Paolo Dalprato

Ollama Easy GUI

Last updated: January 17, 2026

A graphical interface for using AI models on your computer, without subscriptions and without sending data to external services.

Why use AI locally

When you use ChatGPT, Claude or other online AI assistants, your conversations travel to remote servers. For many uses that's perfectly fine, but there are situations where you'd prefer your data to stay on your computer: confidential business documents, personal reflections, projects you don't want to share with anyone.

Ollama Easy GUI solves this problem: it's an application that lets you chat with AI models that run entirely on your PC. No data leaves your computer, no subscription to pay, no limit on the number of messages.

The Github repository is ollama-easy-gui (https://github.com/paolodalprato/ollama-easy-gui)

What you can do with this application

The application offers the essential features for working with local AI:

- **Chat with AI models** choosing from dozens of free options
- **Attach documents** for analysis: PDFs, images, Word files and text
- **Save and export** conversations in various formats
- **Customize the behavior** of models with permanent instructions
- **Extend AI capabilities** by allowing it to read files or search for information

Essential requirements

To use Ollama Easy GUI you need:

- A computer with Windows, macOS or Linux
- At least 8 GB of RAM (16 GB recommended)

- Disk space for AI models (from 2 to 50 GB depending on the model)
- An internet connection for the initial download

> **Do I need a GPU?**
>
> No, it's not mandatory. Models work with CPU only, although more slowly, but a suitable NVIDIA graphics card for the models you intend to use significantly speeds up responses.

> **This manual is for Windows**
>
> The application was developed and tested on Windows. It should also work on macOS and Linux, but these systems haven't been tested yet. The instructions in this manual refer to Windows; if you use another operating system, see the appendix Notes for macOS and Linux (https://docs.ai-know.pro/ollama-easy-gui-en/other-systems/) for the main differences.

How this manual is organized

The manual guides you step by step from installation to advanced use:

1. **Preparing your computer**: installing the necessary programs
2. **Installation**: downloading and launching Ollama Easy GUI
3. **Your first chat**: creating a conversation and choosing a model
4. **Managing models**: downloading new models and understanding which to choose
5. **Customizing responses**: configuring AI behavior
6. **Attachments and export**: working with documents and saving conversations
7. **MCP: extending AI**: allowing AI to use external tools
8. **Troubleshooting**: what to do when something doesn't work

Who this is for

This manual is intended for those who:

- Are familiar with computer use but not with programming
- Want to try local AI without having to learn terminal commands
- Are looking for an alternative to cloud services for privacy or cost reasons
- Are curious to understand how AI models work "behind the scenes"

No advanced technical skills are required: where necessary, procedures are explained step by step with images.

Table of contents

- Hiding the sidebars
- Closing the application

Your first chat

- Creating a new conversation
- Sending a message
- Managing conversations
- Exporting a response
- Changing model during conversation
- The first experiment
- Tips for effective prompts

Managing models

- Local models vs Hub
- Installed models
- Downloading new models
- Understanding model names
- Which model to choose
- Removing a model
- Models and MCP

Customizing responses

- What is a system prompt
- Base prompt: personality per model
- Global system prompt: universal instructions
- How they work together
- Best practices

Attachments and export

- Attaching files
- Attachment limits

- Exporting conversations
- Where data is saved
- Export formats compared

MCP: extending AI

- What is MCP in simple terms
- Compatible models
- Activating MCP
- Configuring tools
- MCP in action
- Repositories
- Privacy and MCP
- MCP troubleshooting

Troubleshooting

- Installation problems
- Startup problems
- Model problems
- Attachment problems
- MCP problems
- Checking the logs
- Getting support

Notes for macOS and Linux

- macOS
- Linux
- Common differences
- Known problems
- Not tested

Preparing your computer

Before installing Ollama Easy GUI, you need to prepare your computer with three support programs. Don't worry if you don't know them: I'll guide you step by step.

Prerequisites overview

Program	What it's for	Estimated time
Ollama	Runs AI models on your computer	5 minutes
Git	Downloads the application from the internet	3 minutes
Node.js	Runs the graphical interface	3 minutes

Check first

You might already have some of these programs installed. To verify, open Command Prompt (search for "cmd" in the Start menu) and type the commands indicated in each section.

Installing Ollama

Ollama is the "engine" that runs AI models. Without Ollama, the graphical interface would have nothing to run.

Procedure for Windows

1. Go to ollama.ai (https://ollama.ai)
2. Click the **Download for Windows** button
3. Run the downloaded file and follow the installation instructions
4. When finished, Ollama will start automatically (you'll see a small icon in the notification area)

Verify the installation

Open Command Prompt and type:

```
ollama --version
```

If you see a version number (e.g. "ollama version 0.5.4"), the installation was successful.

Download a first model

Ollama alone doesn't include AI models: you need to download at least one. To start, I recommend **llama3.2** which is lightweight and works well on most computers.

In Command Prompt type:

```
ollama pull llama3.2
```

The download takes a few minutes (the model is about 2 GB). Once completed, the model will be available forever on your computer.

> **Alternative models**
>
> You can download other models later directly from the Ollama Easy GUI interface, without using Command Prompt.

Installing Git

Git is a tool that developers use to share software. In our case, it's used to download Ollama Easy GUI from GitHub, the site where the code is published.

Procedure for Windows

1. Go to git-scm.com (https://git-scm.com)
2. Click on **Download for Windows**
3. Run the downloaded file
4. During installation, you can leave all default options (click "Next" until the end)

Verify the installation

Close and reopen Command Prompt (important: it must be a new window), then type:

```
git --version
```

If you see something like "git version 2.47.1", Git is installed correctly.

Installing Node.js

Node.js is an environment that allows running applications written in JavaScript. Ollama Easy GUI uses Node.js to run its graphical interface.

Procedure for Windows

1. Go to nodejs.org (https://nodejs.org)
2. Download the **LTS** version (Long Term Support), the one recommended for most users
3. Run the downloaded file and follow the instructions

Verify the installation

Close and reopen Command Prompt, then type:

```
node --version
```

You should see a version number (e.g. "v22.12.0"). If the number starts with 16 or higher, you're all set.

Summary

If you followed all the steps, you now have:

- [x] Ollama installed and running
- [x] At least one AI model downloaded (llama3.2)
- [x] Git ready to download software
- [x] Node.js ready to run the application

You're ready to install Ollama Easy GUI. Move on to the next chapter.

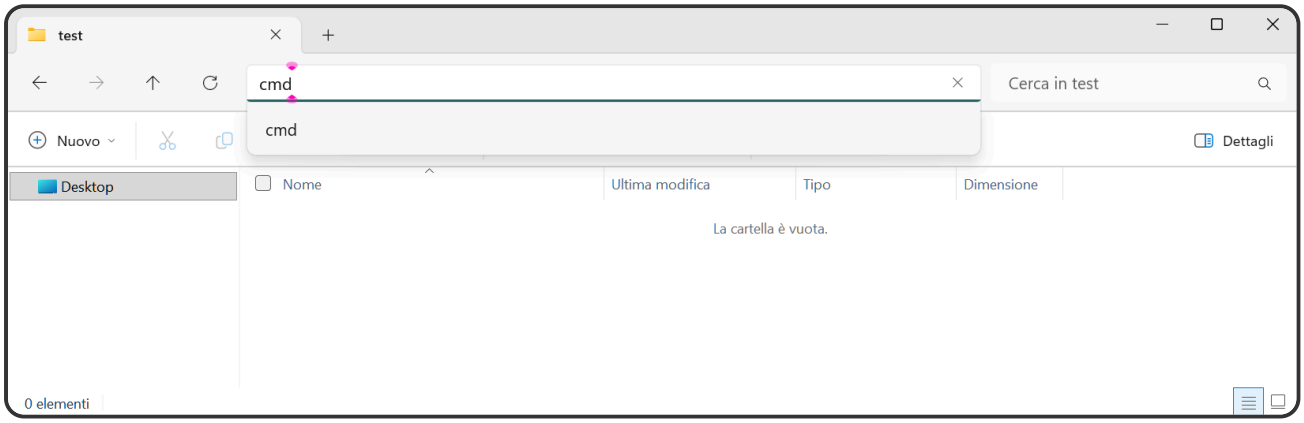
> ⚠ **Problems?**
>
> If something doesn't work, check the Troubleshooting (https://docs.ai-know.pro/ollama-easy-gui-en/troubleshooting/) section. The most common problems involve administrator permissions during installation or the need to restart the computer.

Installation

With the prerequisites ready, you can now download and launch Ollama Easy GUI. The procedure takes just a few minutes.

Downloading the application

Choose a folder where you want to install the application: it can be an existing folder or you can create a new one. Open the folder with File Explorer, in the address bar at the top type **cmd** and press Enter, Command Prompt opens already positioned in the installation folder.



(https://docs.ai-know.pro/ollama-easy-gui-en/img/cmd.png)

Now download the application with Git:

```
git clone https://github.com/paolodalprato/ollama-easy-gui.git
```

Git will create a folder called `ollama-easy-gui` with all the necessary files.

Installing dependencies

Enter the newly created folder:

```
cd ollama-easy-gui
```

Install the libraries required by the application:

```
npm install
```

This command downloads some additional components. The operation takes about a minute and you only need to run it the first time.

Launching the application

You have two ways to launch Ollama Easy GUI: the quick method and the method with configuration.

Quick method (npm start)

The simplest way: open Command Prompt in the application folder and type:

```
npm start
```

The application starts and shows the address where you can reach it, typically `http://localhost:3003`. Open this address in your browser to use the interface.

Method with configuration (.bat file)

On Windows you can use a batch file that offers additional features: update checking and optimal configuration for your graphics card.

First create the configuration file: in the application folder you'll find the file `start-ollama-easy-gui.bat.example`. Copy it and rename it to `start-ollama-easy-gui.bat` (you can also give it another name, the important thing is that the extension is `.bat`).

Then edit the file with a text editor (Notepad works fine) to adapt it to your hardware:

```
:: Number of layers to run on the GPU (higher = faster, but requires more VRAM)
set "OLLAMA_GPU_LAYERS=18"

:: Enable Flash Attention for faster responses
set "OLLAMA_FLASH_ATTENTION=1"

:: CPU threads to use for inference
set "OLLAMA_NUM_THREADS=24"
```

What values to use?

