✅ GitHub Push Complete - Next Steps

🎉 Push Status: SUCCESSFUL

Date: December 30, 2025
Repository: profyt7/carelinkai
Branch: main
Latest Commit: 91eefc0

📦 What Was Pushed

Commit 1: `4c0d980` - Automated Follow-ups Implementation

Files Pushed:
- ✅ `.github/workflows/process-followups.yml` - GitHub Actions workflow
- ✅ `AUTOMATED_FOLLOWUPS_SETUP_GUIDE.md` - Setup documentation
- ✅ `AUTOMATED_FOLLOWUPS_SETUP_GUIDE.pdf` - PDF version
- ✅ `package.json` & `package-lock.json` - Resend dependency
- ✅ `src/lib/email/inquiry-email-service.ts` - Email service updates
- ✅ `prisma/schema.prisma` - Schema validation
- ✅ `scripts/create-test-documents.ts` - Test script

Commit 2: `91eefc0` - Documentation Update

Files Pushed:
- ✅ `GITHUB_PUSH_SUCCESS.md` - Push success summary (token redacted)

🔍 Verification Completed

✅ Git Status

```
On branch main
Your branch is up to date with 'origin/main'
```

✅ Workflow File Present

```
.github/workflows/process-followups.yml
- Size: 1,230 bytes
- Modified: Dec 30 18:26
- Status: Pushed to GitHub
```

✅ Workflow Configuration

```yaml
name: Process Follow-ups
schedule: '0 */6 * * *'  # Every 6 hours
manual_trigger: workflow_dispatch enabled
endpoint: /api/follow-ups/process
authentication: CRON_SECRET required
```

🎯 NEXT STEPS - Action Required

1. Configure GitHub Secret ⚠️ REQUIRED

The workflow needs a `CRON_SECRET` to authenticate with your API endpoint.

Steps:
1. Go to: https://github.com/profyt7/carelinkai/settings/secrets/actions
2. Click **“New repository secret”**
3. Name: `CRON_SECRET`
4. Value: Generate a secure random string (or use the same one you’ll set on Render)
`bash`
`  # Generate a secure secret (run locally or on Render):`
`  openssl rand -base64 32`
5. Click **“Add secret”**

---

2. Configure Render Environment Variables ⚠️ REQUIRED

Add these environment variables to your Render service:

Variable	Value	Where to Get
`RESEND_API_KEY`	Your Resend API key	https://resend.com/api-keys
`CRON_SECRET`	Same value as GitHub secret	Use same value from step 1
`EMAIL_FROM`	Your sender email	e.g., `noreply@carelinkai.com`

Steps:
1. Go to: https://dashboard.render.com
2. Select your `carelinkai` service
3. Navigate to **Environment** tab
4. Add each variable above
5. Click **“Save Changes”**
6. Render will automatically redeploy

---

3. Verify GitHub Actions Workflow

Check if workflow appears:
1. Visit: https://github.com/profyt7/carelinkai/actions
2. Look for **“Process Follow-ups”** workflow in the left sidebar
3. Verify it shows:
- ✅ Workflow file found
- ✅ No syntax errors
- ✅ Schedule: `0 */6 * * *`
- ✅ Manual trigger enabled

Expected appearance:
- Workflow name: “Process Follow-ups”
- Status: Ready (no runs yet, waiting for first scheduled trigger)
- Next scheduled run: Within 6 hours from now

---

4. Test the Workflow Manually (Optional but Recommended)

Before waiting for the scheduled run, test it manually:

Steps:
1. Go to: https://github.com/profyt7/carelinkai/actions/workflows/process-followups.yml
2. Click **“Run workflow”** button
3. Select branch: `main`
4. Click **“Run workflow”** to confirm
5. Watch the workflow execution in real-time
6. Check for:
- ✅ Successful checkout
- ✅ API call returns 200 status
- ✅ Response shows processed inquiries

Expected output:

```
HTTP Status: 200
Response: {
  "success": true,
  "processed": X,
  "sent": Y,
  "message": "Follow-up processing completed"
}
Follow-ups processed successfully
```

---

5. Create Test Inquiry

To fully test the system, create a test inquiry:

```
# Create test inquiry
curl -X POST https://carelinkai.onrender.com/api/inquiries \
  -H "Content-Type: application/json" \
  -d '{
    "firstName": "Test",
    "lastName": "User",
    "email": "your-email@example.com",
    "phone": "555-0123",
    "message": "Test inquiry for automated follow-up system",
    "facilityInterest": "Memory Care"
  }'
```

Then wait 6 hours OR trigger the workflow manually to process it.

---

6. Monitor First Execution

After the workflow runs (either manually or on schedule):

Check GitHub Actions Logs

1. Go to: https://github.com/profyt7/carelinkai/actions
2. Click on the latest “Process Follow-ups” run
3. Review logs for:
   - ✅ HTTP 200 response
   - ✅ Number of inquiries processed
   - ✅ Number of emails sent
   - ❌ Any errors or failures

Check Render Logs

1. Go to: https://dashboard.render.com
2. Select your service → **Logs** tab
3. Look for entries like:
   `[Follow-ups] Processing X inquiries for automated follow-up`
   `[Follow-ups] Sent follow-up email to email@example.com`
   `[Follow-ups] ✅ Processing completed: X processed, Y sent`

Check Resend Dashboard

1. Go to: https://resend.com/emails
2. Verify follow-up emails were sent
3. Check delivery status
4. Review bounce/spam reports

Check Database

```sql
-- Verify follow-ups were recorded
SELECT
  id,
  email,
  status,
  createdAt,
  lastFollowUpSent,
  EXTRACT(HOUR FROM (lastFollowUpSent - createdAt)) as hours_until_followup
FROM "Inquiry"
WHERE lastFollowUpSent IS NOT NULL
ORDER BY lastFollowUpSent DESC
LIMIT 10;
```

