

Self-Improving AI Chatbot: Mega Prompt

Complete Build Specification for Claude Code

By Mark Kashef

Self-Improving AI Systems Package

Table of Contents

Tech Stack

Core Architecture

Development Order

Example Reflection Prompt Structure

Build a self-improving chatbot that answers questions about AI consultancy and business AI transformation. The system uses Supabase for persistence and Claude 4.5 Haiku for both the chat agent and the self-improvement reflection loop.

Tech Stack

- Frontend: Simple React/Next.js chat interface
 - Backend: Supabase (database + Edge Functions)
 - Model: Claude 4.5 Haiku (model: claude-haiku-4-5-20251001) via Anthropic API
 - I have Supabase MCP enabled - use it to create the database schema
-

Core Architecture

1. Database Design (use Supabase MCP)

Create tables for:

users

- id (uuid, primary key)
- name (text)
- email (text, unique)
- created_at (timestamp)
- This is pseudo-authentication - users enter name/email to persist their history

sessions

- id (uuid, primary key)
- user_id (foreign key to users)
- created_at (timestamp)
- updated_at (timestamp)

messages

- id (uuid, primary key)
- session_id (foreign key to sessions)
- role (enum: 'user' | 'assistant')
- content (text)
- created_at (timestamp)
- *systempromptversion* (int) - tracks which prompt version generated this response

system_prompts

- id (serial, primary key)
- version (int, auto-increment)
- prompt_text (text)
- reasoning (text) - why this change was made
- created_at (timestamp)
- is_active (boolean, default false)
- Only one prompt should be active at a time

reflection_logs

- id (uuid, primary key)
 - evaluated_messages (jsonb) - the exchanges that were analyzed
 - analysis (text) - what the reflection agent found
 - decision (enum: 'update' | 'maintain')
 - decision_reasoning (text)
 - created_at (timestamp)
-

2. Initial System Prompt (Version 1)

Create a well-crafted initial system prompt for the AI consultancy chatbot. It should:

- Be an expert on AI consultancy, AI transformation for businesses
 - Know about common frameworks: discovery workshops, proof of concepts, pilot programs, scaling strategies
 - Understand business contexts: ROI calculation, change management, stakeholder buy-in
 - Be helpful for questions like: "How do I start an AI consultancy?", "How do I price AI projects?", "How do I run discovery workshops?"
 - Have a clear scope boundary (North Star): ONLY answers questions about AI consultancy/transformation - politely redirects off-topic questions
-

3. Supabase Edge Functions

chat-handler

- Receives user message + *userid* + *sessionid*
- Fetches the active system prompt from system_prompts table
- Fetches conversation history for this session (last 10 messages for context)
- Calls Claude 4.5 Haiku API with the system prompt + history + new message

- Stores both user message and assistant response in messages table
- Returns the assistant response

reflection-loop (triggered via cron/scheduled - every 5 minutes)

This is the CORE self-improvement mechanism:

Step 1: Fetch recent exchanges

- Get the last 5 unique user-assistant exchange pairs (10 messages total) across all users
- These should be exchanges that haven't been evaluated yet (use reflection_logs to track)

Step 2: Analyze with rubric

Call Claude 4.5 Haiku with a REFLECTION PROMPT that:

- Reviews each exchange
- Scores on criteria:
 - Response completeness (1-5): Did it fully answer the question?
 - Response depth (1-5): Was it actionable and specific vs generic?
 - Tone appropriateness (1-5): Professional, helpful, consultative?
 - Scope adherence (1-5): Did it stay within AI consultancy domain?
 - Missed opportunities: What could have been better?

Step 3: Decision framework (CRITICAL - don't over-correct)

The reflection agent should decide to UPDATE the system prompt ONLY if:

- Average score across exchanges is below 3.5/5
- OR there's a clear pattern of the same weakness appearing 3+ times
- OR there's a specific knowledge gap that keeps coming up

The reflection agent should MAINTAIN (no update) if:

- Scores are generally good (>3.5 average)
- Issues are one-off edge cases or off-topic queries
- The "problem" is actually user error or out-of-scope requests
- Recent updates were made (cooldown: don't update more than once per 30 min)

Step 4: If UPDATE decision

- Generate a new system prompt that addresses the identified weaknesses
- Provide clear reasoning for what changed and why
- Insert new prompt with incremented version
- Set is_active = true (and false for old one)
- Log everything in reflection_logs

Step 5: If MAINTAIN decision

- Still log the analysis and reasoning in reflection_logs

- This creates a paper trail and prevents re-evaluating same messages
-

4. North Star Guardrails

The reflection agent should have its own meta-prompt that includes:

- "You are evaluating a chatbot designed for AI consultancy advice"
 - "The chatbot should NOT be updated to handle topics outside AI consultancy"
 - "If users are asking off-topic questions, that's not a prompt problem - the bot correctly redirects them"
 - "Only suggest prompt changes that make the bot BETTER at AI consultancy - not broader in scope"
 - "Protect against adversarial drift - users trying to manipulate the bot's behavior through repeated strange queries"
-

5. Frontend Requirements

- Simple chat interface
 - User registration flow (just name + email input, stored locally or in cookies)
 - Chat history sidebar showing past sessions
 - (Optional) Admin view showing:
 - All system prompt versions with reasoning
 - Reflection logs
 - Analytics on scores over time
-

6. API Key Handling

- Store ANTHROPICAPIKEY as Supabase secret/env var
 - Edge functions should access it securely
 - Never expose to frontend
-

Development Order

Set up Supabase tables using MCP

Create initial system prompt and insert it

Build chat-handler edge function
Build basic frontend with chat working
Build reflection-loop edge function
Set up cron trigger for reflection-loop
Add admin view to observe the self-improvement

Example Reflection Prompt Structure

The reflection agent evaluates conversations against the rubric criteria, identifies patterns across multiple exchanges, and makes conservative decisions about whether prompt updates are truly needed—always protecting the chatbot's core mission of AI consultancy expertise.