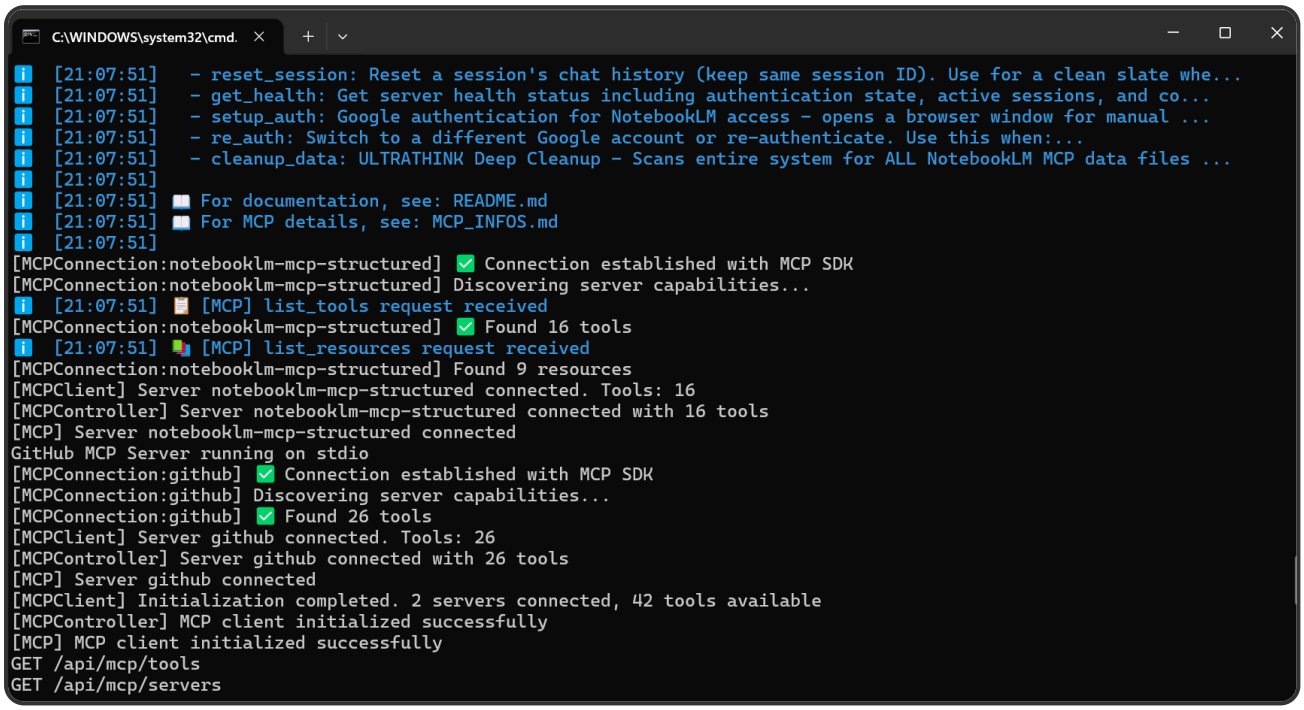
- **OLLAMA_GPU_LAYERS**: with 6 GB of VRAM you can try 20-25, with 8 GB try 30-35
- **OLLAMA_NUM_THREADS**: set a number equal to your CPU cores (e.g. 8 for a quad-core with hyperthreading)

From now on, launch the application by double-clicking the `start-ollama-easy-gui.bat` file. At each startup the file automatically checks for updates and installs them.

Accessing the interface

Whichever method you used, the interface will be available in the browser at:

```
http://localhost:3003
```



(https://docs.ai-know.pro/ollama-easy-gui-en/img/terminale-avvio.png)

If there are no error messages in the terminal window (as in the screenshot), the application has started correctly. Open the address in your browser to access the graphical interface.

Updating the application

If you use the `.bat` file, updates are automatically downloaded and installed at each startup.

If you launch with `npm start` instead, you can update manually:

1. Open Command Prompt in the application folder
2. Download updates:

   ```
   git pull
   ```

3. Update dependencies:

   ```
   npm install
   ```

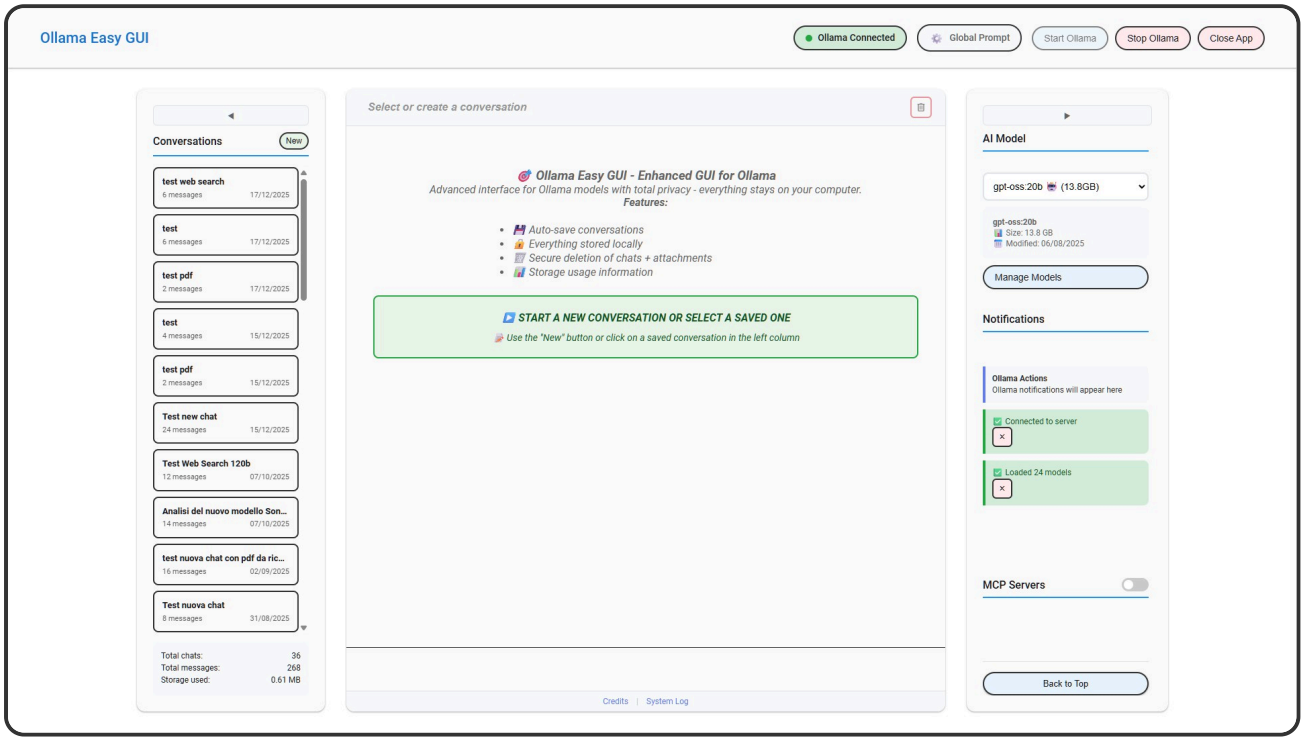
Your settings are safe

Updates don't overwrite your conversations, saved prompts or custom configurations. All your data is in the `app/data` folder which is not touched by updates.

The interface

Before you start using the application, let's take a moment to get familiar with the interface and its components.

General overview



(https://docs.ai-know.pro/ollama-easy-gui-en/img/start.jpg)

The interface is organized into four main areas:

- **Top bar**: title, Ollama status and global controls
- **Left sidebar**: conversation list
- **Central area**: the chat itself
- **Right sidebar**: model selection and settings

Top bar

The top bar contains the main application controls:

- **Title**: "Ollama Easy GUI" - clicking it opens the credits with version, author and license information
- **Ollama Status**: visual indicator showing whether the Ollama service is active or not

- **Global Prompt**: opens the editor for instructions that apply to all models (detailed in the Customization chapter)
- **Start Ollama**: starts the Ollama service if it's not active
- **Stop Ollama**: stops the Ollama service
- **Close App**: completely closes the application

Left sidebar

The left column manages conversations:

- **New button**: creates a new conversation
- **Conversation list**: all saved chats, sorted by date

Clicking on a conversation opens it in the central area.

At the bottom of the sidebar you'll find some **statistics**:

- Total number of chats
- Total number of messages
- Disk space used

Central area

This is the main space where the conversation takes place:

- **Chat title**: at the top, shows the current conversation name. It's editable text: click on it to rename the chat
- **Messages**: your messages and the model's responses, in chronological order
- **Input area**: at the bottom, where you write messages
- **Attach button** (paperclip): to add files to the conversation
- **Send button**: sends the message (or press Enter)

On each model response you'll find icons to:

- **Copy** the text to clipboard
- **Export** the response in various formats (Markdown, text, Word)

Central area footer

At the bottom of the central area you'll find two elements:

- **Log**: opens the application log viewer, useful for diagnosing problems. Logs are divided by category (App, Chat, MCP, Models). For details see the "Checking the logs" section in the Troubleshooting chapter
- **Credits**: shows application information (version, author, license)

Right sidebar

The right column contains the main settings:

Model selection

The **AI Model** dropdown shows all available models. The model selected here becomes the default for new conversations.

Models with an asterisk (*) after their name have a custom system prompt.

