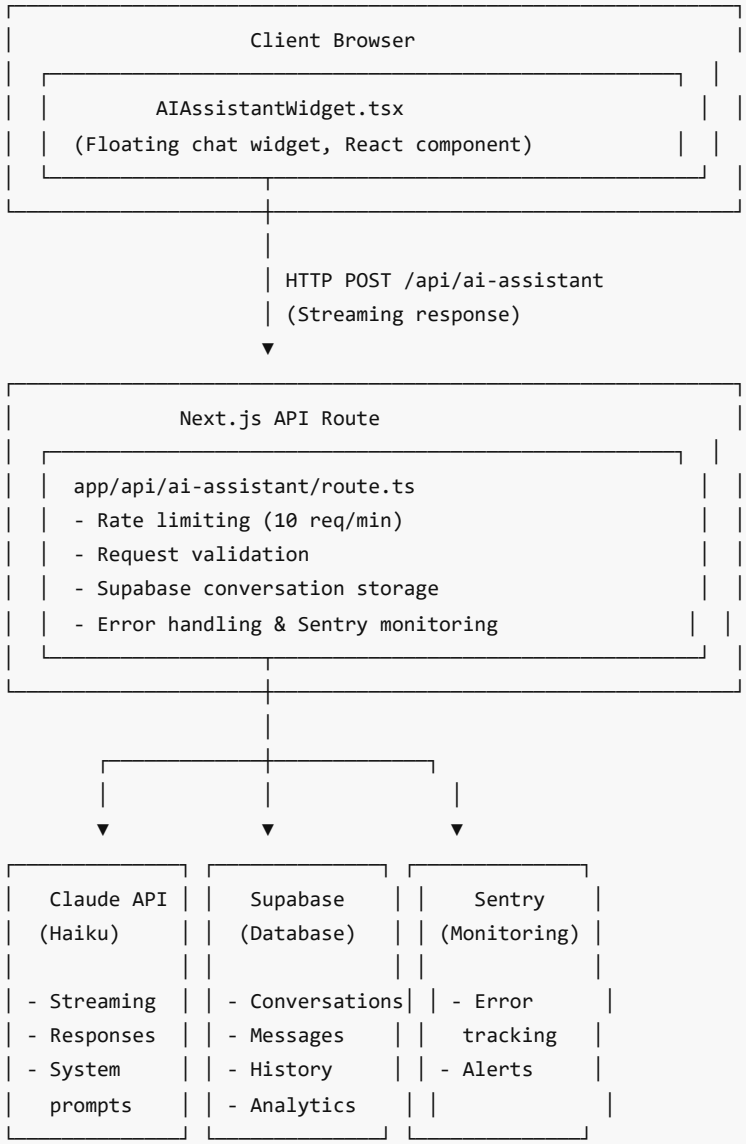
AI Assistant Architecture Overview

Date: January 26, 2025
Purpose: System architecture documentation for AI Assistant feature
Status: Production Ready

🏗️ System Architecture

High-Level Overview



📦 Component Structure

1. Frontend Component

File: `components/AIAssistantWidget.tsx`

Responsibilities:

- Render floating chat widget UI
- Handle user input and message display

- Stream responses from API
- Manage widget state (open/closed/minimized)
- Keyboard navigation (WCAG 2.1 AA)
- Markdown link parsing

Key Features:

- Client-side component (`'use client'`)
- Real-time streaming responses
- Conversation history management
- Error handling and display
- Accessibility compliance

---

2. API Route

File: `app/api/ai-assistant/route.ts`

Responsibilities:

- Handle POST requests from widget
- Rate limiting (10 requests/minute per IP)
- Validate incoming messages
- Stream responses from Claude API
- Store conversations in Supabase
- Error monitoring with Sentry

Key Features:

- Streaming responses (Server-Sent Events)
- Rate limiting with in-memory store
- Conversation persistence
- Comprehensive error handling
- TypeScript type safety

---

3. Claude Integration

File: `lib/claude.ts`

Responsibilities:

- Initialize Anthropic SDK client
- Format system prompt from knowledge base
- Stream chat responses
- Handle API errors gracefully
- Validate messages before sending

Key Features:

- Model: `claude-3-haiku-20240307`
- Streaming support
- Error categorization (402, 403, 429, etc.)
- Message validation

---

4. Knowledge Base

File: `lib/knowledge-base.ts`

Responsibilities:

- Define RWS services and pricing

- Provide FAQ responses
- Format system prompt for Claude
- Include guardrails and guidelines
- Provide website links and CTAs

Key Features:

- Centralized pricing (from `lib/config/pricing.ts`)
- Service descriptions
- FAQ database
- Guardrails for off-topic questions
- CTA strategy (questionnaire, booking)

---

5. Rate Limiting

File: `lib/rate-limit.ts`

Responsibilities:

- Track request counts per IP
- Enforce 10 requests/minute limit
- Return rate limit errors

Key Features:

