Automated Follow-up System - Feature #4 Phase 3

Overview

The Automated Follow-up System intelligently schedules and sends follow-up communications to families based on inquiry stage, urgency, and engagement patterns. This system uses a rules-based engine to determine optimal follow-up timing and channels (email, SMS).

Architecture

Components

1. **Follow-up Rules Engine** (`src/lib/followup/followup-rules.ts`)
   - Defines rules for automated follow-up scheduling
   - Evaluates which rules apply to a given inquiry
   - Supports conditional logic based on inquiry properties

2. **Follow-up Scheduler** (`src/lib/followup/followup-scheduler.ts`)
   - Schedules follow-ups based on rules
   - Manages manual follow-ups
   - Handles cancellation and rescheduling

3. **SMS Service** (`src/lib/sms/sms-service.ts`)
   - Integrates with Twilio for SMS delivery
   - Formats messages appropriately
   - Handles phone number formatting

4. **Follow-up Processor** (`src/lib/followup/followup-processor.ts`)
   - Processes due follow-ups in background
   - Handles multi-channel delivery
   - Updates follow-up status

5. **Inquiry Hooks** (`src/lib/hooks/inquiry-hooks.ts`)
   - Auto-schedules follow-ups when inquiries are created
   - Re-evaluates follow-ups when inquiry stage changes

Features

1. Rules-Based Scheduling

The system includes 7 default follow-up rules:

Rule	Trigger	Action	Timing	Priority
Urgent Inquiry Immediate Follow-up	New inquiry with URGENT urgency	SMS	1 hour	HIGH
New Inquiry First Follow-up	24 hours after initial contact	Email	24 hours	MEDIUM
Second Follow-up	No response after 3 days	Email	72 hours	MEDIUM
Third Follow-up	No response after 7 days	Email	168 hours	LOW
Tour Reminder	1 day before scheduled tour	SMS	24 hours before	HIGH
Post-Tour Follow-up	1 day after tour	Email	48 hours after	HIGH
High Urgency No Response	High urgency with no response for 2 days	SMS	48 hours	HIGH

2. Multi-Channel Delivery

- **Email**: Uses existing SMTP configuration
- **SMS**: Integrates with Twilio (optional)
- **Phone Call**: Creates task for staff
- **Task**: Creates actionable item for staff

3. Auto-Scheduling

Follow-ups are automatically scheduled when:
- A new inquiry is created
- An inquiry stage changes
- Inquiry urgency is updated

4. Manual Override

Staff can:
- Schedule custom follow-ups
- Cancel scheduled follow-ups
- Reschedule follow-ups
- Mark follow-ups as completed

5. Background Processing

Follow-ups are processed via cron job that:
- Runs every 15 minutes (configurable)
- Sends due follow-ups

- Updates overdue follow-ups
- Tracks delivery status

API Endpoints

1. Schedule Manual Follow-up

```
POST /api/inquiries/:id/follow-ups
```

Request:

```
{
  "scheduledFor": "2025-12-20T10:00:00Z",
  "type": "EMAIL",
  "subject": "Optional subject",
  "content": "Optional custom message"
}
```

Response:

```
{
  "success": true,
  "followUp": {
    "id": "clx...",
    "inquiryId": "clx...",
    "scheduledFor": "2025-12-20T10:00:00Z",
    "type": "EMAIL",
    "status": "PENDING",
    "createdAt": "2025-12-18T10:00:00Z"
  }
}
```

2. Get Follow-ups for Inquiry

```
GET /api/inquiries/:id/follow-ups?status=PENDING
```

Response:

```
{
  "success": true,
  "followUps": [
    {
      "id": "clx...",
      "type": "EMAIL",
      "scheduledFor": "2025-12-20T10:00:00Z",
      "status": "PENDING",
      "subject": "Follow-up on your inquiry",
      "createdAt": "2025-12-18T10:00:00Z"
    }
  ]
}
```

3. Update Follow-up

```
PATCH /api/follow-ups/:id
```

Request (Cancel):

```
{
  "action": "cancel"
}
```

Request (Reschedule):

```
{
  "action": "reschedule",
  "scheduledFor": "2025-12-21T10:00:00Z"
}
```

Request (Complete):

```
{
  "action": "complete"
}
```

4. Delete Follow-up

```
DELETE /api/follow-ups/:id
```

5. Process Due Follow-ups (Cron)

```
POST /api/follow-ups/process
Authorization: Bearer {CRON_SECRET}
```

This endpoint should be called by a cron job every 15 minutes.