Manage Models button

Opens the model management window with two tabs:

- **Local Models**: models already downloaded to the computer
- **Download from Hub**: online catalog to download new models

MCP section

When you activate the MCP switch, the panel of configured MCP servers appears. Each server shows:

- **Name and description**
- **Connection status**: 🟢 connected, 🔴 disconnected, ⚪ disabled
- **Enable/Disable button**: to activate or deactivate the server

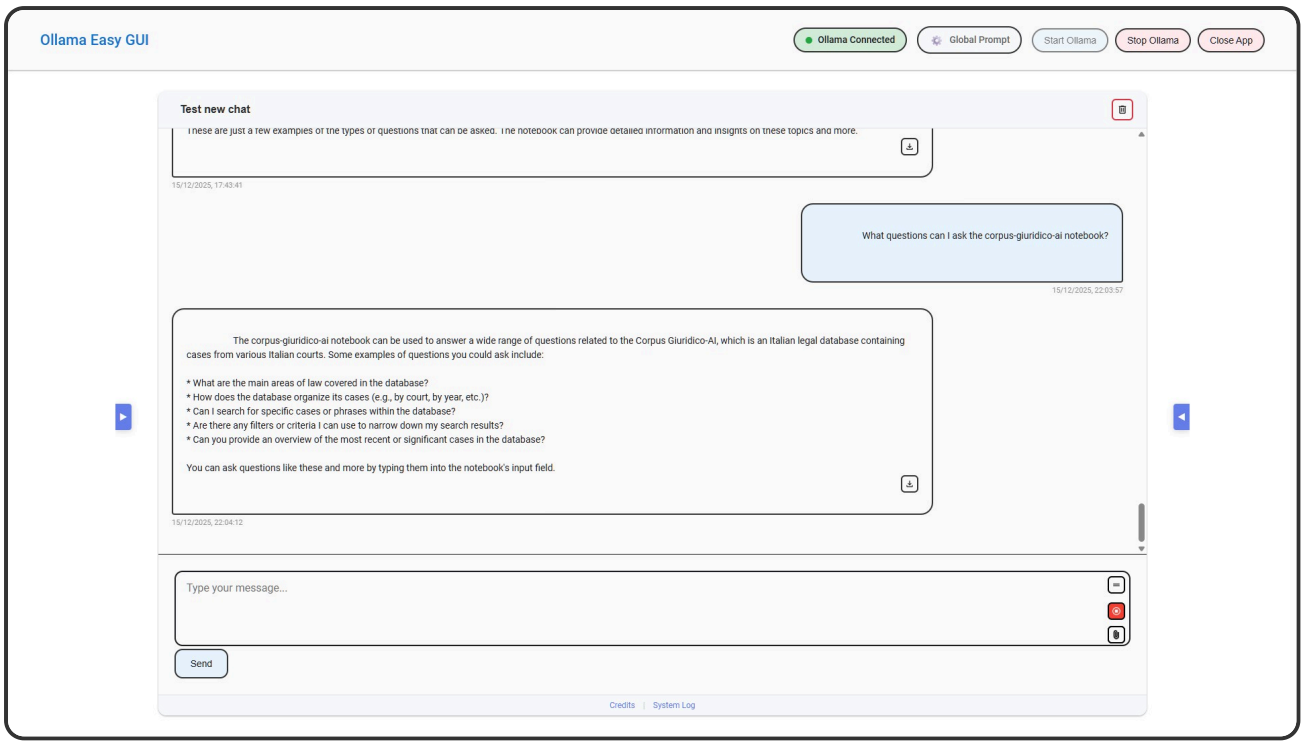
The MCP topic is covered in depth in its dedicated chapter.

Notifications

At the bottom of the sidebar, application notifications appear: confirmations of completed operations, warnings and any errors.

Hiding the sidebars

On small screens or if you prefer more space for the chat, you can hide both sidebars by clicking the arrows at the edges. The interface adapts to show only the central area.



(https://docs.ai-know.pro/ollama-easy-gui-en/img/sidebar-collassata.jpg)

To reopen a sidebar, click the corresponding arrow again.

Closing the application

To close Ollama Easy GUI you have two options:

- **Close App** in the top bar: closes the application cleanly, stopping all processes
- **Close the Command Prompt window**: if you launched the app from terminal, closing that window terminates the application

Ollama keeps running

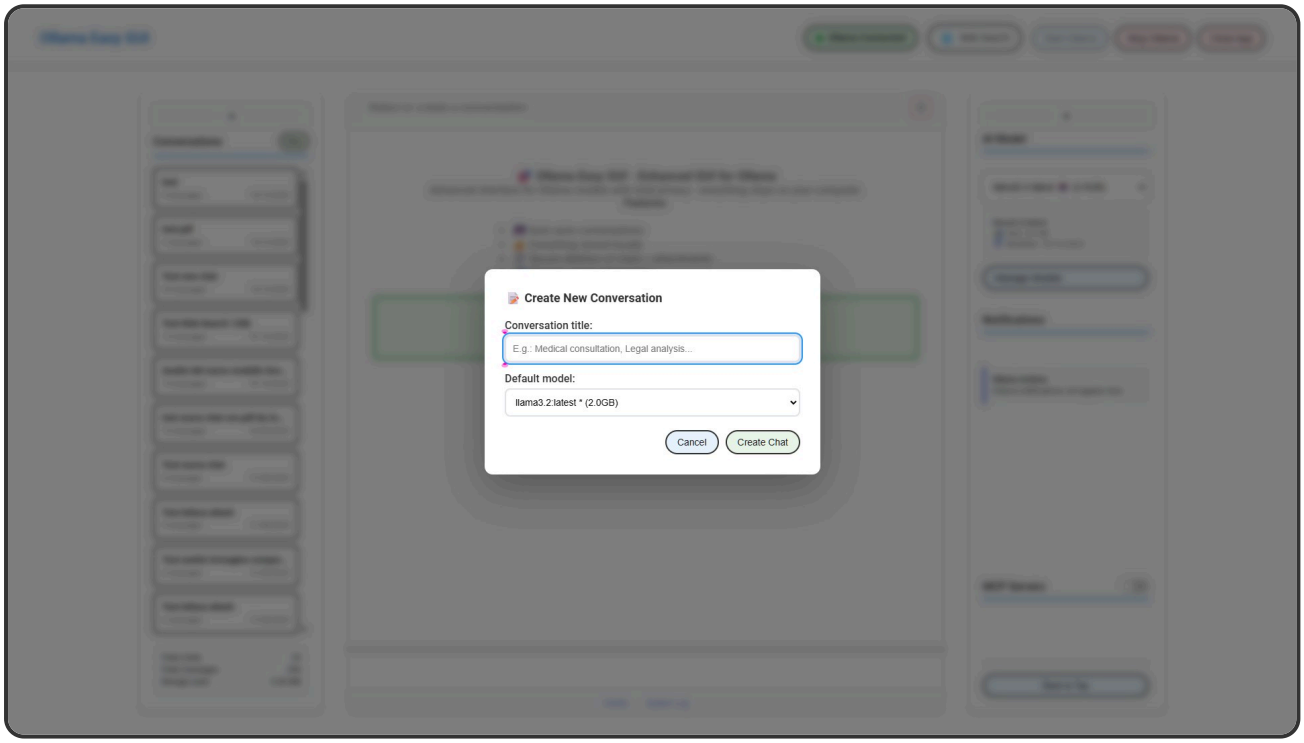
Closing Ollama Easy GUI doesn't stop the Ollama service, which continues running in the background. If you want to stop Ollama too, use the **Stop Ollama** button before closing the app, or close Ollama from its icon in the Windows notification area.

Your first chat

Now that you know the interface, let's see how to use it to chat with an AI model.

Creating a new conversation

To start a chat, click the **New** button at the top of the left sidebar. A popup opens that lets you select the model to use:



(https://docs.ai-know.pro/ollama-easy-gui-en/img/nuova-chat.jpg)

The popup shows the default model (the one selected in the right sidebar), but you can change it for this specific conversation. Confirm to open a new chat ready to receive your first message.

Default model

The model selected in the right sidebar becomes the default for new conversations. If you always use the same model, set it there so you don't have to choose it every time.

Sending a message

Write your message in the text area at the bottom and press **Enter** or click the send button. The model will start responding immediately, with text appearing progressively like in a normal chat.

> **Streaming responses**
>
> Responses appear word by word as they're generated. This lets you read the beginning of the response without waiting for it to complete.

Managing conversations

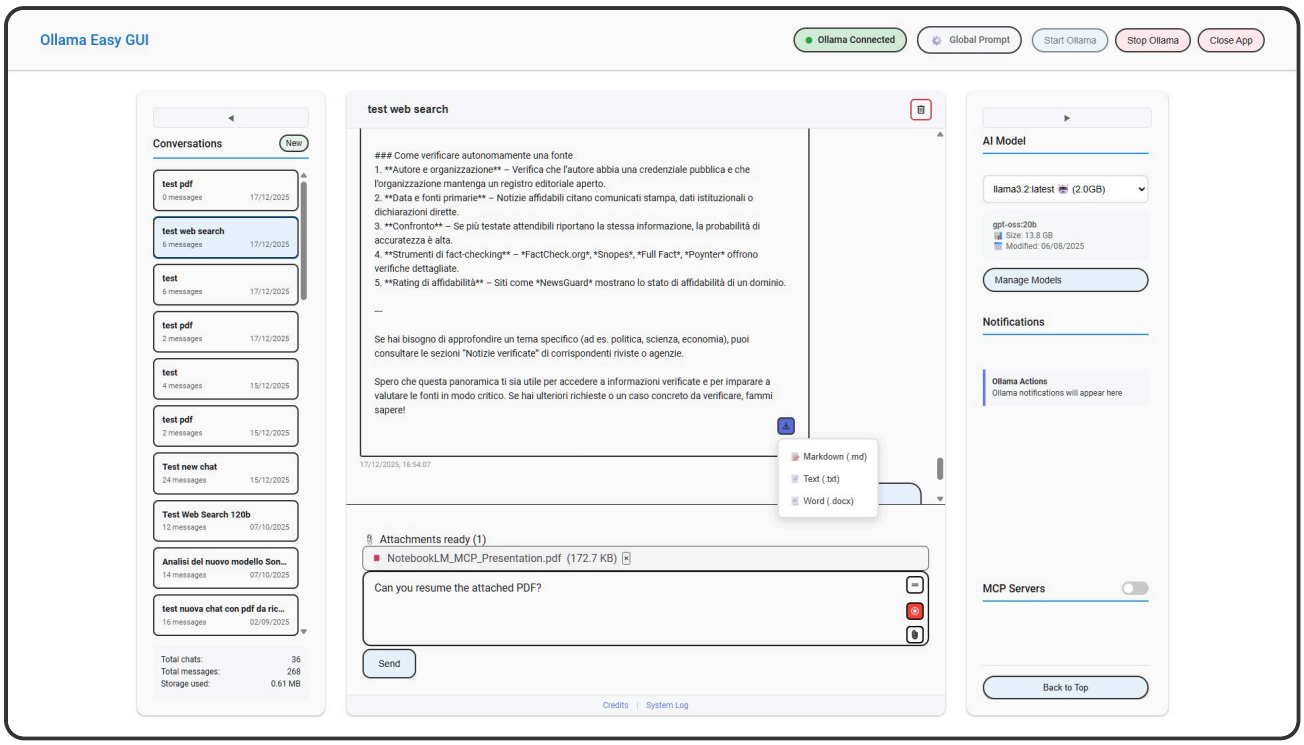
Each conversation is automatically saved in the left sidebar. You can:

- **Rename** a conversation by clicking on its title (when it's open in the central area)
- **Resume** a past conversation by selecting it from the list
- **Delete** a conversation with the trash button

Conversations are saved on your computer in the `app/data/conversations` folder. Each conversation has its own subfolder containing both messages and any attachments.

