AI Response Generation System

Overview

The AI Response Generation system uses OpenAI GPT-4 to create personalized, context-aware responses to family inquiries about senior care.

Features

- **Context-Aware**: Analyzes inquiry details, care needs, and urgency
- **Tone Matching**: Adjusts tone based on urgency level
- **Personalization**: Includes specific details about the inquiry
- **Home Matching**: Can include recommended homes
- **Next Steps**: Suggests clear actions for families
- **Email Integration**: Sends formatted emails automatically

API Endpoints

Generate AI Response

```
POST /api/inquiries/:id/generate-response
```

Request Body:

```
{
  "type": "INITIAL" | "FOLLOW_UP" | "TOUR_CONFIRMATION" | "GENERAL",
  "tone": "PROFESSIONAL" | "WARM" | "EMPATHETIC" | "URGENT",
  "includeNextSteps": true,
  "includeHomeDetails": true,
  "sendEmail": false
}
```

Response:

```
{
  "success": true,
  "response": {
    "id": "response-id",
    "content": "Generated response text...",
    "status": "DRAFT" | "SENT" | "DELIVERED" | "FAILED"
  }
}
```

Usage Examples

Generate Initial Response

```
const response = await fetch('/api/inquiries/inquiry-123/generate-response', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({
    type: 'INITIAL',
    includeNextSteps: true,
    sendEmail: false, // Preview first
  }),
});
```

Generate and Send Urgent Response

```
const response = await fetch('/api/inquiries/inquiry-456/generate-response', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({
    type: 'INITIAL',
    tone: 'URGENT',
    sendEmail: true, // Send immediately
  }),
});
```

Tone Guidelines

- **PROFESSIONAL**: Clear, informative, business-like
- **WARM**: Friendly, welcoming, supportive
- **EMPATHETIC**: Understanding, compassionate, acknowledging emotions
- **URGENT**: Quick, action-oriented, calm but responsive

Email Template

Emails are sent with:
- CareLinkAI branding
- Professional formatting
- Contact information
- Reference ID for tracking
- Mobile-responsive design

Environment Variables

Required:
- `OPENAI_API_KEY` : OpenAI API key
- `SMTP_HOST` : Email server host
- `SMTP_PORT` : Email server port
- `SMTP_USER` : Email username
- `SMTP_PASS` : Email password
- `SMTP_FROM` : From email address

Best Practices

1. **Preview First**: Generate without sending to review content
2. **Match Tone**: Use appropriate tone for urgency level
3. **Include Context**: Enable home details for better responses
4. **Track Responses**: Monitor delivery status
5. **Follow Up**: Schedule follow-ups for non-urgent inquiries

Error Handling

The system handles:
- Missing inquiry data
- API failures (falls back gracefully)
- Email delivery failures (marks as FAILED)
- Invalid parameters (validation errors)

Response Types

INITIAL

First response to a new inquiry. Includes:
- Thank you message
- Acknowledgment of situation
- How CareLinkAI can help
- Recommended homes (if applicable)
- Clear next steps
- Contact information

FOLLOW_UP

Follow-up for existing inquiries. Includes:
- Reference to previous inquiry
- Progress check
- Additional assistance offer
- Next steps reminder

TOUR_CONFIRMATION

Tour confirmation message. Includes:
- Tour detail confirmation
- What to expect
- Preparation tips
- Contact information for changes

GENERAL

General informational response. Includes:
- Address specific questions
- Helpful information
- Additional resources

Architecture

Components

1. **InquiryResponseGenerator** (`src/lib/ai/inquiry-response-generator.ts`)
   - Core AI response generation logic
   - OpenAI GPT-4 integration
   - Context building and prompt engineering
   - Tone matching based on urgency

2. **InquiryEmailService** (`src/lib/email/inquiry-email-service.ts`)
   - Email sending functionality
   - HTML email formatting
   - Professional templates
   - Error handling

3. **API Route** (`src/app/api/inquiries/[id]/generate-response/route.ts`)
   - HTTP endpoint for response generation
   - Authorization checks
   - Database persistence
   - Email sending coordination

4. **Response Templates** (`src/lib/ai/response-templates.ts`)
   - Predefined templates for common scenarios
   - Template selection logic
   - Subject line management

Database Schema

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])
  createdAt   DateTime        @default(now())
  updatedAt   DateTime        @updatedAt
}
```

Security Considerations

1. **Authorization**: Only ADMIN and OPERATOR roles can generate responses
2. **Data Privacy**: Inquiry data is handled according to HIPAA guidelines
3. **API Key Security**: OpenAI API key stored in environment variables

4. **Email Security**: SMTP credentials stored securely

Future Enhancements

- Multi-language support
- A/B testing for response effectiveness
- Response analytics and optimization
- SMS integration
- Voice call scripts
- Automated follow-up scheduling
- Sentiment analysis
- Response quality scoring
- Template learning from successful responses

Testing

Manual Testing

1. Create a test inquiry via the admin panel
2. Navigate to the inquiry detail page
3. Click “Generate AI Response”
4. Review the generated content
5. Test with different tones and types
6. Test email sending functionality

API Testing

```
# Test response generation
curl -X POST http://localhost:3000/api/inquiries/inquiry-id/generate-response \
  -H "Content-Type: application/json" \
  -d '{
    "type": "INITIAL",
    "tone": "WARM",
    "includeNextSteps": true,
    "includeHomeDetails": true,
    "sendEmail": false
  }'
```

Troubleshooting

Common Issues

Issue: “OpenAI API key not configured”
- **Solution**: Add `OPENAI_API_KEY` to `.env` file

Issue: “Email sending failed”
- **Solution**: Check SMTP configuration in `.env`

Issue: “Inquiry not found”
- **Solution**: Verify inquiry ID is correct

Issue: “Unauthorized”
- **Solution**: Ensure user has ADMIN or OPERATOR role

Debug Mode

Enable debug logging by setting:

```
DEBUG=carelinkai:ai:*
```

Support

For issues or questions:
- GitHub Issues: profyt7/carelinkai (https://github.com/profyt7/carelinkai/issues)
- Documentation: `/docs/AI_RESPONSE_GENERATION.md`
- Email: support@carelinkai.com

Last Updated: December 18, 2025
Version: 1.0.0
Feature: #4 - AI-Powered Inquiry Response & Follow-up System