CareBot Deployment Status

Date: December 30, 2025
Status: ✅ Code Pushed to GitHub Successfully

1. Push Summary

Commits Pushed

- **Latest Commit:** `1726077` - System commit
- **CareBot Commit:** `470d2ce` - Add CareBot - 24/7 AI chatbot for family assistance
- **Previous:** `c9f8ca9` - Add AI tour recommendation UI enhancements

Push Details

```
From: c9f8ca9
To: 1726077
Branch: main → origin/main
Repository: profyt7/carelinkai
Status: SUCCESS
```

2. CareBot Features Deployed

Core Components

1. **Chat Interface** (`src/components/family/carebot/ChatInterface.tsx`)
   - Real-time messaging UI
   - Message history display
   - Typing indicators
   - Auto-scroll functionality

2. **Message Input** (`src/components/family/carebot/MessageInput.tsx`)
   - Text input with send button
   - Enter key submission
   - Disabled state during API calls

3. **Welcome Screen** (`src/components/family/carebot/WelcomeScreen.tsx`)
   - Greeting message
   - Suggested questions
   - One-click question templates

4. **CareBot Page** (`src/app/dashboard/carebot/page.tsx`)
   - Full-page chat experience
   - DashboardLayout integration
   - Session management

5. **API Endpoint** (`src/app/api/carebot/chat/route.ts`)
   - OpenAI GPT-4 integration
   - Context-aware responses
   - Session history tracking
   - Error handling

Key Features

- ✅ 24/7 AI-powered assistance
- ✅ Assisted living information
- ✅ Tour scheduling guidance
- ✅ Care level explanations
- ✅ Resident support queries
- ✅ Real-time responses
- ✅ Context-aware conversations

---

3. Render Deployment Monitoring

Automatic Deployment Trigger

Based on your Render settings, the deployment should be triggered automatically by the GitHub push.

Expected Deployment Steps

1. **Build Phase**
   - Render detects new commits
   - Runs `bash render-build.sh`
   - Installs dependencies
   - Generates Prisma Client
   - Applies database migrations

2. **Start Phase**
   - Runs `npm start`
   - Next.js server starts
   - Health checks pass

3. **Live Phase**
   - Application accessible at production URL
   - CareBot endpoint active at `/api/carebot/chat`

How to Monitor Deployment

Via Render Dashboard

1. Visit: https://dashboard.render.com/web/srv-d3iso13utbr73d5fm1g
2. Navigate to **Deployments** tab
3. Look for latest deployment triggered by commit `470d2ce`
4. Monitor build logs in real-time

Key Indicators

- ✅ **Build Status:** Should show “Live” in green
- ✅ **Deployment Time:** Typically 3-5 minutes

- ⏳ **Current Status:** Check for “In Progress” or “Live”

---

4. Verification Checklist

Post-Deployment Checks

1. Application Access

- [ ] Main application loads without errors
- [ ] Dashboard accessible
- [ ] No 404 errors on CareBot routes

2. CareBot Functionality

- [ ] Navigate to `/dashboard/carebot`
- [ ] Welcome screen displays correctly
- [ ] Can send messages
- [ ] Receives AI responses
- [ ] Message history persists in session

3. API Endpoint

- [ ] `/api/carebot/chat` returns 200 status
- [ ] OpenAI integration working
- [ ] Response times < 10 seconds
- [ ] Error handling functional

4. Database

- [ ] No migration errors in logs
- [ ] Prisma Client generated correctly
- [ ] All existing features still working

---

5. Environment Variables Required

Ensure these are set in Render:

```
# Required for CareBot
OPENAI_API_KEY=sk-proj-...  # Your OpenAI API key

# Existing variables (should already be set)
DATABASE_URL=postgresql://...
NEXTAUTH_SECRET=...
NEXTAUTH_URL=...
```

⚠️ **Critical:** If `OPENAI_API_KEY` is not set, CareBot will fail with 500 errors.

