AI Agent Workflow

Overview

Your workflow builds a simple AI coder agent inside n8n that can be triggered by a chat trigger node. The Agent node ties together a language model (Google Gemini / PaLM), a memory node (Simple Memory), and chat triggers so the agent can receive messages and respond. In short:

- **When chat message received** are entry points they start the flow when a message arrives.
- **AI Agent** is the central node that orchestrates reasoning, tools, and the response.
- **Google Gemini Chat Model** is the language model the agent uses to generate replies.
- **Simple Memory** holds recent conversation context (a buffer) so the agent remembers what was said recently.

Node-by-node explanation

---

1) When chat message received (type: `chatTrigger`)

- What it is: A specialized trigger for chat-centric flows (often used with live chat UIs or integrated chat services). It raises events when users send messages.
- How it connects: In your workflow it sends data directly to the **AI Agent** node, parallel to the webhook entry.
- Human analogy: A live chat window on a website — every time the user sends a message the chat trigger rings the bell.

---

2) AI Agent (type: `@n8n/n8n-nodes-langchain.agent`)

- What it is: The brain node. It coordinates inputs (user message), memory, the language model, tools, and produces an output message.
- How your agent is wired: It receives inputs from both the Webhook and the Chat Trigger (these are the ways the agent starts). The agent also has two special connections wired:
- `ai_languageModel` → **Google Gemini Chat Model**. This tells the agent which LLM to call when it needs to generate text.
- `ai_memory` → **Simple Memory**. This tells the agent where to read and write short-term context.
- Human analogy: The agent is like a coworker who takes the incoming message, remembers what was said earlier, asks the language model for wording, and returns a natural reply.
- Important bits to check:
- Make sure the agent node's fields (prompt templates, tool settings, or instructions) are filled — an empty agent might not know how to behave.

- If you add tools (external actions like database lookups or web requests), they should be configured and connected inside the agent settings.

---

3) Google Gemini Chat Model (type: `@n8n/n8n-nodes-langchain.lmChatGoogleGemini`)

- What it is: The language model implementation for Google Gemini (PaLM). This node wraps the model's API so n8n can ask it to generate replies.
- Credentials: You attached `googlePalmApi` credentials. Ensure these credentials are valid and have quota.
- What it does: Receives a prompt from the agent, returns generated text, and the agent uses that text as its reply.
- Human tip: If you see authentication or quota errors here, the agent won't produce replies. Check the credentials (API key, project, billing) in n8n's credential manager.

---

4) Simple Memory (type: `memoryBufferWindow`)

- What it is: A memory buffer that keeps the recent conversation (last N messages) so the agent can refer back to earlier turns.
- Why it matters: Without memory, the agent would treat each incoming message as an isolated request. With memory, it can keep context like the user's name, earlier questions, or variables.
- Config details: The node usually needs configuration for the size of the buffer (how many previous messages to keep) and possibly a `memoryKey` or name under which it stores the memory.
- Common problem (the "Simple Memory" error you saw): If the memory node is not configured correctly, or the agent expects memory but the node is missing required fields, n8n will throw an error when the workflow runs. We'll go deeper into troubleshooting below.

Example: Testing the workflow manually

1. Start n8n and make sure the workflow is active (toggle `active: true`).
2. Send a test HTTP POST to your webhook endpoint (use `curl` or Postman). Example `curl`:

```
curl -X POST 'https://your-n8n-host/webhook/wtZEpLjacJ7rmOEt/f69aeeed-8899-4297-a746-c96c548537da'
  -H 'Content-Type: application/json'
  -d '{"message": "Hi! Who are you?" }'
```

1. Watch the n8n executions panel. You'll see the webhook execution, the agent execution, and any calls the agent made to the Gemini model.
2. Inspect the output JSON of the AI Agent node to read the reply.
