Migration Fix Guide: Phase 2 Assessments & Incidents Fields

Issue Summary

Failed Migrations:
1. `20251208170953_add_assessments_incidents_fields` (original)
2. `20251208170953_add_assessments_incidents_fields.failed_backup` (backup attempt)

Database: PostgreSQL (`carelinkai_db`) on Render
Error Code: P3009 - Migration failed in the target database
Root Cause: BOTH migrations are recorded as "failed" in the `_prisma_migrations` table from previous deployment attempts

The `_prisma_migrations` Table Issue

Critical Understanding: Even though we removed the `.failed_backup` folder from the codebase, the **database still has a record** of both migrations in the `_prisma_migrations` table:

```sql
-- Example of what's in the database:
SELECT migration_name, finished_at, rolled_back_at, applied_steps_count
FROM _prisma_migrations
WHERE migration_name LIKE '20251208170953%';

-- Results show:
-- 1. 20251208170953_add_assessments_incidents_fields (status: failed)
-- 2. 20251208170953_add_assessments_incidents_fields.failed_backup (status: failed)
```

Why This Happens: When Prisma attempts to apply a migration, it immediately creates a record in `_prisma_migrations` with a timestamp. If the migration fails mid-execution, that record remains marked as "failed". Removing the migration folder from the codebase does NOT remove this database record.

Why Both Must Be Resolved: Prisma checks the `_prisma_migrations` table before applying new migrations. If ANY migration is in a "failed" state, it blocks all subsequent migrations. Therefore, we must mark BOTH failed records as "rolled back" before deploying the new safe migration.

Impact

- New migrations cannot be applied until BOTH failed migrations are resolved
- Phase 2 features (Assessments and Incidents tabs) may have incomplete database schema
- Production deployments are blocked
- The error message references the `.failed_backup` migration even though the folder no longer exists in the codebase

Solution Overview

This fix includes:
1. **Migration Resolution Script**: Marks **BOTH** failed migrations as “rolled back”
2. **New Idempotent Migration**: Safely adds all required fields (can run multiple times)
3. **Comprehensive Testing**: Verification steps to ensure success

Key Changes in This Update:
- ✅ `scripts/resolve-failed-migration.sh` now resolves both migrations
- ✅ `package.json` `migrate:resolve-manual` script chains both resolution commands
- ✅ Documentation explains the database table issue clearly

Step-by-Step Fix Process

Prerequisites

- Access to Render dashboard or production database
- Environment variable `DATABASE_URL` configured
- Node.js and npm installed
- Prisma CLI available (`npx prisma`)

Option 1: Automated Fix (Recommended)

This uses the provided script to automate the resolution process.

1. Set Database URL

If running locally, ensure your `DATABASE_URL` points to the production database:

```
export DATABASE_URL="postgresql://carelinkai_db_user:password@dpg-xxxx.oregon-postgres.render.com/carelinkai_db"
```

⚠️ **WARNING**: Be extremely careful when setting this. You are working with the production database!

2. Run the Resolution Script

```
npm run migrate:resolve
```

This script will:
- Check if `DATABASE_URL` is set
- Show current migration status
- Ask for confirmation
- Mark **BOTH** failed migrations as “rolled back”
- Show updated migration status

Expected Output:

```
✅ DATABASE_URL is set

📋 Current migration status:
----------------------------
[Shows migration status including both failed migrations]

⚠️  This script will mark BOTH failed migrations as rolled back:
   Migration 1: 20251208170953_add_assessments_incidents_fields
   Migration 2: 20251208170953_add_assessments_incidents_fields.failed_backup

ℹ️  Note: The .failed_backup migration was recorded in the database
   during a previous deployment attempt, even though the folder
   was removed from the codebase.

Do you want to proceed? (yes/no): yes

🔄 Resolving failed migrations...

1️⃣  Resolving: 20251208170953_add_assessments_incidents_fields
2️⃣  Resolving: 20251208170953_add_assessments_incidents_fields.failed_backup

✅ Both migrations marked as rolled back successfully
```

3. Deploy the New Migration

```
npm run migrate:deploy
```

This will apply the new idempotent migration:
`20251208181611_add_assessments_incidents_fields_safe`

Expected Output:

```
Applying migration `20251208181611_add_assessments_incidents_fields_safe`
The following migration(s) have been applied:

migrations/
  └─ 20251208181611_add_assessments_incidents_fields_safe/
    └─ migration.sql

✅ All migrations have been applied successfully
```

4. Verify the Fix

Check migration status:

```
npx prisma migrate status
```

Expected Output:

```
Database schema is up to date!
```

Option 2: Manual Fix

If you prefer manual control or the automated script fails:

1. Connect to Production Database

Using Render Shell or your preferred SQL client.

2. Check Migration Status

```
npx prisma migrate status
```

You should see the failed migration listed.

