Open WebUI Setup Guide: ChatGPT UI for Local AI

February 2, 2026 · by Mark Bartlett

Download this guide as PDF

> **Quick Answer:** Open WebUI gives you a ChatGPT-like web interface for any local model. One Docker command gets you running: `docker run -d -p 3000:8080 --add-host=host.docker.internal:host-gateway -v open-webui:/app/backend/data –name open-webui ghcr.io/open-webui/open-webui:main`. Point your browser at localhost:3000, connect it to Ollama, and you have a polished chat interface with RAG, voice chat, web search, and multi-model support — all running locally.

📚 **More on this topic:** Ollama Troubleshooting · Local RAG Guide · Voice Chat with Local LLMs · Best Models for Chat · Planning Tool

Typing prompts into a terminal works, but it gets old fast. If you want a real chat interface — conversations, file uploads, voice input, multiple models — Open WebUI is what most people end up using.

It's a self-hosted web app with 120k+ GitHub stars that connects to Ollama, LM Studio, or any OpenAI-compatible API. Think ChatGPT's interface, but everything runs on your machine. One Docker command to install, no subscription required.

Here's how to set it up and get the most out of it.

Installation

Docker (Recommended)

If Ollama is already running on your machine:

Windows / macOS:

```
docker run -d -p 3000:8080 \
  --add-host=host.docker.internal:host-gateway \
  -v open-webui:/app/backend/data \
  -e WEBUI_SECRET_KEY=your-secret-key-here \
  --name open-webui --restart always \
  ghcr.io/open-webui/open-webui:main
```

Linux:

```
docker run -d --network=host \
  -v open-webui:/app/backend/data \
  -e OLLAMA_BASE_URL=http://127.0.0.1:11434 \
  --name open-webui --restart always \
  ghcr.io/open-webui/open-webui:main
```

On Linux with `--network=host`, the UI is at `http://localhost:8080` instead of port 3000.

Important: Set `WEBUI_SECRET_KEY` to any fixed string. Without it, Open WebUI generates a new key every time the container restarts, which logs you out after every update.

Docker with GPU Passthrough

If you want GPU acceleration for built-in Whisper (voice chat) or RAG embeddings, use the `:cuda` image:

```
docker run -d -p 3000:8080 --gpus all \
  --add-host=host.docker.internal:host-gateway \
  -v open-webui:/app/backend/data \
  --name open-webui --restart always \
  ghcr.io/open-webui/open-webui:cuda
```

You need the NVIDIA Container Toolkit installed first:

```
sudo nvidia-ctk runtime configure --runtime=docker
sudo systemctl restart docker
```

Docker Compose (Open WebUI + Ollama Together)

If you want both in containers:

```
services:
  ollama:
    image: ollama/ollama:latest
    restart: unless-stopped
    volumes:
      - ollama:/root/.ollama

  open-webui:
    image: ghcr.io/open-webui/open-webui:main
    ports:
      - "3000:8080"
    environment:
      - OLLAMA_BASE_URL=http://ollama:11434
      - WEBUI_SECRET_KEY=your-secret-key-here
    depends_on:
      - ollama
    volumes:
      - open-webui:/app/backend/data
    restart: unless-stopped

volumes:
  ollama: {}
  open-webui: {}
```

Save as `docker-compose.yml` and run `docker compose up -d`.

Without Docker (pip)

```
pip install open-webui
open-webui serve
```

Or with `uv` (faster):

```
uvx --python 3.11 open-webui@latest serve
```

Python 3.11 is recommended. Python 3.13 has known dependency issues.

First Run

1. Open `http://localhost:3000` (or `:8080` on Linux with host networking)
2. Create an account — the **first account automatically becomes admin**
3. Select a model from the dropdown — if Ollama is running, your pulled models appear automatically
4. Start chatting

If no models show up, you need to pull one first:

```
ollama pull qwen3:8b
```

Then refresh the model list in Open WebUI (click the refresh icon next to the model selector).

Key Features

Open WebUI isn't just a chat box. Here's what it can actually do.

Multi-Model Support

Switch between models mid-conversation from the dropdown. You can also select **multiple models** to compare responses side by side — useful for evaluating which model handles a task better.