Exporting a response

You can save a single response in different formats by clicking the download icon at the bottom right of each response:



(https://docs.ai-know.pro/ollama-easy-gui-en/img/chat-con-download.jpg)

Available formats are:

- **Markdown** (.md): ideal if you use editors that support it
- **Text** (.txt): the most universal format
- **Word** (.docx): for formal documents

Changing model during conversation

You can change models at any time by selecting a different one from the dropdown menu in the right sidebar. The new model will continue the conversation from where you left off, but will have a different "personality" based on its characteristics.

> **Model behavior**
>
> Different models respond differently. Some are more creative, others more precise, others better suited for specific tasks like programming. Experiment to find the one you prefer.

The first experiment

Try sending a simple message to verify everything works:

```
Hello! Can you briefly introduce yourself?
```

If you receive a response, congratulations: you've just run your first local AI inference. Everything happened on your computer, without any data being sent to the internet.

Tips for effective prompts

To get better responses:

- **Be specific**: instead of "tell me about history", ask "summarize the causes of World War I in 5 points"
- **Provide context**: explain what you want to achieve and why
- **Request a format**: specify if you want a list, a paragraph, a table
- **Iterate**: if the first response isn't perfect, ask to modify or elaborate

Local models work better with clear instructions. In the next chapter we'll see how to download additional models and choose the one best suited to your needs.

Managing models

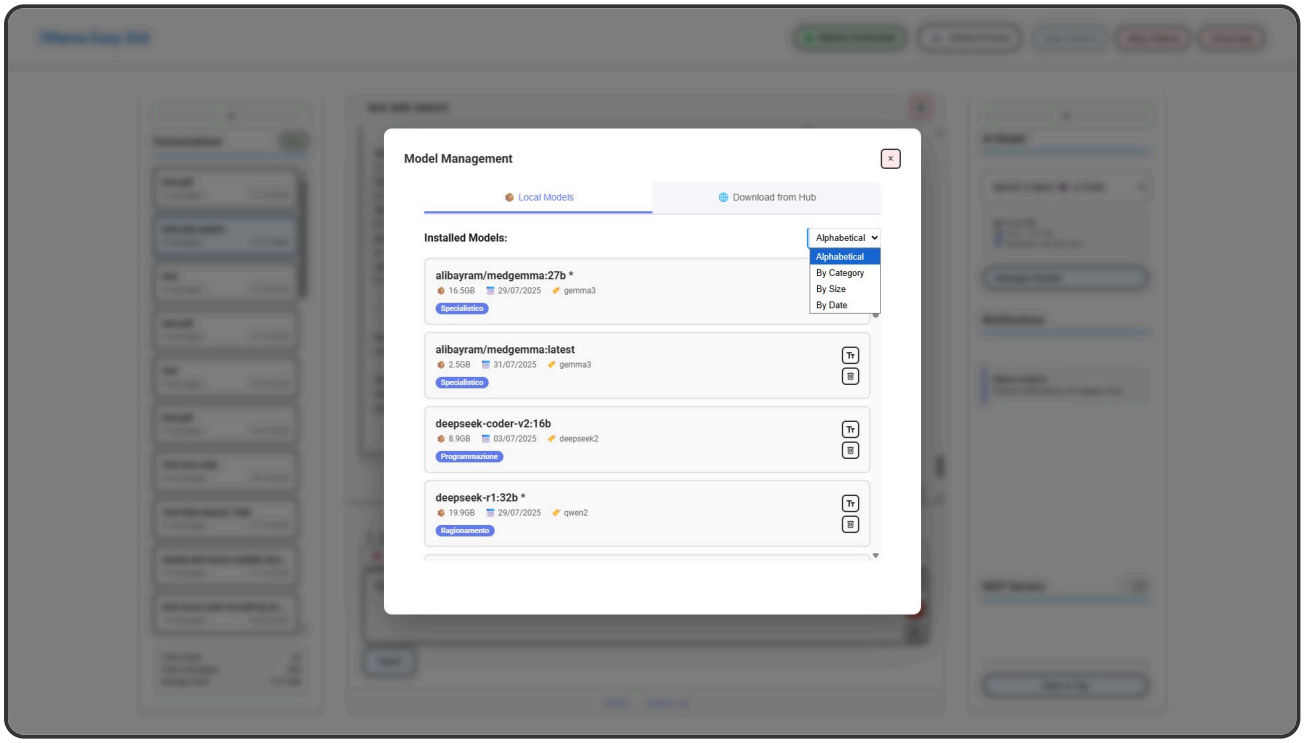
AI models are the heart of the application. In this chapter we'll see how to download new models, understand the differences between them and choose the one best suited to your needs.

Local models vs Hub

The model management interface has two tabs:

- **Local Models**: models already downloaded on your computer, ready to use
- **Hub Search**: online catalog of models available for download

To access model management, click the **Manage Models** button in the right sidebar.



(https://docs.ai-know.pro/ollama-easy-gui-en/img/modelli-locali.jpg)

Installed models

In the local models tab you see all the models available on your computer. For each model the following are shown:

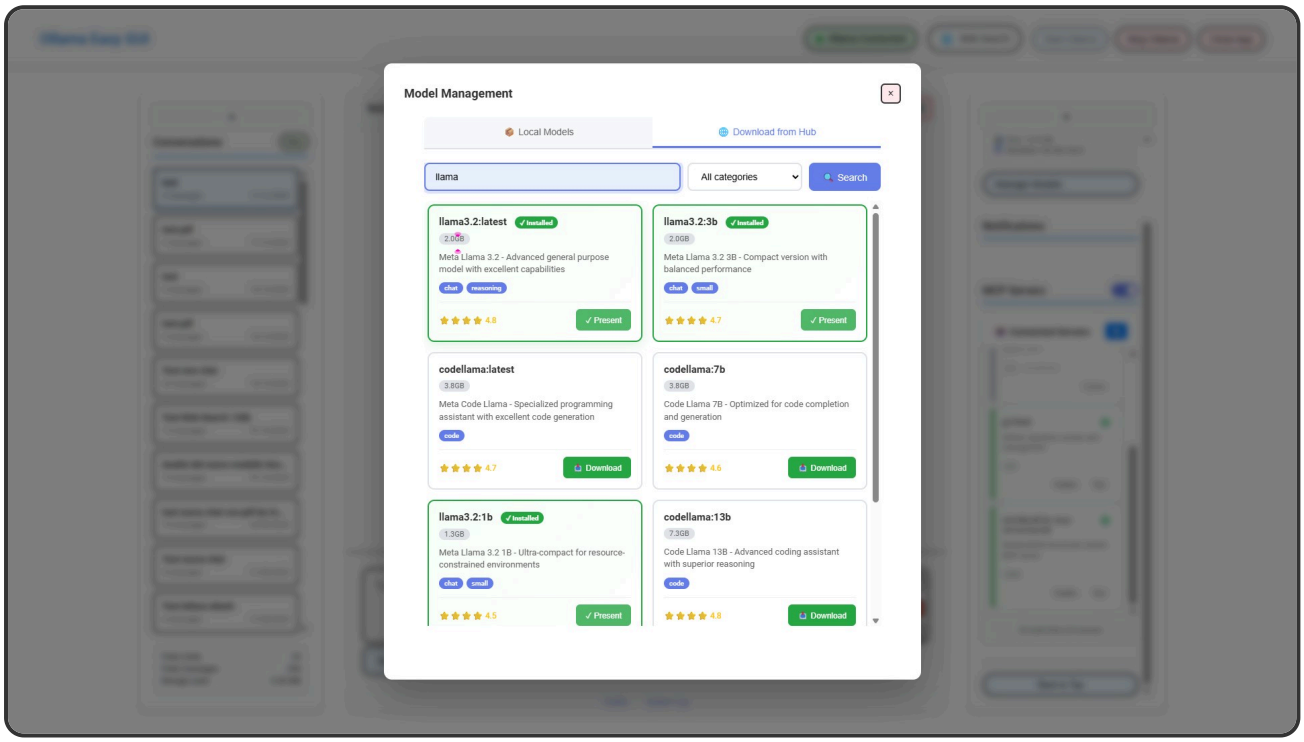
- **Name and version**: for example "llama3.2:3b"
- **Size**: how much disk space it takes

- **Date**: when it was downloaded
- **Category**: Chat, Code, Reasoning, Multimodal

You can sort the list by name, size, date or category using the buttons at the top.