3. Mark Both Migrations as Rolled Back

```
npm run migrate:resolve-manual
```

Or directly (both commands required):

```
npx prisma migrate resolve --rolled-back
"20251208170953_add_assessments_incidents_fields"
npx prisma migrate resolve --rolled-back "20251208170953_add_assessments_incidents_fie
lds.failed_backup"
```

4. Deploy New Migration

```
npm run migrate:deploy
```

5. Verify Success

```
npx prisma migrate status
```

Option 3: Render Dashboard (No SSH Access)

If you’re deploying via Render and don’t have direct database access:

1. Add Resolution Command to Render Deploy

In your Render service settings, update the **Build Command**:

```
npm install && npm run migrate:resolve-manual && npm run migrate:deploy && npm run
build
```

⚠️ **Note**: This approach runs the resolution on every deploy. After the first successful deployment, you should remove `npm run migrate:resolve-manual` from the build command.

2. Trigger a Manual Deploy

Go to your Render dashboard and trigger a manual deploy.

3. Monitor Logs

Watch the deployment logs to ensure:
- Migration resolution succeeds
- New migration applies successfully
- Build completes without errors

4. Verify Application

- Check that the application starts successfully
- Test the Assessments and Incidents tabs in the Operator Residents module
- Verify data persistence

Verification Steps

After applying the fix, verify everything is working:

1. Check Database Schema

```
npx prisma db pull
```

This should match your `schema.prisma` file.

2. Check Migration History

```
npx prisma migrate status
```

Expected output:

```
Database schema is up to date!
```

3. Test Phase 2 Features

1. Log in as an Operator
2. Navigate to a Resident detail page
3. Go to the “Assessments” tab
4. Create a new assessment
5. Go to the “Incidents” tab
6. Create a new incident
7. Verify data saves correctly

4. Check Application Logs

Monitor for any database errors or migration-related issues:

```
# On Render
# Go to your service → Logs
```

Technical Details

What the New Migration Does

The new idempotent migration (`20251208181611_add_assessments_incidents_fields_safe`) safely adds the following fields:

AssessmentResult Table

- `status` (TEXT) - Assessment status (COMPLETED, IN_PROGRESS, etc.)
- `conductedBy` (TEXT) - Staff member who conducted the assessment
- `conductedAt` (TIMESTAMP) - When assessment was conducted
- `notes` (TEXT) - Observations and notes
- `recommendations` (TEXT) - Recommendations based on results

ResidentIncident Table

- `status` (TEXT) - Incident status (REPORTED, UNDER_REVIEW, RESOLVED, etc.)
- `location` (TEXT) - Where incident occurred
- `reportedBy` (TEXT) - Staff member who reported
- `reportedAt` (TIMESTAMP) - When incident was reported
- `witnessedBy` (TEXT) - Witnesses
- `actionsTaken` (TEXT) - Immediate actions taken
- `followUpRequired` (BOOLEAN) - Whether follow-up is needed
- `resolutionNotes` (TEXT) - How incident was resolved
- `resolvedAt` (TIMESTAMP) - When incident was resolved
- `resolvedBy` (TEXT) - Staff member who resolved

Indexes Created

- `AssessmentResult` : conductedAt, type, status
- `ResidentIncident` : type, severity, status, reportedAt

Why It's Idempotent

The migration uses PostgreSQL's `DO` blocks with `IF NOT EXISTS` checks:

```
DO $$
BEGIN
    IF NOT EXISTS (SELECT 1 FROM information_schema.columns
                   WHERE table_name='AssessmentResult' AND column_name='status') THEN
        ALTER TABLE "AssessmentResult" ADD COLUMN "status" TEXT;
    END IF;
END $$;
```

This means:
- ✅ Safe to run multiple times
- ✅ Won't fail if columns already exist
- ✅ Won't duplicate existing data
- ✅ Handles partial application gracefully

Troubleshooting

Issue: “Migration already applied”

Cause: The migration was already applied successfully
Solution: Check `npx prisma migrate status`. If schema is up to date, no action needed.

Issue: “Cannot connect to database”

Cause: `DATABASE_URL` is incorrect or database is unreachable
Solution:
1. Verify `DATABASE_URL` environment variable
2. Check database connection from Render dashboard
3. Ensure database is not paused or suspended

Issue: “Migration fails with column already exists”

Cause: Partial migration was applied
Solution: The new idempotent migration should handle this. If it still fails:
1. Check which columns exist:
`sql`
```
   SELECT column_name FROM information_schema.columns
   WHERE table_name IN ('AssessmentResult', 'ResidentIncident');
```
2. Manually add missing columns or contact support

Issue: “Permission denied”

Cause: Database user lacks ALTER TABLE permissions
Solution: Ensure the database user has full permissions on the schema

Issue: Script permission denied

Cause: Script file is not executable
Solution:

```
chmod +x scripts/resolve-failed-migration.sh
```

Rollback Plan

If the fix causes issues, you can rollback:

1. Identify Last Good Migration

```
npx prisma migrate status
```

2. Manually Rollback (if needed)

Since Prisma doesn’t have automatic rollback, you would need to:

1. Drop the new columns manually:

```
-- AssessmentResult
ALTER TABLE "AssessmentResult" DROP COLUMN IF EXISTS "status";
ALTER TABLE "AssessmentResult" DROP COLUMN IF EXISTS "conductedBy";
ALTER TABLE "AssessmentResult" DROP COLUMN IF EXISTS "notes";
ALTER TABLE "AssessmentResult" DROP COLUMN IF EXISTS "recommendations";

-- ResidentIncident
ALTER TABLE "ResidentIncident" DROP COLUMN IF EXISTS "status";
ALTER TABLE "ResidentIncident" DROP COLUMN IF EXISTS "location";
ALTER TABLE "ResidentIncident" DROP COLUMN IF EXISTS "reportedBy";
ALTER TABLE "ResidentIncident" DROP COLUMN IF EXISTS "reportedAt";
ALTER TABLE "ResidentIncident" DROP COLUMN IF EXISTS "witnessedBy";
ALTER TABLE "ResidentIncident" DROP COLUMN IF EXISTS "actionsTaken";
ALTER TABLE "ResidentIncident" DROP COLUMN IF EXISTS "followUpRequired";
ALTER TABLE "ResidentIncident" DROP COLUMN IF EXISTS "resolutionNotes";
ALTER TABLE "ResidentIncident" DROP COLUMN IF EXISTS "resolvedAt";
ALTER TABLE "ResidentIncident" DROP COLUMN IF EXISTS "resolvedBy";
```

1. Mark the new migration as rolled back:

```
npx prisma migrate resolve --rolled-back
"20251208181611_add_assessments_incidents_fields_safe"
```

⚠️ **Note**: Only perform rollback if absolutely necessary and Phase 2 data is not critical.

Post-Fix Actions

After successfully applying the fix:

1. ✅ **Remove resolution script from Render build command** (if added)
2. ✅ **Update team** about the fix
3. ✅ **Monitor application logs** for 24 hours
4. ✅ **Test Phase 2 features** thoroughly
5. ✅ **Document any additional issues** found

Prevention for Future Migrations

To avoid similar issues in the future:

1. Always Test Migrations Locally First

```
# Test on local database
npm run prisma:migrate
```

2. Use Staging Environment

Apply migrations to staging before production:

```
# Set staging DATABASE_URL
export DATABASE_URL="<staging-database-url>"
npm run migrate:deploy
```

3. Make Migrations Idempotent

Always use `IF NOT EXISTS` or `DO` blocks for schema changes that might be partially applied.

4. Backup Before Major Migrations

```
# On Render, use their backup feature
# Or manually backup:
pg_dump $DATABASE_URL > backup_$(date +%Y%m%d_%H%M%S).sql
```

5. Monitor Deployment Logs

Always watch logs during deployment to catch migration failures early.

Support

If you encounter issues not covered in this guide:

1. Check application logs on Render
2. Review Prisma documentation: https://www.prisma.io/docs/guides/migrate
3. Check the `_prisma_migrations` table directly for migration status
4. Contact the development team with:
   - Error messages
   - Migration status output
   - Database logs

Quick Reference

Common Commands

```
# Check migration status
npx prisma migrate status

# Resolve failed migration (automated)
npm run migrate:resolve

# Resolve failed migration (manual)
npm run migrate:resolve-manual

# Deploy migrations
npm run migrate:deploy

# Generate Prisma Client
npm run prisma:generate

# View database in Prisma Studio
npm run prisma:studio
```

Files Modified

- `prisma/migrations/20251208181611_add_assessments_incidents_fields_safe/migration.sql` - New idempotent migration
- `scripts/resolve-failed-migration.sh` - Automated resolution script
- `package.json` - Added `migrate:resolve` and `migrate:resolve-manual` scripts
- `MIGRATION_FIX_GUIDE.md` - This documentation

Files Renamed/Moved

- ~~ `prisma/migrations/20251208170953_add_assessments_incidents_fields/` → `20251208170953_add_assessments_incidents_fields.failed_backup/` ~~ (Initial rename - still in migrations directory)
- **UPDATE (Dec 8, 2025)**: `prisma/migrations/20251208170953_add_assessments_incidents_fields.failed_backup/` → `backup/failed_migrations/20251208170953_add_assessments_incidents_fields.failed_backup/`

Why the Backup Was Moved

Issue Discovered: Even with the `.failed_backup` suffix, Prisma was still attempting to apply the migration because it remained in the `prisma/migrations/` directory. Prisma treats ANY folder in the migrations directory as a valid migration, regardless of naming.

Solution: The failed migration backup was moved out of the migrations directory entirely to `/backup/failed_migrations/` to ensure Prisma does not attempt to apply it during deployments.

Commit: `f231e94` - “fix: Remove failed migration backup from migrations directory”

Summary

✅ **Problem**: TWO failed migrations blocking deployments (original + .failed_backup)
✅ **Root Cause**: Database `_prisma_migrations` table has records for both migrations

✅ **Solution**: Resolve BOTH failed migrations + apply new idempotent migration
✅ **Safety**: New migration can run multiple times safely
✅ **Verification**: Multiple verification steps provided
✅ **Support**: Comprehensive troubleshooting included

Estimated Time: 5-10 minutes for resolution and deployment

Last Updated: December 8, 2025
Version: 1.1 (Updated to resolve both failed migrations)
Status: Ready for Production Deployment