Deployment Monitoring Guide

🚀 Deployment Status

Commit: `fb75e1e` - “fix: Resolve failed database migration blocking deployments”
Pushed: ✅ Successfully pushed to GitHub
Time: December 18, 2025

📊 What to Monitor

1. Render Dashboard

Visit: https://dashboard.render.com/web/srv-d3isoajuibrs73d5fh7g

Look for:
- New deployment triggered automatically
- Build status changing from “Building” → “Live”

2. Build Logs

Click on the latest deployment to view logs.

Key Success Indicators:

Step 1: Migration Resolution

```
=========================================
Failed Migration Resolution Script
=========================================

Migration: 20251218162945_update_homes_to_active
Action: Mark as rolled back

✅ Database URL configured
🔧 Attempting to resolve failed migration...
✅ Migration resolved successfully
```

Step 2: Migration Deployment

```
Applying migration `20251218162945_update_homes_to_active`
The following migration(s) have been applied:

migrations/
  └─ 20251218162945_update_homes_to_active/
    └─ migration.sql

Your database is now in sync with your schema.
```

Step 3: Build Success

```
Build completed successfully
Starting web service...
Service is live ✅
```

3. Application Health

After deployment shows “Live”, check:

Health Endpoint:

```
curl https://carelinkai.onrender.com/api/health
```

Expected Response:

```
{
  "status": "ok",
  "database": "connected"
}
```

⏱️ Expected Timeline

Phase	Duration	Status
GitHub Push	Complete	✅ Done
Render Detection	1-2 min	⏳ Waiting
Build Start	+30 sec	⏳ Pending
Migration Resolution	+15 sec	⏳ Pending
Migration Apply	+20 sec	⏳ Pending
Build Complete	+3-5 min	⏳ Pending
Service Live	+30 sec	⏳ Pending
Total	**6-8 min**	🔄 In Progress

✅ Verification Checklist

After deployment shows “Live”:

1. Basic Health

- [ ] Application loads: https://carelinkai.onrender.com
- [ ] No 500 errors in UI
- [ ] Health endpoint returns 200 OK

2. Database State

Check that the migration applied successfully:

Via Render Shell:

```
# Open Render Shell for your service
# Then run:
npx prisma migrate status
```

Expected Output:

```
Status:

Datasource: db (PostgreSQL)

The following migration(s) are applied:

migrations/
  └─ 20251218162945_update_homes_to_active/
    └─ migration.sql (applied)
```

3. Data Verification

Verify all homes are ACTIVE:

Via Render Shell:

```
npx prisma studio
# Or use psql:
psql $DATABASE_URL -c "SELECT id, name, status FROM \"AssistedLivingHome\";"
```

Expected: All homes should have `status = 'ACTIVE'`

4. Feature Testing

Test the features that were failing:

Tour Submission

1. Go to: https://carelinkai.onrender.com/find-care
2. Select a home
3. Fill out tour request form
4. Click “Submit”
5. **Expected**: Success message, no errors

Home Inquiries

1. Go to: https://carelinkai.onrender.com/dashboard/inquiries
2. Create a new inquiry

3. **Expected**: Inquiry created successfully

🚨 Troubleshooting

If Build Fails

Error: Migration Still Failed

Logs show:

```
Error: P3009
migrate found failed migrations in the target database
```

Solution:
1. Check resolution script ran successfully
2. Manually resolve via Render Shell:

```
bash
  npx prisma migrate resolve --rolled-back 20251218162945_update_homes_to_active
  npx prisma migrate deploy
```

Error: Migration Already Applied

Logs show:

```
No pending migrations to apply.
```

This is GOOD! It means:
- The migration was already applied successfully
- The fix worked
- Proceed to verification

Error: Database Connection

Logs show:

```
Can't reach database server
```

Solution:
1. Check DATABASE_URL environment variable in Render
2. Verify PostgreSQL service is running
3. Check database service logs in Render

If Verification Fails

Tour Submission Still Failing

Check:
1. Browser console for errors
2. Network tab for failed API calls
3. Render logs for application errors

Common Issues:
- NEXTAUTH_URL not set correctly

- Missing environment variables
- Frontend cache issues (hard refresh: Ctrl+Shift+R)

Database Shows No Active Homes

Fix:

```
# Via Render Shell
npx prisma studio
# Or
psql $DATABASE_URL -c "UPDATE \"AssistedLivingHome\" SET status = 'ACTIVE' WHERE
status != 'ACTIVE';"
```

📝 What Changed

Files Modified

1. **Migration SQL**: Removed SELECT statement
2. **Resolution Script**: Auto-resolves failed migration
3. **Package.json**: Updated migrate:deploy command
4. **Documentation**: Added comprehensive guides

How The Fix Works

```
Deployment Starts
    ↓
npm run migrate:deploy
    ↓
scripts/resolve-failed-migration-20251218.sh runs
    ↓
Marks failed migration as "rolled back"
    ↓
prisma migrate deploy runs
    ↓
Applies fixed migration (UPDATE only)
    ↓
Migration succeeds ✅
    ↓
Build continues
    ↓
Application starts
    ↓
Deployment complete ✅
```

🎯 Success Criteria

Deployment is successful when:
1. ✅ Build shows “Live” status
2. ✅ No P3009 errors in logs
3. ✅ Migration applied successfully
4. ✅ Application loads without errors

5. ✅ Tour submissions work
6. ✅ All homes have ACTIVE status

📞 Next Steps

Immediate (Within 10 Minutes)

1. Monitor Render dashboard for deployment status
2. Check build logs for success indicators
3. Verify application health endpoint

Short Term (Within 1 Hour)

1. Test tour submission functionality
2. Verify all features working
3. Check for any unexpected errors

Long Term (Within 24 Hours)

1. Monitor application stability
2. Review error logs for any issues
3. Update documentation if needed

📚 Related Documentation

- **Main Fix Documentation**: `DATABASE_FIX_COMPLETE.md`
- **Resolution Script**: `scripts/resolve-failed-migration-20251218.sh`
- **Migration File**: `prisma/migrations/20251218162945_update_homes_to_active/migration.sql`

✅ Current Status

Item	Status
Fix Implemented	✅ Complete
Committed to Git	✅ Complete
Pushed to GitHub	✅ Complete
Render Deployment	🔄 In Progress
Verification	⏳ Pending

Last Updated: December 18, 2025
Monitor At: https://dashboard.render.com/web/srv-d3isoajuibrs73d5fh7g

---

Note: This deployment should resolve the database issues that were causing tour submission failures and other problems. The migration resolution script will automatically handle the failed migration, allowing deployments to proceed normally.