Configuration

Environment Variables

Add the following to your `.env` file:

```
# Twilio Configuration (optional - for SMS)
TWILIO_ACCOUNT_SID=your-twilio-account-sid
TWILIO_AUTH_TOKEN=your-twilio-auth-token
TWILIO_PHONE_NUMBER=+1234567890

# Cron Job Secret (for automated processing)
CRON_SECRET=your-secure-random-secret
```

Twilio Setup

1. Create a Twilio account at https://www.twilio.com

2. Get your Account SID and Auth Token from the console
3. Purchase a phone number for SMS sending
4. Add credentials to environment variables

Email Setup

The system uses the existing SMTP configuration:

```
SMTP_HOST=smtp.gmail.com
SMTP_PORT=587
SMTP_USER=your-email@gmail.com
SMTP_PASS=your-app-password
SMTP_FROM=noreply@carelinkai.com
```

Cron Job Setup

Option 1: Render.com Cron Jobs

1. Go to your Render service dashboard
2. Navigate to “Settings” → “Cron Jobs”
3. Add new cron job:
   - **Command**: `curl -X POST https://carelinkai.onrender.com/api/follow-ups/process -H "Authorization: Bearer YOUR_CRON_SECRET"`
   - **Schedule**: `*/15 * * * *` (every 15 minutes)

Option 2: External Cron Service (cron-job.org)

1. Sign up at https://cron-job.org
2. Create new cron job:
   - **URL**: `https://carelinkai.onrender.com/api/follow-ups/process`
   - **Schedule**: Every 15 minutes
   - **Request Method**: POST
   - **Headers**: `Authorization: Bearer YOUR_CRON_SECRET`

Option 3: Vercel Cron (if using Vercel)

Add to `vercel.json`:

```
{
  "crons": [{
    "path": "/api/follow-ups/process",
    "schedule": "*/15 * * * *"
  }]
}
```

Usage Examples

Auto-scheduling on Inquiry Creation

When an inquiry is created, follow-ups are automatically scheduled:

```
// This happens automatically in POST /api/inquiries
const inquiry = await prisma.inquiry.create({ data: {...} });
await afterInquiryCreated(inquiry.id);
// Follow-ups are now scheduled based on rules
```

Manual Follow-up Scheduling

```
import { followUpScheduler } from '@/lib/followup/followup-scheduler';

// Schedule a manual follow-up
await followUpScheduler.scheduleManualFollowUp(
  'inquiry-id',
  new Date('2025-12-20T10:00:00Z'),
  'EMAIL',
  'Custom follow-up message'
);
```

Cancel Follow-up

```
await followUpScheduler.cancelFollowUp('followup-id');
```

Reschedule Follow-up

```
await followUpScheduler.rescheduleFollowUp(
  'followup-id',
  new Date('2025-12-21T10:00:00Z')
);
```

Process Due Follow-ups (Background Job)

```
import { followUpProcessor } from '@/lib/followup/followup-processor';

await followUpProcessor.processDueFollowUps();
```

Custom Rules

You can create custom follow-up rules:

```typescript
import { FollowUpRulesEngine, FollowUpRule } from '@/lib/followup/followup-rules';

const customRules: FollowUpRule[] = [
  {
    name: 'VIP Referral Immediate Contact',
    description: 'Immediate follow-up for referral source inquiries',
    conditions: {
      source: ['REFERRAL'],
      urgency: ['HIGH', 'URGENT'],
    },
    action: {
      type: 'SMS',
      delayHours: 0.5, // 30 minutes
      priority: 'HIGH',
    },
  },
  // Add more custom rules...
];

const customEngine = new FollowUpRulesEngine(customRules);
```

Database Schema

The system uses the following models from the Prisma schema:

FollowUp Model

```
model FollowUp {
  id            String          @id @default(cuid())
  inquiryId     String
  scheduledFor  DateTime
  type          FollowUpType    @default(EMAIL)
  subject       String?
  content       String?         @db.Text
  status        FollowUpStatus  @default(PENDING)
  completedAt   DateTime?
  completedBy   String?
  metadata      Json?
  inquiry       Inquiry         @relation(fields: [inquiryId], references: [id], onDel
ete: Cascade)
  createdAt     DateTime        @default(now())
  updatedAt     DateTime        @updatedAt
}
```

