a **chat-based AI agent in n8n using the LLaMA 3 model via Ollama**, instead of OpenAI.
That's entirely possible!

Let's walk through how to set this up.

What You'll Build

An **AI Q&A agent** that:

1. Accepts a user's question from a webhook or chat UI
2. Sends it to **LLaMA 3 via Ollama API** running locally
3. Returns the answer in real-time

Requirements

- **n8n** installed (self-hosted or cloud)
- **Ollama** installed on your local/server environment
- LLaMA 3 model already pulled via:
- `ollama pull llama3`
- Your **n8n** instance must be able to reach `http://localhost:11434`

 Note: If you're running n8n in Docker, make sure it can access the Ollama container or expose Ollama on a network interface n8n can reach.

Step-by-Step Guide: n8n + Ollama + LLaMA 3 Agent

---

1. Set up a Webhook Trigger (Chat Input)

Node: `Webhook`

- HTTP Method: `POST`
- Path: `llama3-chat`
- Respond immediately: **No** (we'll use a Webhook Response node)

This will accept chat messages like:

```
{
```

```
  "message": "What is the future of AI in education?"
}
```

□ 2. Send Message to LLaMA 3 via Ollama

Node: `HTTP Request`
Configure it like this:

- **HTTP Method**: `POST`
- **URL**: `http://localhost:11434/api/chat`
- **Headers**:
- {
-   "Content-Type": "application/json"
- }
- **Body Content Type**: `JSON`
- **Body Parameters**:
- {
-   "model": "llama3",
-   "messages": [
-     {
-       "role": "user",
-       "content": "={{$json["body"]["message"]}}"
-     }
-   ]
- }

□ 3. Return the Response

Node: `Webhook Response`

- Response code: `200`
- Response body:
- {
-   "answer": "={{$json["message"]["content"]}}"
- }

□ Ollama’s response format:

```
{
  "model": "llama3",
  "created_at": "...",
  "message": {
    "role": "assistant",
    "content": "Here is the AI's answer..."
  },
  ...
}
```

□ Full n8n Workflow (LLaMA 3 via Ollama)

Here's the complete JSON to import into your n8n instance chat.json :

```
{
  "nodes": [
    {
      "parameters": {
        "httpMethod": "POST",
        "path": "llama3-chat"
      },
      "id": "WebhookTrigger",
      "name": "Webhook",
      "type": "n8n-nodes-base.webhook",
      "typeVersion": 1,
      "position": [200, 300]
    },
    {
      "parameters": {
        "authentication": "none",
        "url": "http://localhost:11434/api/chat",
        "method": "POST",
        "responseFormat": "json",
        "jsonParameters": true,
        "options": {},
        "bodyParametersJson": "={\n  \"model\": \"llama3\",\n  \"messages\":
[\n    {\n      \"role\": \"user\",\n      \"content\":
\"{{$json[\"body\"][\"message\"]}}\"\n    }\n  ]\n}"
      },
      "id": "HTTP_Request",
      "name": "Ollama - LLaMA 3",
      "type": "n8n-nodes-base.httpRequest",
      "typeVersion": 1,
      "position": [450, 300]
    },
    {
      "parameters": {
        "responseMode": "lastNode",
        "options": {}
      },
      "id": "WebhookResponse",
      "name": "Webhook Response",
      "type": "n8n-nodes-base.webhookResponse",
      "typeVersion": 1,
      "position": [700, 300]
    }
  ],
  "connections": {
    "Webhook": {
      "main": [
        [
          {
            "node": "Ollama - LLaMA 3",
```

```
            "type": "main",
            "index": 0
          }
        ]
      ]
    },
    "Ollama - LLaMA 3": {
      "main": [
        [
          {
            "node": "Webhook Response",
            "type": "main",
            "index": 0
          }
        ]
      ]
    }
  }
}
```

□ Testing It

You can test this by sending a `POST` request to:

```
http://localhost:5678/webhook/llama3-chat
```

With this JSON body:

```
{
  "message": "Tell me about the latest AI trends in 2025."
}
```

Expected response:

```
{
  "answer": "The latest AI trends in 2025 include..."
}
```

□ Want Chat Memory?

