Caregivers API Debug Fix

Issue

The caregivers page continues to show “Something went wrong” / “Failed to load caregivers” error after multiple RBAC fix attempts.

Root Cause Analysis

Previous Attempts

1. **Commit 67866bc**: Fixed Prisma singleton issue
2. **Commit f82c73c**: Migrated to Phase 4 RBAC system

Both commits were successful in terms of deployment and code structure, but the 500 error persisted.

Actual Problem

The API was returning a generic 500 Internal Server Error with response:

```
{
  "error": "Internal server error"
}
```

This generic error was hiding the real issue. Without detailed server-side logs, it was impossible to identify the exact failure point.

Solution

Changes Made

1. **Added Comprehensive Logging**: Step-by-step logging at every critical point in the API flow
2. **Removed Problematic OrderBy**: Removed `orderBy: { employmentStatus: 'asc' }` which might cause issues
3. **Added Null-Safe Data Transformation**: All data transformations now have null checks and default values
4. **Added Array Validation**: Explicit checks for `Array.isArray()` before mapping
5. **Individual Error Handling**: Try-catch within the map function to catch transformation errors per caregiver
6. **Fixed Type Filter Bug**: Line 41 was checking `status !== 'ALL'` instead of `type !== 'ALL'`

Code Changes

Before

```
// Query caregivers with all related data
const caregivers = await prisma.caregiver.findMany({
  where: caregiverWhere,
  include: {
    user: {
      select: {
        firstName: true,
        lastName: true,
        email: true,
        phoneNumber: true,
      }
    },
    certifications: { /* ... */ }
  },
  orderBy: {
    employmentStatus: 'asc' // <-- Potentially problematic
  }
});

return NextResponse.json({
  caregivers: caregivers.map((caregiver) => ({
    id: caregiver.id,
    user: {
      firstName: caregiver.user.firstName, // <-- No null checks
      /* ... */
    },
    specializations: caregiver.languages || [], // <-- No array check
    /* ... */
  })),
});
```

After

```
// Step-by-step logging
console.log('[Caregivers API] Step 4: Querying database...');

// Query without problematic orderBy
const caregivers = await prisma.caregiver.findMany({
  where: caregiverWhere,
  include: {
    user: {
      select: {
        id: true, // <-- Added id for better debugging
        firstName: true,
        lastName: true,
        email: true,
        phoneNumber: true,
      }
    },
    certifications: { /* ... */ }
  },
  // Removed orderBy
});

console.log('[Caregivers API] Step 5: Found', caregivers.length, 'caregivers');

// Transform with error handling
const transformedCaregivers = caregivers.map((caregiver) => {
  try {
    return {
      id: caregiver.id,
      user: {
        firstName: caregiver.user?.firstName || '', // <-- Null-safe with defaults
        lastName: caregiver.user?.lastName || '',
        email: caregiver.user?.email || '',
        phoneNumber: caregiver.user?.phoneNumber || null,
      },
      photoUrl: caregiver.photoUrl || null,
      specializations: Array.isArray(caregiver.languages) ? caregiver.languages :
[], // <-- Array check
      employmentType: caregiver.employmentType || 'FULL_TIME',
      employmentStatus: caregiver.employmentStatus || 'ACTIVE',
      certifications: Array.isArray(caregiver.certifications) ? care-
giver.certifications.map(cert => ({
        id: cert.id,
        expiryDate: cert.expiryDate,
      })) : []
    };
  } catch (transformError) {
    console.error('[Caregivers API] Error transforming caregiver:', caregiver.id, tran
sformError);
    throw transformError;
  }
});

console.log('[Caregivers API] Step 7: Returning', transformedCaregivers.length, 'care-
givers');
return NextResponse.json({
  caregivers: transformedCaregivers,
});
```

Logging Flow

The new logging will show exactly where the error occurs:
1. **Step 1**: Checking permissions
2. **Step 2**: Parsing query params
3. **Step 3**: Getting user scope
4. **Step 4**: Querying database
5. **Step 5**: Found X caregivers
6. **Step 6**: Transforming data
7. **Step 7**: Returning data

If an error occurs at any step, we'll see exactly which step failed and the full error details.

Benefits

1. **Detailed Error Tracking**: Know exactly where the failure occurs
2. **Null Safety**: Prevents crashes from missing data
3. **Array Validation**: Ensures arrays are actual arrays before mapping
4. **Individual Error Handling**: One bad caregiver record won't crash the entire response
5. **Better Debugging**: Complete visibility into the API flow

Deployment

Files Changed

- `src/app/api/operator/caregivers/route.ts`
- `CAREGIVERS_API_DEBUG_FIX.md` (this file)

Testing Steps

After deployment, monitor Render logs for:

1. **Success Case**: Should see all 7 steps in logs
   ```
   [Caregivers API] Starting request...
     [Caregivers API] Step 1: Checking permissions...
     [Caregivers API] User authorized: user@example.com ADMIN
     [Caregivers API] Step 2: Parsing query params...
     ...
     [Caregivers API] Step 7: Returning 5 caregivers
   ```
2. **Error Case**: Should see which step failed
   ```
   [Caregivers API] Step 4: Querying database...
     [Caregivers API] ERROR - Failed at some step
     [Caregivers API] Error message: [actual error]
   ```

Verification

1. Visit https://carelinkai.onrender.com/operator/caregivers
2. Check if page loads successfully
3. If error persists, check Render logs for detailed error message
4. Use the step number to identify the exact failure point

Next Steps

If this fix doesn’t resolve the issue, the detailed logs will tell us:
- Exactly which step is failing
- The actual error message from Prisma/Database
- Whether it’s an auth issue, database issue, or data transformation issue

This will allow for a targeted fix instead of guessing.

Related Issues

- Caregivers page “Failed to load caregivers” error
- Phase 4 RBAC migration
- Phase 6 Caregiver Management

Commits

- Previous: 67866bc, f82c73c
- This fix: (will be added after commit)