- In-memory store (Map-based)
- IP-based tracking
- Configurable limits

---

6. Database Schema

File: `supabase/migrations/20250125_create_ai_assistant_tables.sql`

Tables:

- `ai_assistant_conversations` - Conversation metadata
- `ai_assistant_messages` - Individual messages

Features:

- UUID primary keys
- Row Level Security (RLS)
- Timestamps (TIMESTAMPTZ)
- Indexes for performance

🔄 Data Flow

1. User Sends Message

```
User types message
    ↓
AIAssistantWidget.tsx
    ↓
POST /api/ai-assistant
    ↓
Rate limit check
    ↓
Validate message
```

```
    ↓
Store user message in Supabase
    ↓
Call Claude API (streaming)
    ↓
Stream chunks to client
    ↓
Store AI response in Supabase
    ↓
Display in widget
```

2. Conversation History

```
Widget loads
    ↓
Check for existing conversationId
    ↓
If exists: Fetch messages from Supabase
    ↓
Display in widget
    ↓
Continue conversation
```

🔐 Security & Performance

Security

1. **Rate Limiting:** 10 requests/minute per IP
2. **Input Validation:** Max 5000 characters, non-empty
3. **RLS Policies:** Supabase Row Level Security enabled
4. **API Key Protection:** Server-side only, never exposed
5. **Error Handling:** No sensitive data in error messages

Performance

1. **Streaming:** Real-time responses (no waiting for full response)
2. **Lazy Loading:** Widget loads on all pages (consider lazy loading if needed)
3. **Database Indexes:** Optimized queries for conversation history
4. **Error Monitoring:** Sentry for production error tracking

---

🛠️ Technology Stack

Frontend

- **Next.js 16** (App Router)
- **React 18** (Client components)
- **TypeScript** (Type safety)
- **TailwindCSS** (Styling)

Backend

- **Next.js API Routes** (Serverless functions)
- **Claude 3 Haiku API** (AI responses)
- **Supabase** (Database & storage)
- **Sentry** (Error monitoring)

Infrastructure

- **Vercel** (Hosting & deployment)
- **Supabase Cloud** (Database)
- **Anthropic Cloud** (AI API)

📊 Database Schema

`ai_assistant_conversations`

```
- id (UUID, PRIMARY KEY)
- created_at (TIMESTAMPTZ)
- updated_at (TIMESTAMPTZ)
- user_ip (TEXT)
- metadata (JSONB)
```

`ai_assistant_messages`

```
- id (UUID, PRIMARY KEY)
- conversation_id (UUID, FOREIGN KEY)
- role (TEXT) -- 'user' or 'assistant'
- content (TEXT)
- created_at (TIMESTAMPTZ)
```

🔧 Configuration

Environment Variables

```
# Required
ANTHROPIC_API_KEY=sk-ant-...
NEXT_PUBLIC_SUPABASE_URL=https://...
NEXT_PUBLIC_SUPABASE_ANON_KEY=eyJ...

# Optional
SENTRY_DSN=https://... (for error monitoring)
```

Rate Limiting

Current: 10 requests/minute per IP
Location: `lib/rate-limit.ts`
Storage: In-memory (resets on server restart)

🚀 Deployment

Vercel Deployment

1. Push to GitHub
2. Vercel auto-deploys
3. Environment variables set in Vercel dashboard
4. Verify widget appears on live site

Database Migration

1. Run migration: `supabase/migrations/20250125_create_ai_assistant_tables.sql`

2. Verify tables created
3. Check RLS policies enabled

📈 Monitoring & Analytics

Sentry

- Error tracking
- Performance monitoring
- User session tracking

Supabase

- Conversation analytics
- Message counts
- User engagement metrics

Claude API

- Token usage
- Response times
- Cost tracking

🔄 Future Enhancements

Potential Improvements

1. **Lazy Loading:** Load widget only when user interacts
2. **React.memo():** Optimize widget re-renders
3. **Caching:** Cache common responses
4. **Analytics:** Track conversation quality
5. **A/B Testing:** Test different system prompts
6. **Model Upgrade:** Consider Claude 3.5 Sonnet for complex queries

---

📚 Related Documentation

- `docs/AI_ASSISTANT_WIDGET_DEPLOYMENT.md` - Deployment guide
- `docs/AI_ASSISTANT_TESTING_GUIDE.md` - Testing procedures
- `docs/MODEL_SELECTION_GUIDE.md` - Haiku vs Sonnet
- `docs/AI_ASSISTANT_GUARDRAILS.md` - Guardrail implementation
- `docs/AI_ASSISTANT_LINKS_GUIDE.md` - CTA strategy

✅ Architecture Checklist

- ☑ Streaming responses implemented
- ☑ Rate limiting configured
- ☑ Database schema created
- ☑ Error handling comprehensive
- ☑ Accessibility compliant (WCAG 2.1 AA)
- ☑ TypeScript types defined
- ☑ Monitoring configured (Sentry)
- ☑ Documentation complete

Last Updated: January 26, 2025
Status: Production Ready

Next Review: When upgrading model or adding features