To extend it into a "conversational" agent with memory:

- Store `chat history` in a variable or external DB
- Append the last N messages to the `messages` array sent to Ollama
- Optional: Set role as `"system"` for persistent instructions (e.g., assistant personality)

□□Web Chat UI:-

Pair this with a simple HTML/JavaScript frontend (like the one I showed earlier) to simulate a chat window calling the webhook.

Perfect — let's create a **simple web-based chat UI** that connects to your **n8n webhook AI agent (LLaMA 3 via Ollama)**.

□ Goal:

Create a **chat interface** that sends user questions to your n8n webhook and displays LLaMA 3's responses like a chatbot.

□ Assumptions

You already have:

- The n8n webhook running at something like:
- `http://localhost:5678/webhook/llama3-chat`

If you're running it on a server, change `localhost` to your actual domain or public IP.

□ HTML + JavaScript Chat UI

Here's a **clean, responsive** UI to plug into your n8n webhook:

□ **`chat.html`**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>LLaMA 3 AI Chat</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background: #f3f3f3;
      margin: 0;
```

```
      padding: 0;
    }

    #chat-container {
      width: 100%;
      max-width: 600px;
      margin: 40px auto;
      background: #fff;
      border-radius: 10px;
      padding: 20px;
      box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
    }

    .message {
      padding: 10px;
      margin: 10px 0;
      border-radius: 5px;
    }

    .user {
      background-color: #e0f7fa;
      text-align: right;
    }

    .bot {
      background-color: #f1f8e9;
      text-align: left;
    }

    #input-container {
      display: flex;
      margin-top: 20px;
    }

    #userInput {
      flex: 1;
      padding: 10px;
      font-size: 16px;
    }

    #sendBtn {
      padding: 10px 20px;
      font-size: 16px;
      background-color: #2196f3;
      color: white;
      border: none;
      cursor: pointer;
    }

    #sendBtn:hover {
      background-color: #1976d2;
    }
  </style>
</head>
<body>

  <div id="chat-container">
```

```
    <div id="chat"></div>
    <div id="input-container">
      <input type="text" id="userInput" placeholder="Ask me anything..." />
      <button id="sendBtn" onclick="sendMessage()">Send</button>
    </div>
  </div>

  <script>
    const webhookURL = "http://localhost:5678/webhook/llama3-chat"; // CHANGE
THIS IF NEEDED

    function addMessage(content, type) {
      const msgDiv = document.createElement("div");
      msgDiv.className = `message ${type}`;
      msgDiv.textContent = content;
      document.getElementById("chat").appendChild(msgDiv);
      msgDiv.scrollIntoView({ behavior: "smooth" });
    }

    async function sendMessage() {
      const inputEl = document.getElementById("userInput");
      const message = inputEl.value.trim();
      if (!message) return;

      addMessage(message, "user");
      inputEl.value = "";

      try {
        const response = await fetch(webhookURL, {
          method: "POST",
          headers: { "Content-Type": "application/json" },
          body: JSON.stringify({ message })
        });

        const data = await response.json();
        if (data.answer) {
          addMessage(data.answer, "bot");
        } else {
          addMessage("Sorry, no response from AI.", "bot");
        }
      } catch (err) {
        addMessage("Error talking to AI: " + err.message, "bot");
      }
    }

    document.getElementById("userInput").addEventListener("keypress",
function (e) {
      if (e.key === "Enter") {
        sendMessage();
      }
    });
  </script>

</body>
</html>
```

□ How to Use

1. Save the code as `chat.html`
2. Open it in a browser
3. Start chatting — it will hit your n8n webhook with each message

---

□ Making It Public (Optional)

If you want to deploy this publicly:

- Host `chat.html` on any static site host (Netlify, Vercel, GitHub Pages, etc.)
- Make sure n8n and Ollama are accessible via a public URL (via reverse proxy or tunnel)
- Replace `http://localhost:5678` with your public URL

Example:

```
const webhookURL = "https://yourdomain.com/webhook/llama3-chat";
```