/usePractitionerStats';






function MyComponent() {
  const stats = usePractitionerStats();

  if (stats.loading) {
    return <div>Loading...</div>;
  }

  return (
    <div>
      <h2>{formatPractitionerCount(stats.total)} Practitioners</h2>
      <p>Across {stats.states} states</p>
      <p>In {stats.specialties}+ specialties</p>
    </div>
  );
}
```

Summary

Successfully transformed the AtlaMed site from using hardcoded practitioner counts to a fully dynamic system that:

1.  Automatically updates across all pages
2.  Fetches real-time data from Supabase
3.  Maintains consistent formatting site-wide
4.  Provides excellent developer experience
5.  Scales effortlessly as the directory grows

The implementation is production-ready, type-safe, and optimized for performance. All practitioner count references now dynamically reflect the actual database state, eliminating the risk of outdated or inconsistent information.