Building a Custom LLM-Powered AI Chatbot with Ollama and Netlify Functions

Ireri Linus Mugendi
Founder, Lino.AI Co. Ltd.

July 7, 2025

Overview

This guide walks you through building a fully custom, AI-powered chatbot using a locally hosted large language model (LLM). You will learn how to:

- Host your own LLM (e.g., Llama 3) on Ubuntu using Ollama
- (Optionally) Fine-tune or customize your model
- Wrap it in a secure Node.js API server
- Expose it to the internet using a tunnel (Localtunnel or Cloudflare Tunnel)
- Connect it to a Netlify-hosted frontend via serverless functions
- Optimize for speed, reliability, and privacy

By the end, you'll have a fully functional, private AI chatbot pipeline — ideal for developers, startups, and researchers building ethical and scalable AI tools.

1 Step 1: Install Ollama and Pull or Fine-Tune Your Model

Install Ollama

```
curl -fsSL https://ollama.com/install.sh | sh
```

Pull a Pretrained Model

```
ollama pull llama3
```

(Optional) Fine-Tune Your Own Model

See: `https://ollama.com/library`

2 Step 2: Create the Node.js API Server

server.js

```
require('dotenv').config();
const express = require('express');
const cors = require('cors');
const axios = require('axios');

const app = express();
app.use(cors());
app.use(express.json());

const API_KEY = process.env.API_KEY;

app.post('/api/chat', async (req, res) => {
  if (req.headers['x-api-key'] !== API_KEY) {
    return res.status(401).json({ error: 'Unauthorized' });
  }

  try {
    const response = await axios.post('http://localhost:11434/api/generate
       ↪ ', {
      model: 'llama3',
      prompt: req.body.message,
      stream: false
    }, { timeout: 20000 });

    res.json({ response: response.data.response });
  } catch (err) {
    res.status(500).json({ error: 'Model timeout or error', details: err.
       ↪ message });
  }
});

app.listen(3001, () => {
  console.log('Lino.AI API Server running on port 3001');
});
```

3 Step 3: Expose Your API Server with a Tunnel

Using Localtunnel

```
npx localtunnel --port 3001
```

Persistent Alternative: Cloudflare Tunnel

https://developers.cloudflare.com/cloudflare-one/connections/connect-apps/install-and-setup/tunnel-guide/

4 Step 4: Connect Netlify Functions to Your API

netlify/functions/chat.js

```
const axios = require('axios');

exports.handler = async (event) => {
  try {
    const { message } = JSON.parse(event.body);
    const response = await axios.post(process.env.VPS_API_URL + '/api/chat
       ↪ ', {
      message
    }, {
      headers: { 'x-api-key': process.env.VPS_API_KEY },
      timeout: 9500
    });

    return {
      statusCode: 200,
      body: JSON.stringify(response.data)
    };
  } catch (err) {
    return {
      statusCode: 500,
      body: JSON.stringify({ error: 'Timeout or server error', details: err
         ↪ .message })
    };
  }
};
```

5 Step 5: Set Environment Variables in Netlify

- VPS_API_URL: e.g., https://icy-stars-post.loca.lt
- VPS_API_KEY: Must match the key in server.js

6 Step 6: Test Everything!

Test the API Directly

```
curl -X POST https://your-tunnel-url/api/chat \
  -H "x-api-key: your_key" \
  -H "Content-Type: application/json" \
  -d '{"message": "Hello"}'
```

7 Step 7: (Optional) Run as a Systemd Service

Create a service file

```
sudo nano /etc/systemd/system/lino-api.service
```

Example contents

```
[Unit]
Description=Lino.AI API Server
After=network.target
```

```
[Service]
ExecStart=/usr/bin/node /home/youruser/vps-api-server/server.js
Restart=always
User=youruser
Environment=NODE_ENV=production
EnvironmentFile=/home/youruser/vps-api-server/.env

[Install]
WantedBy=multi-user.target
```

Enable and Start

```
sudo systemctl daemon-reexec
sudo systemctl enable lino-api
sudo systemctl start lino-api
```

Tips and Notes

- Use **Cloudflare Tunnel** for persistent URLs.
- Keep your model warm using cron jobs or uptime pings.
- Use smaller models or quantized versions for faster responses.
- Monitor logs for timeouts and errors.
- Update your tunnel URL in Netlify whenever it changes.

Troubleshooting

- **Timeouts**: Use a smaller model or increase backend timeout.
- **503 Tunnel Unavailable**: Restart your tunnel and update Netlify.
- **401 Unauthorized**: Check API key in both server and Netlify.
- **CORS Errors**: Add your Netlify domain to CORS whitelist.

Conclusion

You now have a fully custom, locally hosted, LLM-powered AI chatbot pipeline — private, fast, and under your control. This setup is ideal for developers and startups looking to deploy ethical, scalable AI solutions without relying on third-party APIs.

This guide was authored by Ireri Linus Mugendi, founder of Lino.AI Co. Ltd., a company dedicated to building ethical and practical AI solutions for Africa and beyond.