Call Notification System Documentation

Overview

This system sends email notifications for completed voice AI calls based on existing call data stored in the database, instead of relying solely on Twilio webhooks.

Features

- **Database-Driven Notifications**: Send emails based on stored call data
- **Duplicate Prevention**: Tracks which calls have already received email notifications
- **Manual Triggering**: UI and CLI tools for sending notifications on demand
- **Batch Processing**: Process multiple pending notifications at once
- **Error Handling**: Robust error handling with detailed reporting
- **Statistics Dashboard**: View pending, sent, and failed notifications

Architecture

Database Schema

Two new fields were added to the `CallLog` model:

```
model CallLog {
  // ... existing fields ...
  emailSent    Boolean   @default(false)  // Tracks if email was sent
  emailSentAt  DateTime?                  // Timestamp of when email was sent
  // ... other fields ...
}
```

Components

1. **API Route**: `/api/calls/send-notifications`
   - GET: Fetch notification statistics
   - POST: Send pending email notifications

2. **Service Layer**: `lib/call-notification-service.ts`
   - Encapsulates notification logic
   - Reusable across API routes and CLI scripts

3. **UI Dashboard**: `/dashboard/voice-ai/notifications`
   - View notification statistics
   - Manually trigger batch sends
   - View results of last send operation

4. **CLI Script**: `scripts/send-call-notifications.ts`
   - Command-line tool for automation
   - Useful for cron jobs or manual testing

5. **Updated Webhook**: `/api/twilio/call-status`
   - Marks emails as sent when webhook sends them
   - Prevents duplicate notifications

Usage

Through the UI Dashboard

1. Navigate to **Admin > Call Notifications** in the sidebar
2. View current notification statistics
3. Click “Send Pending” to send notifications for calls without emails
4. Click “Resend All” to send notifications for all completed calls (including already sent)

Through the API

Get Statistics

```
curl -X GET \
  http://localhost:3000/api/calls/send-notifications \
  -H 'Cookie: next-auth.session-token=YOUR_SESSION_TOKEN'
```

Response:

```
{
  "pendingCount": 5,
  "totalCompleted": 100,
  "emailsSent": 95
}
```

Send Notifications

```
curl -X POST \
  http://localhost:3000/api/calls/send-notifications \
  -H 'Content-Type: application/json' \
  -H 'Cookie: next-auth.session-token=YOUR_SESSION_TOKEN' \
  -d '{
    "sendAll": false
  }'
```

Response:

```
{
  "success": true,
  "results": {
    "success": 5,
    "failed": 0,
    "skipped": 2,
    "errors": []
  }
}
```

Through CLI

```
# Send all pending notifications
cd /home/ubuntu/go_high_or_show_google_crm/nextjs_space
ts-node scripts/send-call-notifications.ts

# Send notifications for a specific user
ts-node scripts/send-call-notifications.ts --user-id clxxxxxxxx

# Limit the number of notifications sent
ts-node scripts/send-call-notifications.ts --limit 10

# Combine filters
ts-node scripts/send-call-notifications.ts --user-id clxxxxxxxx --limit 5
```

Requirements for Sending

For a call to be eligible for email notification, it must meet ALL of these criteria:

1. **Call Status**: Must be `COMPLETED`
2. **Email Not Sent**: `emailSent` must be `false` (unless using "Resend All")
3. **Voice Agent Settings**:
   - `sendRecordingEmail` must be `true`
   - `recordingEmailAddress` must be configured
4. **Call Data**: Must have either:
   - Transcript data, OR
   - Conversation data

Notification Email Content

Each notification email includes:

- **Caller Information**:
- Name (from Leads database or phone number)
- Phone number
- Email (if available in Leads)

- **Call Details**:

- Date and time
- Duration
- Call reason/purpose (extracted from AI analysis)

- **AI Summary**:

- Conversation summary (if available)
- Full transcript

- **Recording**:

- Link to call recording (if available)

Automation Options

Option 1: Cron Job (Recommended)