Downloading new models

Switch to the **Hub Search** tab to search for models in the Ollama catalog:

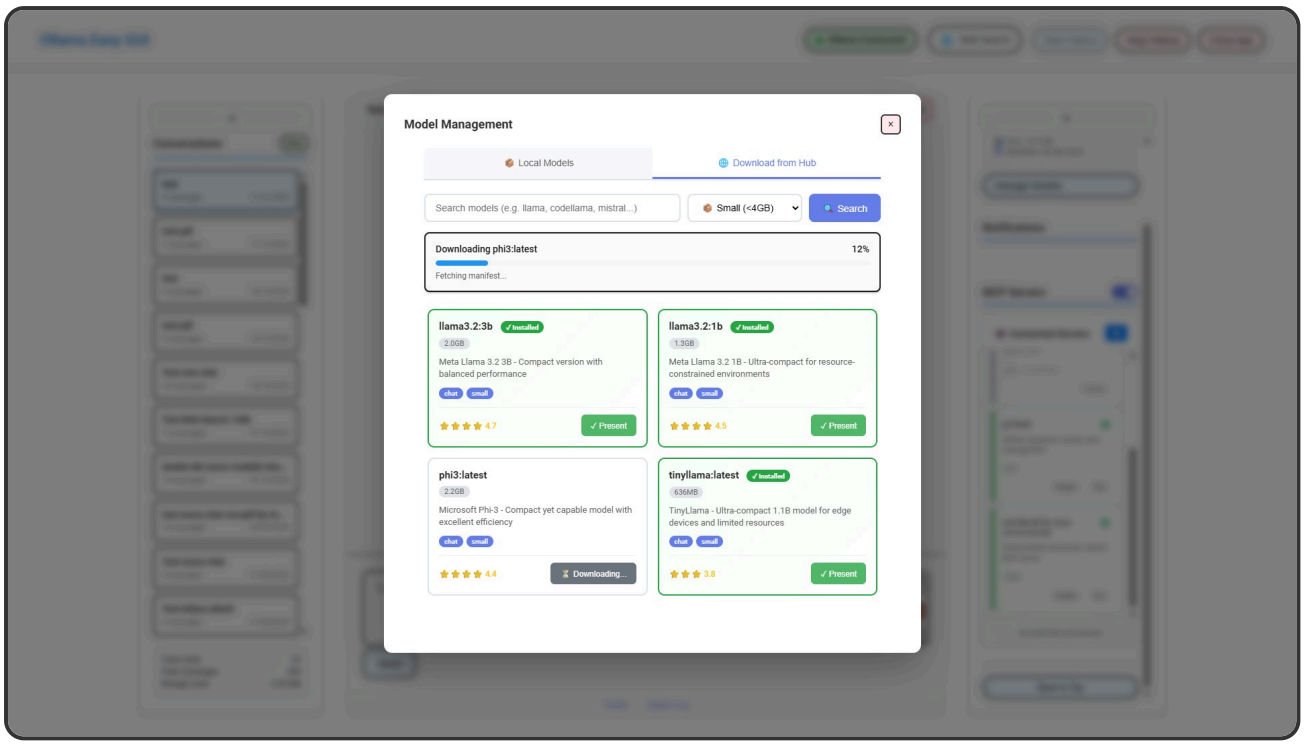


(https://docs.ai-know.pro/ollama-easy-gui-en/img/hub-modelli.jpg)

You can filter by category:

- **Chat**: generic models for conversation
- **Code**: specialized in programming
- **Reasoning**: optimized for logical reasoning
- **Multimodal**: capable of analyzing images too

Find an interesting model and click **Download**. You'll see the download progress in real time:



(https://docs.ai-know.pro/ollama-easy-gui-en/img/download-modello.jpg)

> **Before downloading**
>
> Models can be very large: a 70 billion parameter model requires over 40 GB of disk space.

Also, large models require more RAM and a GPU with enough memory to run at acceptable speed. See the "Which model to choose" section to understand what your hardware can handle.

Understanding model names

Model names follow a precise pattern:

```
name:variant
```

For example: `llama3.2:3b`, `qwen2.5:7b-instruct`, `codellama:13b`

The number after the colon typically indicates size: - **1b-3b**: lightweight models, fast, suitable for less powerful computers - **7b-8b**: good compromise between quality and speed - **13b-14b**: more accurate responses, need 16 GB of RAM - **70b and above**: maximum quality, require powerful hardware

Which model to choose

The choice depends on your hardware and what you want to do:

For computers with 8 GB of RAM

Model	Recommended use
llama3.2:3b	General conversation, fast
qwen2.5:3b	Good for texts in multiple languages
phi3:3.8b	Reasoning and logic

For computers with 16 GB of RAM

Model	Recommended use
llama3.1:8b	General use, excellent responses
qwen2.5:7b	Multilingual, including Italian
mistral:7b	Fast and reliable
codellama:7b	Programming

For computers with GPU (8+ GB VRAM)

Model	Recommended use
llama3.3:70b	Maximum quality
qwen2.5-coder:32b	Advanced programming
command-r:35b	Research and document analysis

Start small

If you don't know which to choose, start with llama3.2:3b to test that everything works, then move to larger models if your hardware allows.

Removing a model

To free up disk space, you can remove models you no longer use. In the local models tab, click the trash icon next to the model to delete.

> **Re-downloading is always possible**
>
> Removing a model isn't irreversible, you can always re-download it from the Hub if you need it.

Models and MCP

Not all models support MCP (Model Context Protocol), the feature that allows AI to use external tools. If you plan to use MCP, choose models that support "function calling":

- llama3.1, llama3.2, llama3.3
- qwen2.5 (all variants)
- mistral, mistral-nemo
- command-r, command-r-plus

Older models like llama2 or codellama don't support MCP. We'll explore this topic further in the dedicated chapter.

Customizing responses

AI models respond based on their default characteristics, but you can modify their behavior using two tools: the base prompt (for a single model) and the global system prompt (for all models).

What is a system prompt

A system prompt is an instruction that is sent to the model before each of your questions. The model considers it as a permanent context that guides its responses.

For example, if you set as system prompt:

```
Always respond in English, even if the question is in another language.
Use a professional but accessible tone.
Limit responses to a maximum of 200 words.
```

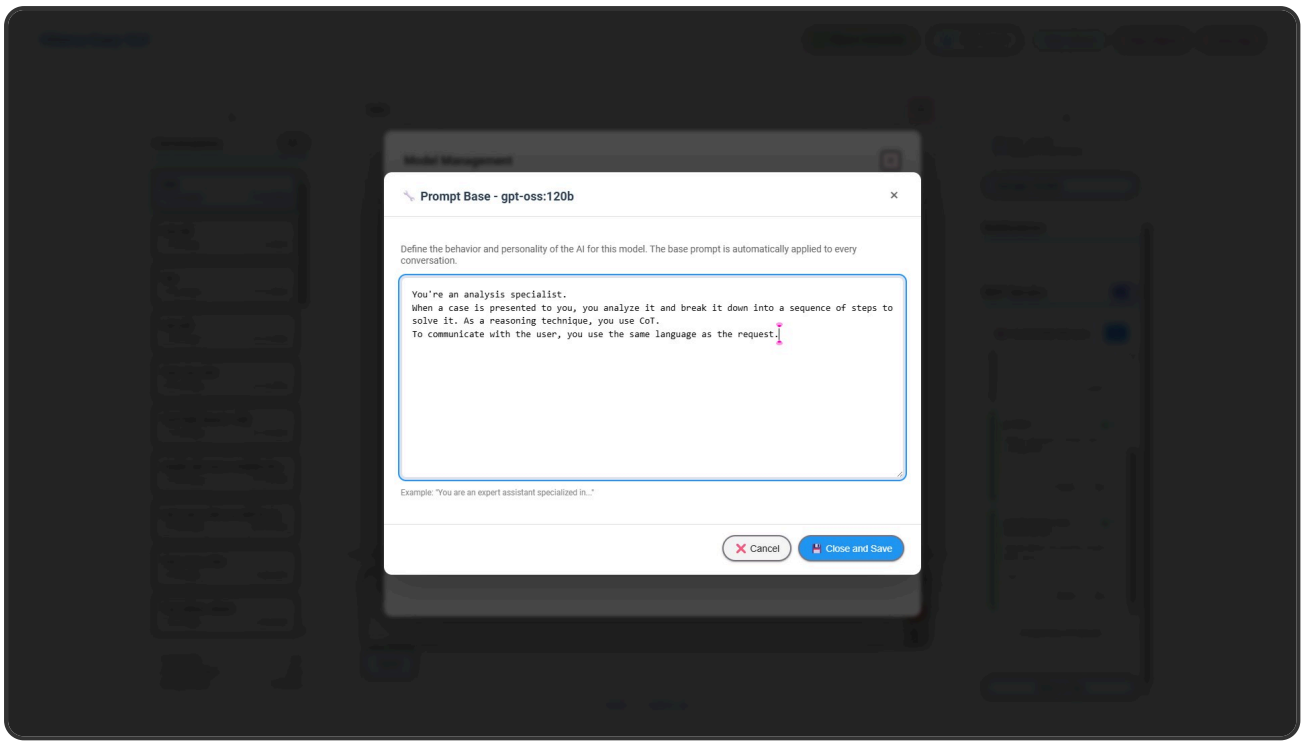
The model will follow these instructions in every conversation.

Base prompt: personality per model

The base prompt is specific to each model. It allows you to give a different "personality" to each one.

