AI Automation / AI Agents

Design an AI agent that handles common support tickets, escalates when needed, and learns from resolutions.
Difficulty: Advanced
Model: GPT-4 / Claude / Gemini
Use Case: Support Automation, Ticket Triage, Customer Service
Updated: May 2026
Why This Prompt Exists
Customer support scales linearly with headcount — until you add AI agents. But poorly designed agents frustrate customers and create more work.

You get:

- agents that give wrong answers (customers escalate angrily)
- agents that can’t handle common questions (defeats the purpose)
- agents that escalate too easily (no cost savings)
- agents that never escalate when needed (customer churn)
- no learning from past conversations (same mistakes repeated)

But good support agents have structure:

- intent classification: what is the customer asking about?
- knowledge base: where does the agent find answers?
- response generation: how does the agent phrase answers?
- escalation rules: when should a human take over?
- feedback loop: how does the agent learn from corrections?

Without design, support agents create more problems than they solve.

This prompt designs effective customer support AI agents.

The Prompt

```
Assume the role of a customer support automation architect who designs AI agents.

Your task is to design an AI agent that handles support tickets.

Generate:

1. SUPPORT SCOPE
   - Products/services covered
   - Ticket types the agent will handle (e.g., password reset, billing question, feature request)
   - Ticket types that always escalate (e.g., security issues, account deletion)

2. INTENT CLASSIFICATION
   - List of intents the agent should recognize
   - Sample phrases for each intent
   - Confidence threshold for escalation (e.g., "escalate if confidence < 0.7")

3. KNOWLEDGE BASE DESIGN
   - Source documents (help center articles, internal wikis, past tickets)
   - Update frequency (real-time / daily / weekly)
   - Search strategy (semantic search + keyword fallback)
```

4. RESPONSE PROTOCOL
   - Tone (professional / friendly / concise)
   - Structure (acknowledgment + answer + next steps)
   - Variables to include (customer name, order number, etc.)

5. ESCALATION RULES
   - When to escalate: [conditions, e.g., "customer says 'speak to a human'"]
   - Escalation target: (specific team, priority queue)
   - Information to pass (conversation history, intent, confidence)

6. FEEDBACK & LEARNING LOOP
   - How to capture human corrections
   - How to update knowledge base from resolutions
   - Review frequency for agent performance

7. READY-TO-USE AGENT PROMPT
   - The system prompt for the support agent

INPUTS:

Products/services to support:
[E.G., "SaaS project management software"]

Common support tickets (from history):
[PASTE 10-20 EXAMPLE TICKETS]

Existing knowledge base (if any):
[E.G., "Help center with 50 articles"]

```
Human support team size:
[E.G., "5 agents, handling 500 tickets/day"]

RULES:
- Start with narrow scope (one product, one ticket type) and expand
- Train intent classifier on real tickets (not synthetic data)
- Set escalation threshold conservatively (better to over-escalate early)
- Log all agent interactions for quality review (sample 5-10% daily)
- Monitor customer satisfaction (CSAT) for agent-handled tickets separately
- Have a human "training wheels" period before full automation
```

How To Use It

- Start with a narrow scope — one product, one ticket type — and expand gradually.
- Train intent classification on real support tickets, not synthetic examples.
- Set escalation thresholds conservatively at first (better to over-escalate).
- Log every agent interaction for quality review and improvement.
- Monitor CSAT for agent-handled tickets separately from human-handled tickets.

Example Input

Products/services to support:
"Project management SaaS — tasks, projects, team collaboration"

Common support tickets:
"Password reset, invitation not received, how to create a task, billing question"

Existing knowledge base:
"Help center with 50 articles"

Human support team size:
"5 agents, 500 tickets/day"

Why It Works
Most customer support agents are built ad hoc — a system prompt and a knowledge base — which leads to inconsistent, often wrong answers.

This framework improves outcomes by forcing:

- scope definition (what will this agent handle?)
- intent classification (what is the customer asking?)
- knowledge base design (where do answers come from?)
- escalation rules (when to involve a human?)
- feedback loops (how does it improve over time?)

Great support agent design doesn't replace humans — it handles the 80% of routine tickets so humans can focus on complex issues.

Build Better AI Systems

Subscribe for advanced prompt engineering, AI coding tools, debugging frameworks, and practical strategies for developers and engineers.

Carefully engineered prompts for people doing real work.

Share this:

- Share on Facebook (Opens in new window) Facebook
- Share on X (Opens in new window) X

See also Research Agent Workflow