Bug Reporting System - Deployment Guide

Overview

This guide walks through deploying the bug reporting system to production on Render.

Pre-Deployment Checklist

- ✅ Code committed to GitHub (commit: 68cebdf)
- ✅ Database migration created and tested
- ✅ API endpoints implemented and validated
- ✅ Frontend components created and integrated
- ✅ Admin dashboard functional
- ✅ Documentation complete

Deployment Steps

1. GitHub Push (COMPLETED ✅)

```
# Already pushed to main branch
git push origin main
```

2. Render Auto-Deployment

Render will automatically detect the push and start deploying:
- Monitor at: https://dashboard.render.com
- Build logs will show: "Detected push to main branch"
- Deployment typically takes 5-10 minutes

3. Database Migration

The migration will be applied automatically during deployment via the build command:

```
# This runs automatically on Render
npx prisma migrate deploy
npx prisma generate
```

If manual migration is needed:

```
# Connect to Render shell
# Run:
npx prisma migrate deploy
```

4. Verify Deployment

Check Build Logs

1. Go to Render Dashboard
2. Select your service

3. Check “Logs” tab
4. Look for:
   - ✅ “Migration applied successfully”
   - ✅ “Build completed successfully”
   - ✅ “Service is live”

Test Bug Reporting

1. Visit: https://getcarelinkai.com
2. Look for floating “Report Bug” button (bottom-right)
3. Click button and test form submission
4. Verify success message

Test Admin Dashboard

1. Login as admin
2. Visit: https://getcarelinkai.com/admin
3. Click “Bug Reports” quick action
4. Verify bug reports page loads
5. Test filtering and status updates

Post-Deployment Validation

Functional Tests

- [ ] Floating button visible on all pages
- [ ] Bug report modal opens correctly
- [ ] Can submit bug report as logged-in user
- [ ] Can submit bug report as guest
- [ ] Admin can view bug reports list
- [ ] Admin can filter by status/severity
- [ ] Admin can update bug status
- [ ] Mobile responsiveness works

Database Verification

```sql
-- Check if BugReport table exists
SELECT table_name
FROM information_schema.tables
WHERE table_name = 'BugReport';

-- Check if enums exist
SELECT typname
FROM pg_type
WHERE typname IN ('BugSeverity', 'BugStatus');

-- Count bug reports (should be 0 initially)
SELECT COUNT(*) FROM "BugReport";
```

API Endpoint Tests

```
# Test bug report submission (requires curl or Postman)
curl -X POST https://getcarelinkai.com/api/bug-reports \
  -H "Content-Type: application/json" \
  -d '{
    "title": "Test Bug",
    "description": "This is a test bug report",
    "severity": "LOW",
    "pageUrl": "https://getcarelinkai.com",
    "browserInfo": "Test Browser",
    "userEmail": "test@example.com",
    "userName": "Test User"
  }'

# Expected response: {"success":true,"id":"..."}
```

Rollback Procedure

If Issues Arise:

1. **Immediate Rollback**:
   ```
   bash
      # Revert to previous commit
      git revert 68cebdf
      git push origin main
   ```

2. **Database Rollback** (if needed):
   ```
   bash
      # Connect to Render shell
      # Drop BugReport table
      npx prisma migrate resolve --rolled-back 20260102212003_add_bug_report_system
   ```

3. **Monitor Logs**:
   - Check Render logs for errors
   - Review database connection issues
   - Verify API endpoint errors

Troubleshooting

Common Issues

1. Migration Fails

Symptoms: "P3006: Migration failed to apply"

Solution:

```
# Connect to Render shell
npx prisma migrate resolve --applied 20260102212003_add_bug_report_system
npx prisma migrate deploy
```

2. Button Not Visible

Symptoms: Floating button doesn't appear

Solution:
- Clear browser cache
- Check browser console for errors
- Verify component imported in layout.tsx

3. Admin Dashboard 403 Error

Symptoms: “Unauthorized. Admin access required.”

Solution:
- Verify user role is ADMIN in database
- Check session authentication
- Review API endpoint permissions

4. Form Submission Fails

Symptoms: “Failed to create bug report”

Solution:
- Check API endpoint logs
- Verify Prisma client generated
- Check database connection

Monitoring

Key Metrics to Monitor

1. **Bug Report Submissions**:
   - Track count of new reports
   - Monitor submission errors
   - Check email notifications sent

2. **API Performance**:
   - Response times for /api/bug-reports
   - Error rates
   - Database query performance

3. **User Engagement**:
   - How many users click the button
   - Completion rate of bug submissions
   - Time to admin response

Logging

All bug reports are logged to console:

```
New bug report created: <bug_id>
```

Email notifications (when implemented):

```
Sending bug report email notification: { to, subject, bugId, severity }
```

Success Criteria

- ✅ Deployment completes without errors
- ✅ Migration applies successfully
- ✅ Bug report button visible on all pages
- ✅ Users can submit bug reports
- ✅ Admin can view and manage reports
- ✅ Mobile functionality works
- ✅ No 500 errors or crashes

Next Steps After Deployment

1. Email Notifications (Optional)

Configure email service for admin notifications:

```
# Add to Render environment variables
SMTP_HOST=smtp.gmail.com
SMTP_PORT=587
SMTP_USER=your-email@gmail.com
SMTP_PASSWORD=your-app-password
BUG_REPORT_EMAIL=profyt7@gmail.com
```

2. Monitor Beta Testing

- Check bug reports daily
- Respond to high severity issues quickly
- Track common issues for patterns
- Update status as bugs are fixed

3. Iterate Based on Feedback

- Enhance form based on user feedback
- Add new fields if needed
- Improve admin dashboard features
- Consider adding email templates

Support Contacts

- **Developer**: profyt7@gmail.com
- **GitHub Repo**: profyt7/carelinkai
- **Production URL**: https://getcarelinkai.com
- **Admin Dashboard**: https://getcarelinkai.com/admin/bug-reports

Deployment Timeline

- **Development Completed**: January 2, 2026
- **Committed to GitHub**: January 2, 2026, 21:20 UTC
- **Deployed to Production**: [To be completed by Render auto-deploy]
- **Estimated Deploy Time**: ~10 minutes from push

Status: 🚀 Ready for Deployment
Last Updated: January 2, 2026
Next Action: Monitor Render deployment logs and verify functionality