To set it:

1. Open model management (**Manage Models**)
2. Find the model you want to customize
3. Click the prompt icon (the icon with two T's)
4. The base prompt input window opens
5. Once you've entered the text, to save and exit you need to click **Close and Save**, if instead you don't want to save the changes just click the **Cancel** button
6. To remove a custom base prompt you need to open the base prompt (steps 1-2-3-4), delete the base prompt text and exit by saving



(https://docs.ai-know.pro/ollama-easy-gui-en/img/base-prompt.jpg)

Examples of useful base prompts

For a writing assistant:

```
You are an expert editor. When asked to review text:
- Correct grammatical and punctuation errors
- Suggest stylistic improvements
- Maintain the author's original tone
- Briefly explain the proposed changes
```

For a programming tutor:

```
You are a patient programming tutor. When explaining code:
- Use simple and concrete examples
- Explain each step, don't assume anything
- If the user makes mistakes, correct them gently
- Always suggest best practices
```

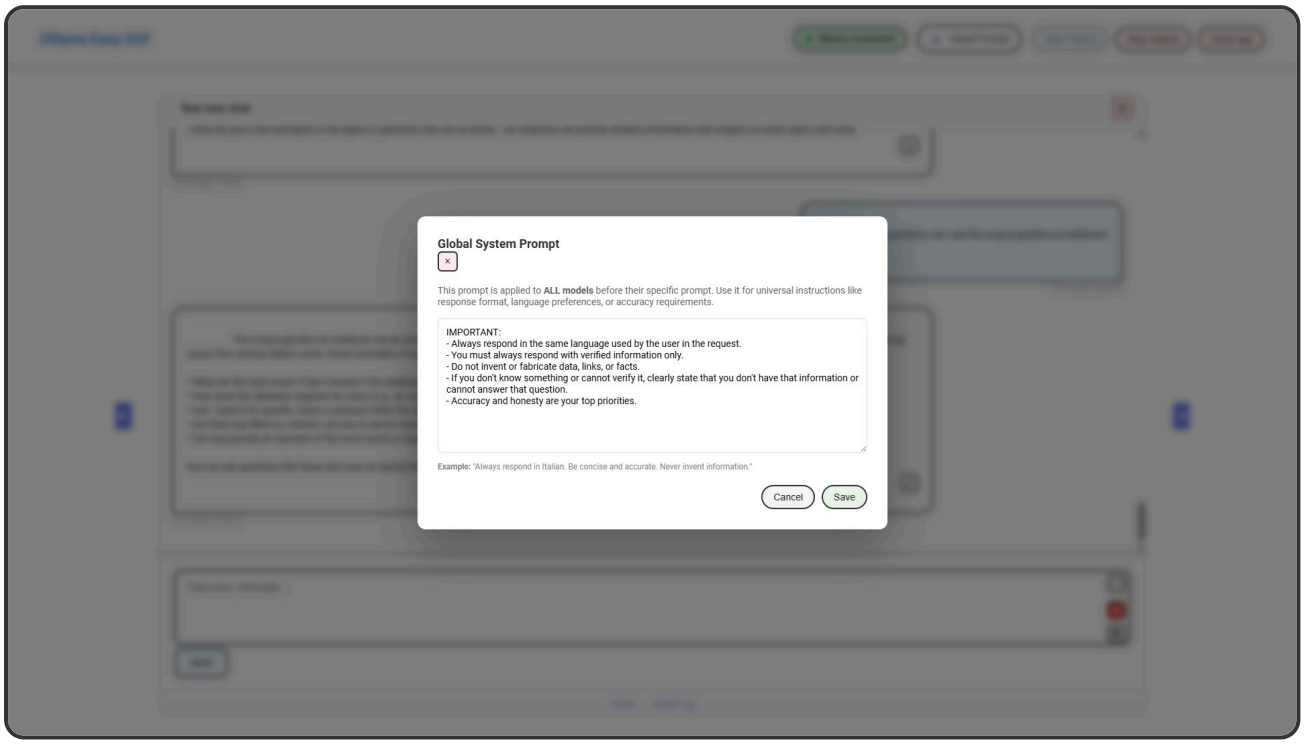
For document analysis:

```
When analyzing documents:
- Extract key points in list form
- Identify any contradictions or gaps
- Maintain an objective approach
- Always cite relevant parts of the document
```

Global system prompt: universal instructions

The global system prompt applies to all models, before their specific base prompt. It's useful for setting general preferences you want to always maintain.

To access it, click the **Global Prompt** button in the application's top bar:



(https://docs.ai-know.pro/ollama-easy-gui-en/img/global-prompt.jpg)

When to use the global prompt

The global prompt is ideal for:

- **Language preferences**: "Always respond in English"
- **Response format**: "Use bullet points when possible"
- **Accuracy rules**: "If you're not sure about something, say so clearly"
- **Communication style**: "Avoid long introductions, get straight to the point"

Practical example

An effective global prompt could be:

```
General rules:
1. Respond in the same language as the question
2. Be concise: prefer short but complete answers
3. If you don't know something, admit it instead of making it up
4. When providing instructions, number them in clear steps

Format:
- Use bold for important terms
- Use lists for more than 3 items
- Include examples when they help understanding
```

How they work together

When you send a message, the model receives instructions in this order:

1. **Global system prompt** (if set)
2. **Model base prompt** (if set)
3. **Your message**

The model considers all this context to formulate the response. If instructions conflict, the ones applied last (the base prompt) tend to prevail.

Best practices

To get better results:

- **Be specific**: vague instructions produce vague results
- **Test changes**: after changing a prompt, ask some test questions
- **Don't overdo it**: prompts that are too long can confuse the model
- **Iterate**: refine prompts over time based on results

Attachments and export

Ollama Easy GUI allows you to attach documents to conversations and export responses in various formats. Let's see how to use these features.

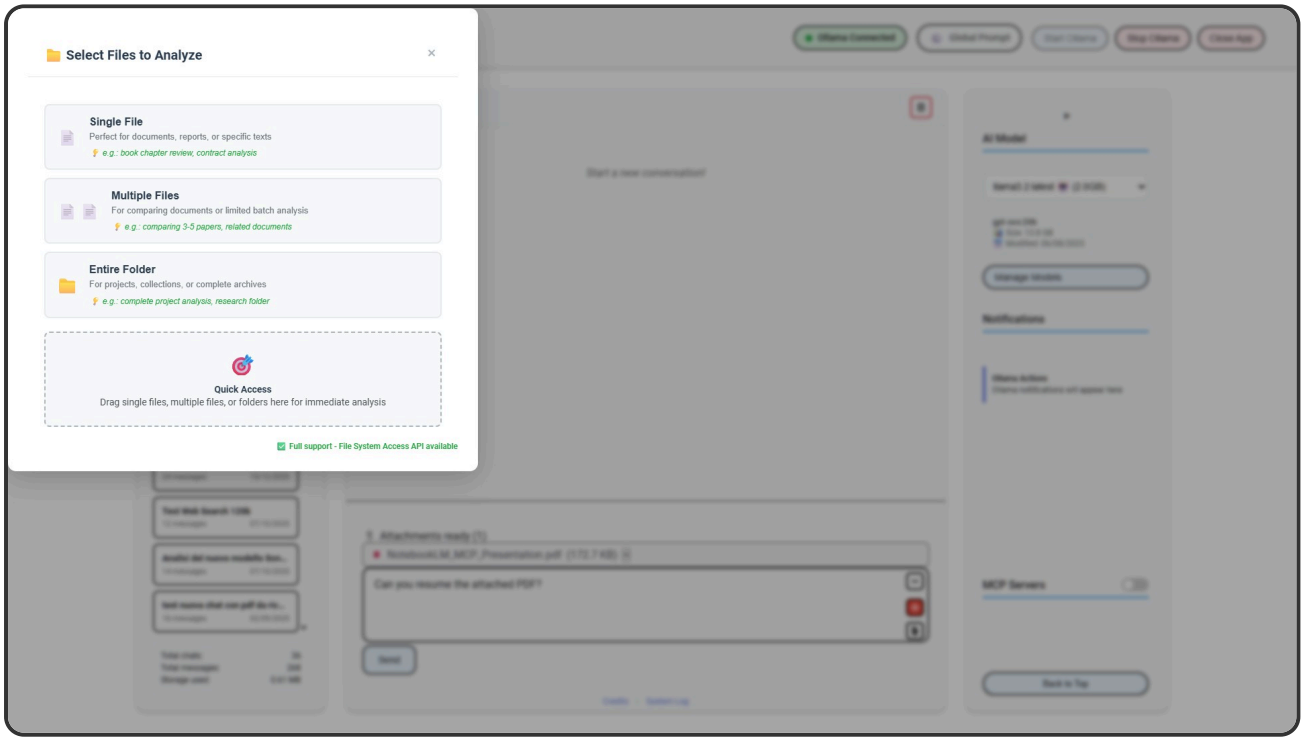
Attaching files

You can attach files to have them analyzed by the model. Supported formats are:

- **Documents**: PDF, Word (.docx), text files (.txt, .md)
- **Images**: JPG, PNG, GIF, WebP
- **Code**: any plain text file

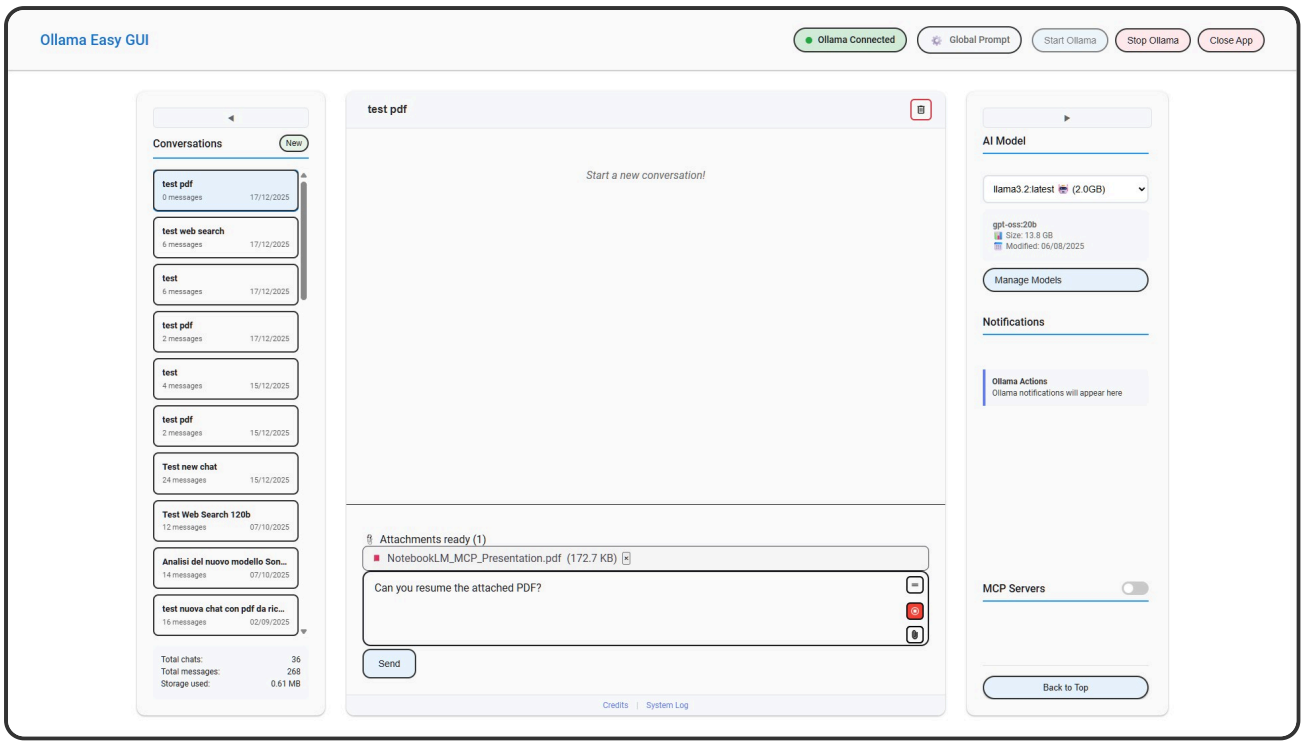
How to attach a file

Click the paperclip icon in the input area or drag files directly into the window:



(https://docs.ai-know.pro/ollama-easy-gui-en/img/allega-file.jpg)

You can select one or more files, or an entire folder. Selected files will appear below the text area:



(https://docs.ai-know.pro/ollama-easy-gui-en/img/file-allegato.jpg)

Write your question and send. The model will receive both your message and the content of the attached files.