📋 Configuration Checklist

Track your progress:

GitHub Configuration

- [ ] `CRON_SECRET` added to GitHub repository secrets
- [ ] Workflow appears in Actions tab
- [ ] Workflow syntax validated (no errors)
- [ ] Manual trigger tested successfully

Render Configuration

- [ ] `RESEND_API_KEY` added to environment variables
- [ ] `CRON_SECRET` added (matches GitHub secret)
- [ ] `EMAIL_FROM` configured with valid sender email
- [ ] Service redeployed after adding variables
- [ ] Deployment successful (check logs)

Testing & Verification

- [ ] Test inquiry created in database
- [ ] Workflow triggered manually (first test)
- [ ] Follow-up email received
- [ ] Database updated with `lastFollowUpSent` timestamp
- [ ] Resend dashboard shows sent email
- [ ] Render logs show successful processing

Monitoring Setup

- [ ] GitHub Actions email notifications enabled
- [ ] Render alert rules configured for errors
- [ ] Sentry error tracking verified
- [ ] Weekly review scheduled for follow-up metrics

🔔 Scheduled Execution Details

When Will It Run?

Schedule: Every 6 hours at minute 0
- 12:00 AM UTC (midnight)
- 6:00 AM UTC
- 12:00 PM UTC (noon)
- 6:00 PM UTC

Next Run: The workflow will automatically run at the next scheduled time after:
- GitHub Actions processes the new workflow file (~1-2 minutes)
- The next hour that matches the cron schedule

Example: If pushed at 2:30 PM UTC, first run will be at 6:00 PM UTC.

🚨 Troubleshooting

Workflow Doesn't Appear

Problem: Workflow not showing in GitHub Actions
Solution:
1. Wait 1-2 minutes for GitHub to index the file
2. Refresh the Actions page
3. Check YAML syntax: https://www.yamllint.com/
4. Verify file is in `.github/workflows/` directory
5. Ensure branch is `main` (GitHub Actions may only show workflows from default branch)

Manual Trigger Fails

Problem: Workflow run fails immediately
Solution:
1. Check if `CRON_SECRET` is set in GitHub secrets
2. Verify Render service is running and accessible
3. Check Render environment variables are configured
4. Review Render logs for API endpoint errors

API Returns 401 Unauthorized

Problem: `Authorization: Bearer` token rejected
Solution:
1. Verify `CRON_SECRET` matches between GitHub and Render
2. Check for extra spaces or newlines in secret values
3. Regenerate secret and update both locations

API Returns 500 Error

Problem: Server error during processing
Solution:
1. Check Render logs for detailed error message
2. Verify `RESEND_API_KEY` is valid
3. Check database connection
4. Review Sentry for exception details

Emails Not Sending

Problem: Workflow succeeds but no emails received
Solution:
1. Check Resend dashboard for delivery status
2. Verify `EMAIL_FROM` is a validated domain
3. Check spam folder for test emails
4. Review Resend API key permissions
5. Check inquiry status (must be PENDING and > 6 hours old)

📊 Success Metrics

After 24-48 hours of operation, you should see:

Expected Outcomes

- ✅ 4 successful workflow runs per day (every 6 hours)
- ✅ Follow-up emails sent to inquiries > 6 hours old with status PENDING
- ✅ Database updated with `lastFollowUpSent` timestamps
- ✅ No duplicate emails (system checks for existing `lastFollowUpSent`)
- ✅ Clean logs with no authentication errors

Key Performance Indicators

```
-- Follow-up coverage rate
SELECT
  ROUND(COUNT(*) FILTER (WHERE lastFollowUpSent IS NOT NULL)::numeric /
        COUNT(*)::numeric * 100, 2) as followup_coverage_rate
FROM "Inquiry"
WHERE status = 'PENDING'
AND createdAt < NOW() - INTERVAL '6 hours';

-- Target: > 95%
```

Monitoring Dashboard

Create a simple monitoring query to run daily:

```
-- Daily follow-up summary
SELECT
  DATE(lastFollowUpSent) as date,
  COUNT(*) as emails_sent,
  COUNT(DISTINCT email) as unique_recipients
FROM "Inquiry"
WHERE lastFollowUpSent >= CURRENT_DATE - INTERVAL '7 days'
GROUP BY DATE(lastFollowUpSent)
ORDER BY date DESC;
```

📚 Additional Resources

Documentation Files

- `AUTOMATED_FOLLOWUPS_SETUP_GUIDE.md` - Complete setup guide
- `GITHUB_PUSH_SUCCESS.md` - Push verification summary
- `src/lib/email/inquiry-email-service.ts` - Email service code
- `src/app/api/follow-ups/process/route.ts` - API endpoint code

External Links

- GitHub Actions Documentation (https://docs.github.com/en/actions)
- Resend API Documentation (https://resend.com/docs)
- Render Cron Jobs Guide (https://render.com/docs/cronjobs)
- Cron Expression Generator (https://crontab.guru)

✅ Ready for Production

All code has been successfully pushed to GitHub! The automated follow-up system is now ready for production deployment.

What's Working:
- ✅ GitHub Actions workflow configured and pushed
- ✅ Schedule set to run every 6 hours
- ✅ Manual trigger available for testing
- ✅ Secure authentication with CRON_SECRET
- ✅ Email service integration with Resend
- ✅ Database tracking for follow-up history
- ✅ Comprehensive error handling and logging

Next Action: Configure the required environment variables (step 1 & 2 above) to activate the system.

Document Created: December 30, 2025, 18:45 UTC
Last Push: Commit `91eefc0`
Status: ✅ Code pushed, awaiting configuration