ChatGPT

Tool Use Servers & Integrations for OpenWebUI

Open WebUI is a self-hosted AI platform that natively supports Retrieval Augmented Generation (RAG) and “bring-your-own” Python function calling. For example, its **Tools Workspace** includes a built-in editor and allows users to add custom Python functions which the LLM can invoke [1]. It also offers a rich RAG engine: users can upload local documents or point to URLs (using a `#` prefix) so that the model retrieves and cites relevant content during chat [1]. You can even perform live web searches or browse websites within the chat, injecting search results or page content as context [2]. Open WebUI’s documentation explicitly shows how to configure RAG to index technical documentation – e.g. by downloading all `.md` docs, uploading them as a knowledge base, and then querying them to get precise, citation-backed answers [3] [4].

Official Tool Server Repos and Plugins

Open WebUI’s team maintains a collection of **OpenAPI-based tool server examples** on GitHub. The openapi-servers repository provides ready-to-run FastAPI servers for common tasks, all accessible via standard HTTP endpoints. For instance, reference implementations include a *Filesystem* server (to safely read/write files), a *Git* server (expose repo content), a *Memory/Knowledge Graph* server (persistent semantic memory), a *Weather* server (real-time forecasts), and more [5]. Each example comes with instructions to launch it (e.g. `uvicorn main:app`) or via Docker Compose [6] [7]. Once running, you point Open WebUI’s **Settings → Tools** to the server’s URL. The integration docs show a demo using a “time” server: clone `openapi-servers`, `cd servers/time`, install requirements, and run `uvicorn main:app --host 0.0.0.0` [6]. Then in Open WebUI’s browser interface (⚙ Settings → Tools), add that `http://localhost:8000` endpoint; Open WebUI will confirm the connection with a tool icon in the chat window [6] [8]. (Global admin can similarly add tools under **Admin → Tools** for all users [9].)

Open WebUI also supports tools written using the older *MCP* (Model Context Protocol) format by bridging them via the **mcpo** proxy. The mcpo GitHub (“MCP-to-OpenAPI proxy”) lets you take any MCP tool (which normally reads from stdin/stdout) and expose it as a secure HTTP server [10]. For example, to run the provided “time” MCP server, you can simply execute:

```
uvx mcpo --port 8000 -- uvx mcp-server-time --local-timezone=America/New_York
```

This starts an OpenAPI server on port 8000; visiting `http://localhost:8000/docs` shows automatically generated docs. Open WebUI can then connect to each proxied tool endpoint (e.g. `http://localhost:8000/time`) as above [10] [11]. (The openapi-servers repo even includes an `mcp-proxy` example and mentions using mcpo or a Python proxy to bridge MCP tools [12].)

Open WebUI’s **Pipelines plugin framework** is another official extension for tool use. The open-webui/pipelines project lets you run “plugin” servers that speak the OpenAI API. In practice, you launch your pipeline service (e.g. a FastAPI or Flask app) and then point Open WebUI’s OpenAI API URL to it. This allows offloading heavy tasks or chaining logic outside the main UI. The Open WebUI docs note that you can “set the OpenAI URL to the Pipelines URL" to unlock advanced features like function calling, usage monitoring, translation, toxicity filtering, and more [13]. For example, community-supplied pipeline examples include complete function-calling agents or rate-limiting services that plug right into Open WebUI without modifying the UI code [13].

Community Tools: In addition to the official repos, the Open WebUI community shares plug-and-play "Tools" you can import via the UI (⚙ → Tools → Community Library). These are Python scripts (for example, web scrapers, SQL query tools, YouTube analyzers, agent frameworks, etc.) that you can add with a click and then enable for your chat model. A popular collection is MartianInGreen's OpenWebUI-Tools repo, which includes advanced agents and web query tools. To install, you usually just paste the code into a new Function in **Workspaces → Functions → New**, or import via the community library. Once added, a Tool shows up in the chat UI (button) and can be enabled per session or by default in model settings [14] [15].