---

6. Testing Instructions

Manual Testing Steps

1. **Open CareBot**
   `Navigate to: https://your-app.onrender.com/dashboard/carebot`

2. **Test Welcome Screen**
   - Verify suggested questions display
   - Click a suggested question
   - Check if message sends

3. **Test Custom Messages**
   - Type: “What services do you offer?”
   - Press Enter or click Send
   - Wait for AI response (should appear within 5-10 seconds)

4. **Test Context Awareness**
   - Ask follow-up question: “Tell me more about memory care”
   - Verify response is contextually relevant

5. **Test Error Handling**
   - Send empty message (should be prevented)
   - Send very long message (should work)

Expected Behavior

- **Response Time:** 3-10 seconds per message
- **Message Format:** Clean markdown rendering
- **Session Persistence:** Messages stay during page refresh
- **Error Messages:** User-friendly error display if API fails

---

7. Troubleshooting

If Deployment Fails

Check Build Logs

1. Go to Render dashboard → Deployments → Latest build
2. Look for error messages in build phase
3. Common issues:
   - Missing environment variables
   - Prisma migration errors
   - TypeScript compilation errors

Check Start Logs

1. Look for runtime errors after build succeeds
2. Common issues:
   - Database connection failures
   - Missing API keys
   - Port binding errors

Rollback Plan

If deployment fails critically:

```
# Revert to previous commit
git revert 470d2ce 1726077
git push origin main
```

If CareBot Doesn't Work

Symptom: 404 Error on `/dashboard/carebot`

- **Cause:** Page not deployed correctly
- **Fix:** Check Next.js build logs for route compilation errors

Symptom: 500 Error on `/api/carebot/chat`

- **Cause:** Missing `OPENAI_API_KEY` or API error
- **Fix:**
  1. Verify environment variable in Render
  2. Check API endpoint logs
  3. Test OpenAI API key validity

Symptom: No AI Responses

- **Cause:** OpenAI API rate limit or quota exceeded
- **Fix:**
  1. Check OpenAI dashboard for usage limits
  2. Verify API key has GPT-4 access
  3. Check Render logs for specific error messages

---

8. Monitoring & Logs

Key Logs to Watch

Application Logs

```
# In Render dashboard → Logs tab
- Look for: "Starting Next.js server..."
- Success: "Ready on port 10000"
- Errors: Any "Error:" or "Failed:" messages
```

CareBot API Logs

```
# Filter logs for: /api/carebot/chat
- Successful requests: "POST /api/carebot/chat 200"
- Failed requests: "POST /api/carebot/chat 500"
- API errors: "OpenAI API error:"
```

Health Check Endpoint

```
# Render automatically checks:
GET / → Should return 200 status
```

9. Next Steps

Immediate Actions

1. ✅ Code pushed to GitHub
2. ⏳ Monitor Render deployment (check dashboard)
3. ⏳ Verify deployment completes successfully
4. ⏳ Test CareBot functionality
5. ⏳ Confirm no errors in production logs

Post-Deployment

1. Test CareBot with real user queries
2. Monitor OpenAI API usage
3. Gather user feedback
4. Plan iterative improvements

Future Enhancements

- Add message persistence to database
- Implement conversation history across sessions
- Add user satisfaction ratings
- Integrate CareBot with tour scheduling
- Add multilingual support

10. Success Criteria

CareBot deployment is successful when:
- ✅ All commits pushed to GitHub
- ✅ Render deployment completes without errors
- ✅ CareBot page loads at `/dashboard/carebot`
- ✅ AI responses are generated successfully
- ✅ No 500 errors in production logs
- ✅ Response times are acceptable (< 10 seconds)
- ✅ Existing features remain functional

11. Contact & Support

Render Support

- Dashboard: https://dashboard.render.com
- Logs: Real-time in Render dashboard
- Webhook: Configure in Settings → Build & Deploy

GitHub Repository

- Repo: https://github.com/profyt7/carelinkai
- Latest Commit: 1726077

- CareBot Commit: 470d2ce

OpenAI Support

- Dashboard: https://platform.openai.com
- Check API usage and limits
- Verify GPT-4 access

Status Updated: December 30, 2025
Next Review: After Render deployment completes
Deployment Trigger: Automatic via GitHub webhook