Feature #4 Phase 3: Automated Follow-up System - Implementation Summary

Overview

This document summarizes the implementation of Phase 3 of Feature #4: AI-Powered Inquiry Response & Follow-up System. This phase adds intelligent automated follow-up scheduling and multi-channel delivery.

Date Completed

December 18, 2025

Implemented Components

1. Follow-up Rules Engine (`src/lib/followup/followup-rules.ts`)

- Created rules-based system for automated follow-up scheduling
- Implemented 7 default rules covering various scenarios
- Supports conditional logic based on:
- Inquiry stage
- Urgency level
- Source
- Time since inquiry/last contact
- No response days

Key Features:
- Rule evaluation and matching
- Priority-based rule sorting
- Flexible condition system
- Support for custom rules

2. Follow-up Scheduler (`src/lib/followup/followup-scheduler.ts`)

- Automatic follow-up scheduling based on rules
- Manual follow-up scheduling
- Follow-up cancellation and rescheduling
- Deduplication to prevent duplicate follow-ups
- Query for due follow-ups

Key Methods:
- `scheduleFollowUps(inquiryId)` : Auto-schedule based on rules
- `scheduleManualFollowUp(...)` : Manual scheduling
- `cancelFollowUp(followUpId)` : Cancel follow-up
- `rescheduleFollowUp(followUpId, newDate)` : Reschedule
- `getDueFollowUps()` : Get pending due follow-ups

3. SMS Service (`src/lib/sms/sms-service.ts`)

- Twilio integration for SMS delivery
- Graceful handling of missing configuration
- Phone number formatting (E.164)
- Tour reminder messages
- Follow-up SMS formatting

Key Features:
- Optional dependency (works without Twilio)
- Automatic phone number formatting
- Configuration check before sending
- Error handling and logging

4. Follow-up Processor (`src/lib/followup/followup-processor.ts`)

- Background processing of due follow-ups
- Multi-channel delivery (Email, SMS, Phone, Task)
- AI content generation integration
- Status tracking and logging
- Overdue follow-up management

Processing Flow:
1. Fetch due follow-ups
2. Generate content if needed (using AI)
3. Send via appropriate channel
4. Update status
5. Create response record

5. Inquiry Hooks (`src/lib/hooks/inquiry-hooks.ts`)

- Auto-scheduling on inquiry creation
- Re-evaluation on stage change
- Re-evaluation on urgency change
- Non-blocking execution

Hooks:
- `afterInquiryCreated(inquiryId)` : Schedule follow-ups for new inquiry
- `afterInquiryStageChanged(inquiryId, newStage)` : Re-schedule on stage change
- `afterInquiryUpdated(inquiryId, oldData, newData)` : Handle updates

6. API Endpoints

Created:
- `POST /api/follow-ups/process` : Process due follow-ups (cron endpoint)
- `PATCH /api/follow-ups/:id` : Update follow-up (cancel, reschedule, complete)
- `DELETE /api/follow-ups/:id` : Delete follow-up

Updated:
- `POST /api/inquiries/:id/follow-ups` : Already existed, kept as-is
- `GET /api/inquiries/:id/follow-ups` : Already existed, kept as-is

7. Inquiry API Integration

Updated Files:
- `src/app/api/inquiries/route.ts`: Added `afterInquiryCreated` hook
- `src/app/api/inquiries/[id]/route.ts`: Added `afterInquiryUpdated` hook

Default Follow-up Rules

#	Rule Name	Trigger	Action	Timing	Priority
1	Urgent Inquiry Immediate Follow-up	New + URGENT	SMS	1 hour	HIGH
2	New Inquiry First Follow-up	Contacted + 1 day	Email	24 hours	MEDIUM
3	Second Follow-up	Contacted + no response 3 days	Email	72 hours	MEDIUM
4	Third Follow-up	Contacted + no response 7 days	Email	168 hours	LOW
5	Tour Reminder	Tour scheduled	SMS	24 hours before	HIGH
6	Post-Tour Follow-up	Tour + 2 days after	Email	48 hours after	HIGH
7	High Urgency No Response	High urgency + no response 2 days	SMS	48 hours	HIGH

Dependencies Installed

```
npm install twilio --legacy-peer-deps
npm install --save-dev @types/twilio --legacy-peer-deps
```

Configuration

Environment Variables Added

```
# 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
```

Required Setup

1. **Twilio Account** (optional for SMS):
   - Sign up at https://www.twilio.com
   - Get Account SID and Auth Token
   - Purchase phone number
   - Add credentials to `.env`

2. **Cron Job**:
   - Set up cron job to call `/api/follow-ups/process`
   - Recommended: Every 15 minutes
   - Use Render Cron Jobs, cron-job.org, or Vercel Cron

Documentation

- **Comprehensive Guide**: `docs/AUTOMATED_FOLLOWUP_SYSTEM.md`
- **API Reference**: Included in guide
- **Configuration Guide**: Included in guide
- **Troubleshooting**: Included in guide

Testing

Manual Testing Steps

1. **Create Inquiry**:
```
bash
   POST /api/inquiries
   # Verify follow-ups are scheduled
```

2. **Check Follow-ups**:
```
bash
   GET /api/inquiries/:id/follow-ups
   # Should see scheduled follow-ups
```

3. **Process Follow-ups**:
```
bash
   POST /api/follow-ups/process
   # Authorization: Bearer {CRON_SECRET}
```

4. **Verify Delivery**:
   - Check email inbox

- Check Twilio console for SMS delivery
- Check database for response records

Database Queries for Testing

```sql
-- Check scheduled follow-ups
SELECT * FROM "FollowUp" WHERE status = 'PENDING' ORDER BY "scheduledFor";

-- Check sent responses
SELECT * FROM "InquiryResponse" WHERE type = 'AUTOMATED' ORDER BY "createdAt" DESC;

-- Check follow-up statistics
SELECT status, COUNT(*) FROM "FollowUp" GROUP BY status;
```