Integrations with Hosted LLM Services

Open WebUI can also use third-party hosted APIs as backends. Because it is OpenAI-compatible, you can plug in services like **OpenRouter**, **Mistral**, **GroqCloud**, **Anthropic**, etc. simply by configuring the API URL and key. For example, the official documentation highlights that you can "customize the OpenAI API URL to link with … OpenRouter" [16]. Indeed, OpenRouter (a unified LLM API) can serve as the model endpoint: set the OpenAI-compatible base URL to `https://openrouter.ai/api/v1` and your API key. Several community Functions exist to leverage OpenRouter: e.g., scripts that fetch available models from OpenRouter's API and insert them into WebUI's model list, or that send chat messages through OpenRouter with support for citations and reasoning tokens [17] [18]. In practice, the integration is seamless: when you run Open WebUI (Docker or pip), just enter your OpenRouter key in **Connections → OpenAI API** (or set the `OPENAI_API_KEY`/`OPENAI_API_BASE_URL` environment variables). OpenWebUI will then call OpenRouter as if it were OpenAI (which grants access to dozens of models and built-in routing) [16] [19].

Tutorials and Walkthroughs

The OpenWebUI docs and community have several step-by-step guides. The **OpenAPI Tool Servers** section shows exactly how to run and attach a server: start the reference time server locally, then use the Settings UI to connect it [6] [8]. The **Tools** guide explains how to import and enable tools via the web interface [20] [21]. For RAG, there are full walkthroughs. For instance, the official "RAG Tutorial" demonstrates loading all of OpenWebUI's documentation into the system. It guides you to download the docs ZIP from GitHub, upload the `.md` files as a knowledge base, and then ask questions – letting the LLM answer using retrieved content [3] [22]. (In fact, a blog post shows how this yields precise, citation-backed answers: after indexing 40k Wikipedia articles, the system correctly explained the history of Arundel House and cited the passage [4].)

There are also video tutorials and articles. For example, several YouTube walkthroughs cover "OpenWebUI Tools and Agents" and RAG setups. These often demonstrate the UI workflows (enabling tools in chat, adjusting model settings, etc.) and even show advanced agents selecting tools (ReAct/SMART patterns). In summary, most tool-use features in Open WebUI are designed to be "plug-and-play": you either import a script from the community library or spin up a Docker container (for openapi-servers or mcpo), then use the intuitive UI to connect it – no deep coding required.

Sources: Official OpenWebUI documentation and GitHub (e.g. the open-webui and openapi-servers repos) describe tool integrations and features [16] [1] [5] [6] [20]. Tutorials and community posts provide examples of RAG and tool use (see e.g. OpenWebUI's RAG tutorial and community blogs) [3] [4].

---

[1] [2] [13] [16] GitHub - open-webui/open-webui: User-friendly AI Interface (Supports Ollama, OpenAI API, ...)
https://github.com/open-webui/open-webui

[3] [22] Open WebUI RAG Tutorial | Open WebUI
https://docs.openwebui.com/tutorials/tips/rag-tutorial/

[4] Open WebUI tutorial — Supercharge Your Local AI with RAG and Custom Knowledge Bases | by Peter Alexandru Hautelman | Mar, 2025 | Medium
https://medium.com/@hautel.alex2000/open-webui-tutorial-supercharging-your-local-ai-with-rag-and-custom-knowledge-bases-334d272c8c40

[5] [7] [12] GitHub - open-webui/openapi-servers: OpenAPI Tool Servers
https://github.com/open-webui/openapi-servers

[6] [8] [9] [11] Open WebUI Integration | Open WebUI
https://docs.openwebui.com/openapi-servers/open-webui/

[10] ○ MCP Support | Open WebUI
https://docs.openwebui.com/openapi-servers/mcp/

[14] [15] [20] [21] ⚙ Tools | Open WebUI
https://docs.openwebui.com/features/plugin/tools/

[17] OpenRouter with Dynamic Model Fetching Tool • Open WebUI Community
https://openwebui.com/t/yikesawjeez/yikes_openrouter

[18] OpenRouter Function • Open WebUI Community
https://openwebui.com/f/rburmorrison/openrouter

[19] OpenRouter Quickstart Guide | Developer Documentation | OpenRouter | Documentation
https://openrouter.ai/docs/quickstart