There's even a built-in **Arena Mode** that blindly sends your prompt to two random models, lets you pick the better response, and builds an Elo leaderboard. Handy for figuring out which model to keep.

Document Upload & RAG

Drag files into the chat or upload them in **Workspace → Knowledge**. Open WebUI chunks the documents, embeds them, and stores them in a vector database. Then reference them with `#` in your prompt.

Supported formats include PDF, Word, Excel, PowerPoint, and plain text. The built-in RAG supports hybrid search (BM25 + vector) with re-ranking.

For more on how RAG works under the hood, see our local RAG guide.

Voice Chat

Click the microphone icon to talk to your model. Open WebUI supports several STT/TTS backends:

Component	Options
STT	Local Whisper (built-in, free), OpenAI Whisper API, Deepgram, browser Web Speech API
TTS	Browser Web Speech (default), OpenAI TTS, ElevenLabs, Azure, Edge TTS

For fully local voice chat, set STT to Whisper (local) in **Settings → Audio**. The `:cuda` Docker image accelerates Whisper with your GPU.

For a deeper dive on building voice pipelines, see our voice chat guide.

Note: Microphone access requires HTTPS in production. On localhost it works over HTTP.

Web Search

Open WebUI can search the web and feed results into the model's context. Go to **Admin → Settings → Web Search** and configure a search provider.

SearXNG is the recommended option (free, self-hosted):

```
# Add SearXNG to your Docker Compose
docker run -d -p 8888:8080 \
  -v searxng:/etc/searxng \
  --name searxng --restart always \
  searxng/searxng
```

Then in Open WebUI: set `RAG_WEB_SEARCH_ENGINE=searxng` and point the query URL to `http://searxng:8080/search?q=<query>`.

Other supported providers: Brave Search, Tavily, Kagi, Mojeek, Perplexity, and about 10 more.

Image Generation

Connect Open WebUI to a local image generation backend:

Backend	Setup
AUTOMATIC1111	Run with `--api --listen`, set the base URL in Open WebUI
ComfyUI	Export workflow JSON, import into Open WebUI
OpenAI DALL-E	Add your API key (cloud, not local)

Enable in **Admin → Settings → Images**. Once configured, models can generate images inline in chat.

Model Management

Pull models directly from the UI — paste a model name like `qwen3:14b` into the model selector. You can also create **custom model presets** in **Workspace → Models**: wrap any base model with a system prompt, attach knowledge collections, enable specific tools, and save it as a reusable preset.

Connecting Multiple Backends

Open WebUI isn't limited to Ollama. Any OpenAI-compatible API works. Go to **Admin → Settings → Connections → OpenAI** and click "Add Connection."

Backend	API URL	Notes
Ollama	`http://localhost:11434`	Built-in connection, auto-configured
LM Studio	`http://localhost:1234/v1`	Start LM Studio's local server first
llama.cpp server	`http://localhost:8080/v1`	The `/v1` suffix is required
vLLM	`http://localhost:8000/v1`	Full OpenAI-compatible API
OpenAI	`https://api.openai.com/v1`	Requires API key
OpenRouter	`https://openrouter.ai/api/v1`	Cloud gateway to many models

Not sure whether to use Ollama or LM Studio as your backend? See our Ollama vs LM Studio comparison.

All connected backends appear in the same model dropdown. You can mix local and cloud models freely — use a local 8B for quick tasks and route to GPT-4 or Claude when you need more firepower.

Customization Worth Knowing

System Prompts & Presets

Go to **Workspace → Models**, click “+” to create a new preset. Pick a base model, write a system prompt, and save. Your custom model appears in the dropdown alongside the base models.

Good uses: a “coding assistant” preset with a code-focused system prompt, a “writing editor” with grammar rules, a “research assistant” with web search enabled.

Pipelines & Tools

Open WebUI has a plugin system for extending functionality with custom Python code. Use cases include:

- Rate limiting and usage monitoring
- Toxic message filtering
- PII redaction before messages hit the model
- Custom tool integrations

The community maintains 15+ tools at openwebui.com. For most users, the built-in features are enough.

Multi-User Setup

Open WebUI supports multiple users out of the box. The first account is admin. New signups default to “pending” status — the admin approves them before they can chat.

Three roles: **Admin** (full control), **User** (standard access), **Pending** (waiting for approval).