Files Created/Modified

Created Files (11):

1. `src/lib/followup/followup-rules.ts`
2. `src/lib/followup/followup-scheduler.ts`
3. `src/lib/followup/followup-processor.ts`
4. `src/lib/sms/sms-service.ts`
5. `src/lib/hooks/inquiry-hooks.ts`
6. `src/app/api/follow-ups/process/route.ts`
7. `src/app/api/follow-ups/[id]/route.ts`
8. `docs/AUTOMATED_FOLLOWUP_SYSTEM.md`
9. `FEATURE_4_PHASE_3_IMPLEMENTATION.md` (this file)

Modified Files (3):

1. `src/app/api/inquiries/route.ts` - Added hook integration
2. `src/app/api/inquiries/[id]/route.ts` - Added hook integration
3. `.env.example` - Added Twilio and CRON_SECRET variables

Existing Files (used, not modified):

1. `src/lib/ai/inquiry-response-generator.ts` - Used for content generation
2. `src/lib/email/inquiry-email-service.ts` - Used for email delivery
3. `prisma/schema.prisma` - Uses existing FollowUp and InquiryResponse models

Architecture Diagram

```
┌──────────────────────────────────────────────────────────────┐
│                    Inquiry Creation/Update                   │
│                  (API: /api/inquiries)                       │
└──────────────────────────────────────────────────────────────┘
                         │
                         ▼
                ┌─────────────────┐
                │  Inquiry Hooks  │
                └─────────────────┘
                         │
                         ▼
            ┌─────────────────────────┐
            │  Follow-up Rules        │
            │  Engine                 │
            │  (Evaluate Rules)       │
            └─────────────────────────┘
                     │
                     ▼
        ┌─────────────────────────────┐
        │  Follow-up Scheduler        │
        │  (Create FollowUp records)  │
        └─────────────────────────────┘
                 │
                 ▼
        ┌─────────────────────┐
        │  Database           │
        │  (FollowUp table)   │
        └─────────────────────┘
                 │
                 ▼
     ┌────────────────────────────┐
     │  Cron Job (every 15 min)   │
     │  /api/follow-ups/process   │
     └────────────────────────────┘
                 │
                 ▼
     ┌────────────────────────────┐
     │  Follow-up Processor       │
     │  (Process due follow-ups)  │
     └────────────────────────────┘
         │
         ├─────────────┬──────────────┬─────────────┐
         ▼             ▼              ▼             ▼
    ┌───────┐    ┌─────────┐    ┌──────────┐    ┌──────┐
    │ Email │    │   SMS   │    │  Phone   │    │ Task │
    │Service│    │Service  │    │   Call   │    │      │
    └───────┘    └─────────┘    └──────────┘    └──────┘
         │            │               │             │
         └────────────┴───────────────┴─────────────┘
                           │
                           ▼
               ┌────────────────────────┐
               │  Update FollowUp       │
               │  Create Response       │
               │  (Track delivery)      │
               └────────────────────────┘
```

Key Benefits

1. **Automated Engagement**: No manual effort for routine follow-ups
2. **Consistent Communication**: Every inquiry gets appropriate follow-up
3. **Multi-Channel**: Reach families via their preferred method
4. **Intelligent Timing**: Rules-based scheduling optimizes timing
5. **Scalable**: Handles high volume with background processing
6. **Flexible**: Easy to add custom rules or modify existing ones
7. **Trackable**: Complete audit trail of all follow-ups

Performance Considerations

- **Non-blocking Hooks**: Follow-up scheduling doesn't delay API responses
- **Batch Processing**: Cron job processes multiple follow-ups efficiently
- **Rate Limiting**: SMS/Email services respect rate limits
- **Database Indexing**: FollowUp table has proper indexes
- **Deduplication**: Prevents scheduling duplicate follow-ups

Security

- **Authentication**: All endpoints require authentication (except cron)
- **Authorization**: Role-based access control for follow-up management
- **CRON_SECRET**: Protects cron endpoint from unauthorized access
- **Data Privacy**: Follow-up content respects privacy settings
- **Audit Trail**: All actions are logged

Next Steps

1. **Deploy to Production**:
   - Add environment variables to Render
   - Set up cron job
   - Test SMS delivery

2. **Monitor Performance**:
   - Track follow-up delivery rates
   - Measure response rates
   - Adjust rules based on data

3. **Future Enhancements**:
   - Email open tracking
   - Link click tracking
   - A/B testing for messages
   - ML-based timing optimization

Deployment Checklist

- [ ] Add Twilio credentials to production environment
- [ ] Add CRON_SECRET to production environment

- [ ] Set up cron job on Render (or alternative)
- [ ] Test cron endpoint manually
- [ ] Create test inquiry and verify follow-ups are scheduled
- [ ] Monitor logs for any errors
- [ ] Verify SMS delivery (if enabled)
- [ ] Verify email delivery
- [ ] Check database for follow-up records
- [ ] Monitor for first week and adjust rules if needed

Support

For questions or issues:
1. Review `docs/AUTOMATED_FOLLOWUP_SYSTEM.md`
2. Check application logs
3. Query database for follow-up status
4. Contact development team

Conclusion

Phase 3 successfully implements a comprehensive automated follow-up system that:
- Intelligently schedules follow-ups based on rules
- Delivers via multiple channels (email, SMS)
- Processes follow-ups in background
- Provides manual override capabilities
- Integrates seamlessly with existing inquiry system

The system is production-ready and can be deployed immediately after configuration.