Usage examples

Analyzing a document:

> Attach a contract PDF and ask: "Summarize the main clauses of this contract and flag any critical points"

Code review:

> Attach a code file and ask: "Check this Python code and suggest improvements"

Describing an image:

> Attach an image and ask: "Describe what you see in this image" (requires a multimodal model)

Language behavior

When you attach documents, the model tends to respond in the document's language, even if your question is in another language. To force a specific language, indicate it explicitly in the question or in the system prompt.

Attachment limits

There are some limitations to consider:

- **Maximum size**: 50 MB per file
- **Images**: require multimodal models (like llava or bakllava)
- **Complex PDFs**: tables and elaborate layouts might not be interpreted perfectly

For very long documents, consider breaking up the analysis: first ask for a general summary, then dig into specific sections.

Exporting conversations

You can save individual messages or entire conversations in different formats.

Exporting a single message

Hover over a model response and click the download icon. Choose the format:

- **Markdown** (.md): preserves formatting, ideal for documentation
- **Text** (.txt): universal format, readable anywhere
- **Word** (.docx): for formal documents or sharing

Exporting an entire conversation

To export the whole conversation, use the conversation menu in the left sidebar. You'll find the same format options.

Where data is saved

All your data stays on the local computer:

Data type	Location
Conversations	`app/data/conversations/`
Configurations	`app/data/`
Exported files	Browser download folder

Conversation files are simple readable JSON. You can back them up by copying the entire `app/data` folder.

Export formats compared

Format	Pros	Cons
Markdown	Preserves formatting, highlighted code, headings	Requires an editor that supports it
Text	Universally readable, lightweight	Loses formatting
Word	Easy to share, editable	Larger file

For technical documentation

If you export conversations containing code, Markdown format is the best choice: it preserves syntax highlighting and code block formatting.

MCP: extending AI

MCP (Model Context Protocol) is an advanced feature that allows AI models to use external tools, like reading files from your computer or accessing online services. In this chapter we explain what it is, when to use it, and how to configure it.

What is MCP in simple terms

Normally, an AI model can only read what you write and respond. It can't "do" anything in the real world: it can't open files, can't search the internet, can't check what's in a folder.

MCP changes this: it allows you to connect "tools" to the model that it can use during the conversation. For example:

- **Filesystem tool**: the model can read and write files on your computer in enabled folders
- **GitHub tool**: the model can search for information in repositories
- **Custom tools**: you can add others depending on your needs, to connect to applications on your PC or to connect to online services

When MCP is active and you ask the model "**Using filesystem read the file report.txt in folder X**", the model:

1. If it's active, understands it needs to use the filesystem tool
2. Calls the tool with the file path
3. Receives the file content
4. Uses it to answer your question

All this happens automatically, without you having to do anything different.

Compatible models

Not all models support MCP. It requires a capability called "function calling" that allows the model to generate structured calls to tools.

Models that work with MCP

Model	MCP quality	Notes
llama3.1, llama3.2, llama3.3	Excellent	Native support
qwen2.5 (all variants)	Excellent	Also great for Italian
mistral, mistral-nemo	Good	Fast and reliable
command-r, command-r-plus	Excellent	Great for research

Models that DON'T work with MCP

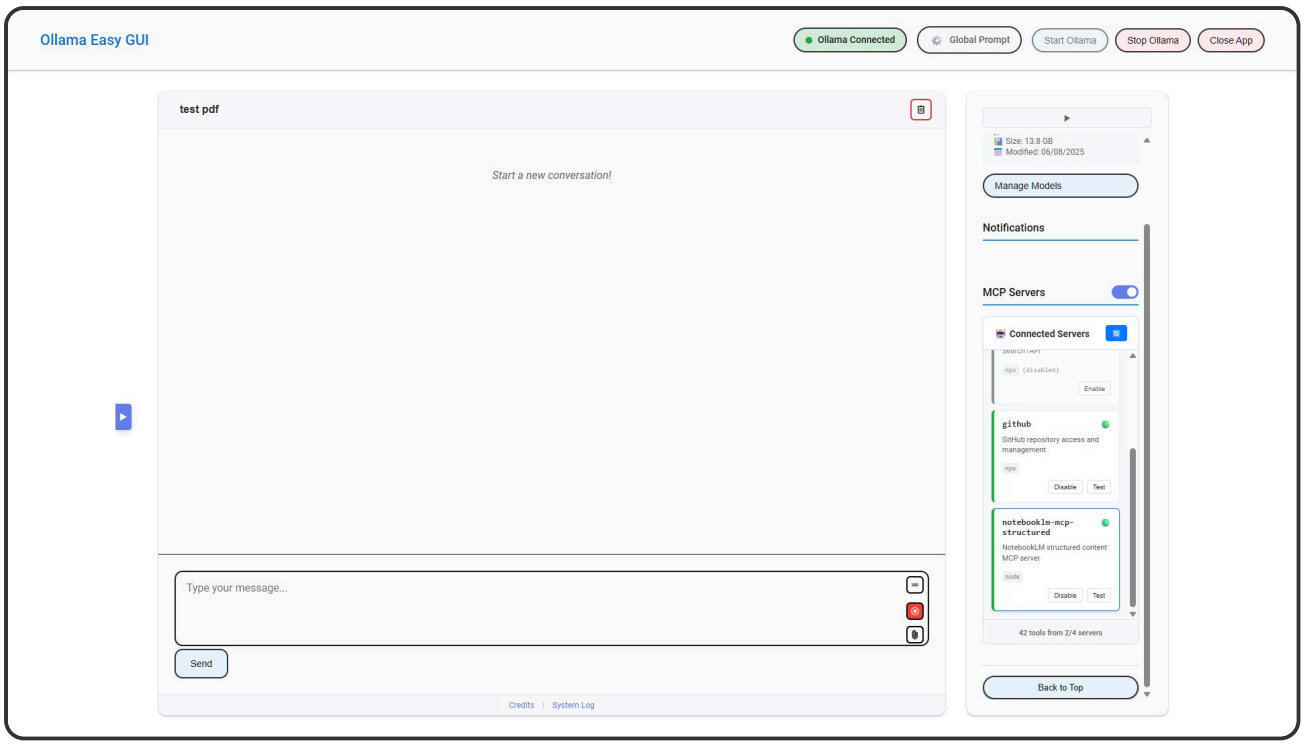
- llama2 (all versions)
- codellama
- phi, phi2
- gemma (version 1)

If you use a model that's not compatible with MCP enabled, it will simply ignore the available tools.

Activating MCP

MCP is disabled by default. To activate it:

1. In the right sidebar, find the **MCP** switch
2. Turn it on (it turns blue)
3. Click the MCP settings icon to see available tools



(https://docs.ai-know.pro/ollama-easy-gui-en/img/mcp-pannello.jpg)

Configured tools appear in the list. Activate the ones you want to make available to the model.

Configuring tools

MCP tools configuration is done in the `app/data/mcp-config.json` file. The first time you need to create it by copying the example:

```
copy app\data\mcp-config.json.example app\data\mcp-config.json
```

Then edit it with a text editor:

```
{
  "mcpServers": {
    "filesystem": {
      "command": "npx",
      "args": ["-y", "@modelcontextprotocol/server-filesystem", "D:/Documents"],
      "enabled": false,
      "description": "File access"
    },
    "github": {
      "command": "npx",
      "args": ["-y", "@modelcontextprotocol/server-github"],
      "env": { "GITHUB_TOKEN": "your-token-here" },
      "enabled": false,
      "description": "GitHub access"
    }
  }
}
```

Filesystem tool

Allows the model to read and write files. You must specify which folders it can access:

```
"args": ["-y", "@modelcontextprotocol/server-filesystem", "D:/Documents", "D:/Projects"]
```

> **Security**
>
> Limit access only to necessary folders. Don't give access to the entire disk.