Set up a cron job to automatically send pending notifications:

```
# Edit crontab
crontab -e

# Add this line to run every hour
0 * * * * cd /home/ubuntu/go_high_or_show_google_crm/nextjs_space && ts-node scripts/
send-call-notifications.ts >> /var/log/call-notifications.log 2>&1

# Or run every 15 minutes
*/15 * * * * cd /home/ubuntu/go_high_or_show_google_crm/nextjs_space && ts-node script
s/send-call-notifications.ts >> /var/log/call-notifications.log 2>&1
```

Option 2: API Webhook

Create a webhook endpoint that can be triggered externally:

```
// Example webhook implementation
app.post('/api/webhooks/send-notifications', async (req, res) => {
  // Verify webhook secret
  if (req.headers['x-webhook-secret'] !== process.env.WEBHOOK_SECRET) {
    return res.status(401).json({ error: 'Unauthorized' });
  }

  // Trigger notification sending
  const results = await callNotificationService.sendPendingNotifications();

  return res.json({ success: true, results });
});
```

Option 3: Scheduled Task (Node.js)

Use node-cron or similar library:

```
import cron from 'node-cron';
import { callNotificationService } from './lib/call-notification-service';

// Run every hour
cron.schedule('0 * * * *', async () => {
  console.log('Running scheduled notification send...');
  const results = await callNotificationService.sendPendingNotifications();
  console.log('Results:', results);
});
```

Troubleshooting

Notifications Not Sending

1. **Check Voice Agent Settings**:
   - Verify `sendRecordingEmail` is enabled
   - Verify `recordingEmailAddress` is configured

2. **Check Call Data**:
   - Verify call has transcript or conversation data
   - Check `CallLog.status` is `COMPLETED`

3. **Check Email Service**:
   - Verify SendGrid API key is configured
   - Check email service logs for errors

4. **Check Database**:
   `sql`
   ```
     -- Find calls eligible for notifications
     SELECT
       cl.id,
       cl.status,
       cl.emailSent,
       va.sendRecordingEmail,
       va.recordingEmailAddress,
       LENGTH(cl.transcription) as has_transcript,
       LENGTH(cl.conversationData) as has_conversation_data
     FROM "CallLog" cl
     JOIN "VoiceAgent" va ON cl."voiceAgentId" = va.id
     WHERE cl.status = 'COMPLETED'
     AND cl."emailSent" = false;
   ```

Duplicate Emails

- The system tracks sent emails using the `emailSent` flag
- Webhook automatically marks emails as sent
- Manual sends respect this flag (unless using “Resend All”)

Performance Issues

- Use the `--limit` parameter to process notifications in batches
- Consider running during off-peak hours
- Monitor database query performance

Migration Notes

Existing Calls

Calls completed before this system was deployed have `emailSent = false` by default. If you don’t want to send notifications for these old calls, you can:

1. **Mark all existing calls as sent**:
   `sql`
   ```
     UPDATE "CallLog"
     SET "emailSent" = true,
         "emailSentAt" = NOW()
     WHERE status = 'COMPLETED'
     AND "emailSent" = false;
   ```

2. **Or filter by date when sending**:
   ```javascript
   ```

```
    // Only send for calls after a certain date
    const cutoffDate = new Date(‘2024-12-01’);

const callLogs = await prisma.callLog.findMany({
where: {
status: ‘COMPLETED’,
emailSent: false,
createdAt: { gte: cutoffDate },
// ... other filters
}
});
```

Best Practices

1. **Monitor Regularly**: Check the notifications dashboard daily
2. **Set Up Alerts**: Create alerts for failed notifications
3. **Test Thoroughly**: Use the “Resend All” feature with test data first
4. **Log Everything**: Keep logs of notification sends for auditing
5. **Handle Errors Gracefully**: The system continues processing even if individual sends fail

Future Enhancements

Potential improvements to consider:

- [ ] Email templates customization
- [ ] Notification preferences per user
- [ ] Retry logic for failed sends
- [ ] Email delivery status tracking
- [ ] Bulk resend by date range
- [ ] Webhook for notification failures
- [ ] Integration with other notification channels (SMS, Slack, etc.)

Support

For issues or questions:

1. Check the logs: `/var/log/call-notifications.log`
2. Check the UI dashboard for error messages
3. Review this documentation
4. Contact the development team