For teams, there’s group-based permissions, LDAP/SSO integration, and even SCIM provisioning for automated user management. Overkill for personal use, but it’s there if you share your setup with family or a small team.

Performance Tips

Set a task model. Open WebUI uses an LLM for background tasks like generating chat titles and tags. By default, it uses your main model — which is slow if you're running a 32B. Go to **Admin → Settings → Interface** and set the task model to something fast like `qwen3:4b`.

Adjust context length carefully. Default is 2048 tokens for Ollama models. You can increase it per model in settings, but longer context means more VRAM usage and slower responses.

Disable proxy buffering behind Nginx. If you put Open WebUI behind a reverse proxy, streaming breaks unless you add:

```
proxy_buffering off;
proxy_cache off;
```

Switch to PostgreSQL for heavy use. The default SQLite database works fine for personal use. If you have many users or large chat histories, set `DATABASE_URL` to point at PostgreSQL for better performance.

Common Problems

Open WebUI can't find Ollama. This is the #1 issue. The fix depends on your setup:

Setup	Fix
Linux, Ollama on host	Use `--network=host` in Docker
Windows/macOS, Ollama on host	Use `--add-host=host.docker.internal:host-gateway`
Both in Docker Compose	Use the service name: `http://ollama:11434`
Still broken	Make sure Ollama allows external connections: set `OLLAMA_HOST=0.0.0.0`

See our Ollama troubleshooting guide for more fixes.

Models don't show in the dropdown.

- Run `ollama list` to verify models are actually pulled

- Check Admin → Settings → Models — make sure visibility is set to "public"
- For external APIs, verify the URL includes `/v1` and the API key is valid
- Set `AIOHTTP_CLIENT_TIMEOUT_MODEL_LIST=3` to fail fast on unreachable endpoints

Logged out after every container update. You didn't set `WEBUI_SECRET_KEY`. Add `-e WEBUI_SECRET_KEY=any-fixed-string` to your Docker command.

Voice input doesn't work. Microphone requires HTTPS (or localhost). If you're accessing Open WebUI over your LAN via IP address, you need to set up HTTPS with a reverse proxy.

Slow response streaming. If behind Nginx, add `proxy_buffering off;` to your config. Also check that your Ollama model isn't spilling to system RAM — that's a 30x slowdown. See our VRAM requirements guide.

Open WebUI vs Alternatives

Open WebUI isn't the only option. Here's when to use something else:

Tool	Best For	Ollama	RAG	Voice	Multi-User
Open WebUI	General-purpose local AI chat	Native	Excellent	Yes	Yes (SSO, RBAC)
SillyTavern	Roleplay, fiction, character cards	Yes	Yes	Yes	No
Text Gen WebUI	Power users, direct model loading	No (is its own backend)	Partial	Yes	No
AnythingLLM	RAG-first, desktop app	Yes	Best-in-class	Basic	Yes
LibreChat	Multi-provider (OpenAI + Claude + local)	Yes	Yes (separate service)	No	Yes (SSO)

Choose Open WebUI if you want the most polished, feature-complete interface for Ollama with RAG, voice, and web search built in.

Choose SillyTavern if you're into roleplay, character-based interactions, or want fine-grained sampler controls (temperature, top-k, mirostat).

Choose Text Generation WebUI if you want to load models directly (GGUF, GPTQ, EXL2) without Ollama as a middleman, or you need LoRA loading at runtime.

Choose AnythingLLM if RAG is your primary use case and you want a desktop app that just works — no Docker required.

Choose LibreChat if you need to switch between OpenAI, Claude, Gemini, and local models in one interface.

Bottom Line

Open WebUI is the default recommendation for anyone running Ollama who wants a proper interface. One Docker command, works immediately, and the feature set rivals commercial tools.

The setup path:

1. Make sure Ollama is running with at least one model pulled
2. Run the Docker command for your OS (above)
3. Open localhost:3000, create your account, start chatting

From there, add features as you need them — RAG for document Q&A, voice chat for hands-free interaction, web search for current information. It all works out of the box.

For choosing which model to chat with, see our best local LLMs for chat guide.

Source: https://insiderllm.com/guides/open-webui-setup-guide/

Free guides for running AI locally