InquiryResponse Model

```
model InquiryResponse {
  id          String          @id @default(cuid())
  inquiryId   String
  content     String          @db.Text
  type        ResponseType    @default(AI_GENERATED)
  channel     ResponseChannel @default(EMAIL)
  sentBy      String?
  sentAt      DateTime?
  status      ResponseStatus  @default(DRAFT)
  metadata    Json?
  subject     String?
  toAddress   String?
  inquiry     Inquiry         @relation(fields: [inquiryId], references: [id], onDelet
e: Cascade)
  createdAt   DateTime        @default(now())
  updatedAt   DateTime        @updatedAt
}
```

Monitoring and Analytics

Follow-up Metrics

Track follow-up performance:

```
// Get follow-up completion rate
const stats = await prisma.followUp.groupBy({
  by: ['status'],
  _count: true,
});

// Get average response time
const avgResponseTime = await prisma.inquiryResponse.aggregate({
  where: {
    type: 'AUTOMATED',
  },
  _avg: {
    // Calculate time between scheduled and sent
  },
});
```

Debugging

Enable detailed logging:

```
// In follow-up processor
console.log('Processing follow-up:', followUp.id);
console.log('Inquiry details:', followUp.inquiry);
console.log('Generated content:', content);
```

Best Practices

1. **Test SMS Service**: Verify Twilio configuration before production
2. **Monitor Delivery**: Check follow-up status regularly

3. **Adjust Rules**: Optimize based on engagement data
4. **Respect Preferences**: Honor contact method preferences
5. **Opt-out Handling**: Implement SMS opt-out mechanism
6. **Rate Limiting**: Be mindful of SMS/email rate limits
7. **Content Quality**: Ensure AI-generated content is reviewed
8. **Timing Optimization**: Adjust timing based on response patterns

Troubleshooting

Follow-ups Not Sending

Problem: Follow-ups are scheduled but not being sent

Solutions:
- Verify cron job is running
- Check SMTP/Twilio credentials
- Review follow-up status in database
- Check application logs for errors

```
-- Check pending follow-ups
SELECT * FROM "FollowUp"
WHERE status = 'PENDING'
AND "scheduledFor" <= NOW();
```

SMS Not Delivering

Problem: SMS messages fail to send

Solutions:
- Verify Twilio account balance
- Check phone number format (E.164)
- Verify phone number is not blocked
- Check Twilio console for delivery logs
- Ensure SMS service is configured (`smsService.isConfigured()`)

Duplicate Follow-ups

Problem: Multiple follow-ups scheduled for same inquiry

Solutions:
- Review rule conditions for conflicts
- Check deduplication logic in scheduler
- Verify hook is not called multiple times

```
// The scheduler checks for existing follow-ups
const existingFollowUp = await prisma.followUp.findFirst({
  where: {
    inquiryId,
    type: rule.action.type,
    status: 'PENDING',
    scheduledFor: { gte: new Date(), lte: scheduledFor },
  },
});
```

Overdue Follow-ups

Problem: Follow-ups marked as OVERDUE

Solutions:
- Ensure cron job runs frequently enough
- Check processing time for follow-ups
- Review system load during processing

Security Considerations

1. **CRON_SECRET**: Use a strong, random secret
2. **Rate Limiting**: Implement rate limits on API endpoints
3. **Access Control**: Verify user permissions for follow-up management
4. **Data Privacy**: Ensure follow-up content respects privacy rules
5. **Audit Logging**: Log all follow-up actions for compliance

Future Enhancements

- [ ] Email open tracking
- [ ] Link click tracking
- [ ] A/B testing for messages
- [ ] Machine learning for optimal timing
- [ ] WhatsApp integration
- [ ] Voice call automation
- [ ] Sentiment analysis on responses
- [ ] Automated response to replies
- [ ] Integration with CRM systems

Support

For issues or questions about the Automated Follow-up System:

1. Check this documentation
2. Review application logs
3. Check database for follow-up status
4. Contact development team

Changelog

Version 1.0.0 (December 18, 2025)

- Initial release
- 7 default follow-up rules
- Email and SMS delivery
- Auto-scheduling on inquiry creation
- Manual override capabilities
- Background processing via cron
- Comprehensive API endpoints