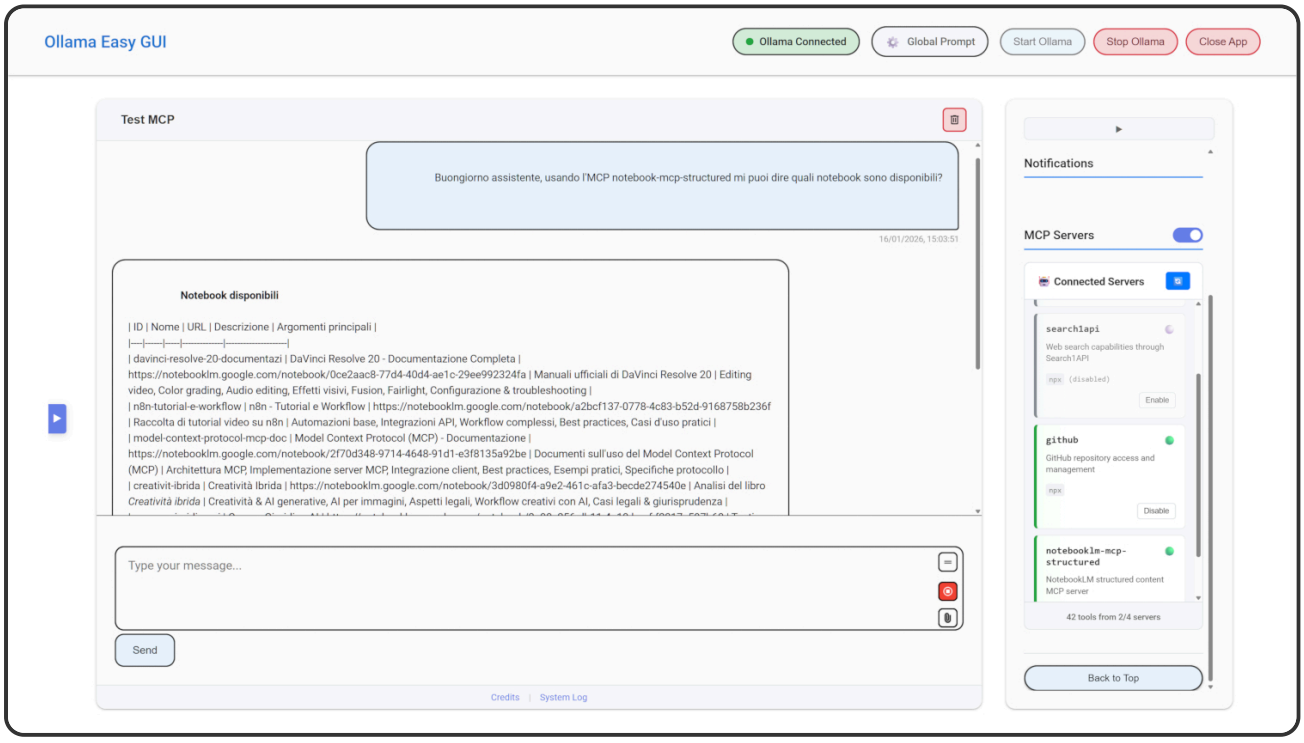
GitHub tool

Allows the model to search in GitHub repositories. Requires a personal access token:

1. Go to GitHub → Settings → Developer settings → Personal access tokens
2. Create a new token with necessary permissions
3. Insert it in the configuration file

MCP in action

Once MCP servers are activated you can make requests, preferably mentioning the MCP server you want to use



(https://docs.ai-know.pro/ollama-easy-gui-en/img/mcp-query.jpg)

Repositories

There's starting to be a wide availability of MCP servers and repositories. One of these is github.com/modelcontextprotocol/servers (https://github.com/modelcontextprotocol/servers)

Privacy and MCP

MCP servers can also work locally on your computer, however even in these cases some MCP/tools might connect to external services:

- **filesystem**: completely local, no data leaves
- **github**: connects to GitHub servers

Before using an MCP/tool, consider what data it might transmit.

MCP troubleshooting

"The model doesn't use the tools" - Verify that MCP is active (blue switch) - Verify that the specific tool is activated in the panel - Try a different model (it might not support function calling)

"Error during tool execution" - Check the application logs for details - Verify that paths in the configuration file are correct - For GitHub, verify that the token is valid

"The tool is slow" - This is normal for operations on many files - Smaller models might take longer to generate correct calls

Troubleshooting

This section collects the most common problems and their solutions. If you can't find an answer here, you can ask for help in the community channels.

Installation problems

"npm" is not recognized as a command

Cause: Node.js is not installed or the terminal wasn't restarted after installation.

Solution: 1. Verify that Node.js is installed: search for "Node.js" in the Start menu 2. Close and reopen Command Prompt 3. If the problem persists, reinstall Node.js

"git" is not recognized as a command

Cause: Git is not installed or is not in the system PATH.

Solution: 1. Reinstall Git from git-scm.com (https://git-scm.com) 2. During installation, make sure the "Git from the command line" option is selected 3. Restart the computer

Error during npm install

Cause: network or permission problems.

Solution: 1. Try running Command Prompt as Administrator 2. If you're behind a corporate proxy, configure npm:

```
npm config set proxy http://proxy.company.com:8080
```

3. Try clearing the cache and retrying:

```
npm cache clean --force
npm install
```

Startup problems

The application won't start

Possible cause 1: Ollama is not running.

Check: look for the Ollama icon in the notification area (near the clock). If it's not there: 1. Search for "Ollama" in the Start menu and launch it 2. Wait a few seconds for it to fully start 3. Try launching Ollama Easy GUI again

Possible cause 2: port 3003 is already in use.

Solution: close other instances of the application.

Blank page in browser

Cause: the server is started but there's a frontend error.

Solution: 1. Open browser developer tools (F12) 2. Check the "Console" tab for error messages 3. Try clearing the browser cache and reloading

Model problems

No models available in the list

Cause: Ollama has no models installed or is not reachable.

Solution: 1. Verify that Ollama is running 2. Download at least one model:

```
ollama pull llama3.2
```

3. Reload the web interface

The model responds very slowly

Cause: insufficient hardware resources or model too large.

Solutions: - Try a smaller model (e.g. switch from 8b to 3b) - Close other applications to free up RAM - If you have a GPU, verify that Ollama is using it - Increase `OLLAMA_NUM_THREADS` in the `.bat` file

The model doesn't understand my language

Cause: some models are trained primarily on English texts.

Solutions: - Use multilingual models like `qwen2.5` or `mistral` - Add in the system prompt: "Always respond in [your language]" - Consider that small models (<3b) may have limited capabilities in languages other than English

Attachment problems

The PDF is not read correctly

Cause: PDF with complex formatting or scanned.

Solutions: - If the PDF is a scan, the text is not extractable: use OCR first - For PDFs with many tables, consider extracting the text manually - Try attaching a simpler version of the document

Images are not analyzed

Cause: the model is not multimodal.

Solution: use a model that supports images: - `llava` - `bakllava` - `llama3.2-vision`

MCP problems

MCP tools don't work

Checklist: 1. Is MCP activated? (switch in the sidebar) 2. Is the specific tool activated in the MCP panel? 3. Does the model support function calling? (use llama3.1, qwen2.5, mistral) 4. Does the mcp-config.json file exist and is it configured correctly?

"Tool not found" error

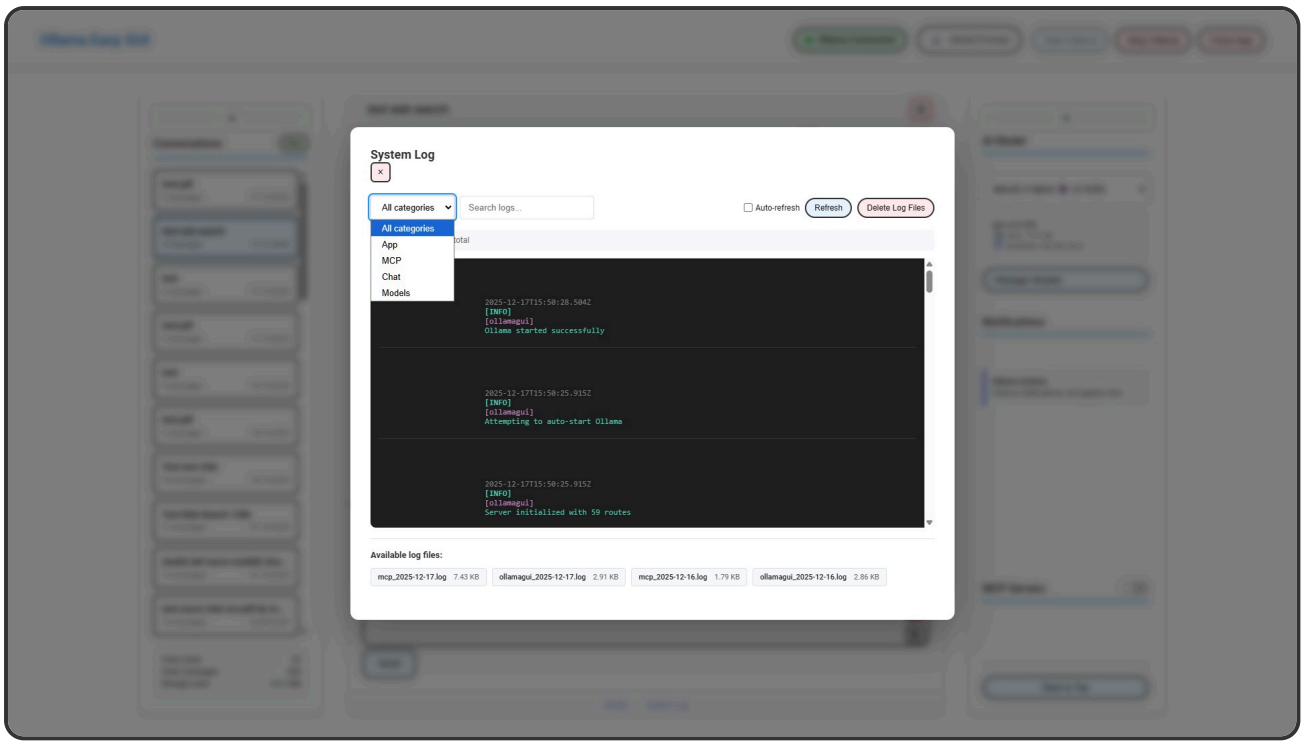
Cause: the MCP server didn't start correctly.

Solution: 1. Verify that Node.js is installed (MCP servers require it) 2. Check logs for specific errors 3. Try restarting the application

Checking the logs

The application keeps track of errors in log files. To view them:

1. Click the **Log** button in the interface footer
2. Select the category to view:
3. **App**: general application errors
4. **Chat**: problems during conversations
5. **MCP**: external tool errors
6. **Models**: problems with download or loading models



(https://docs.ai-know.pro/ollama-easy-gui-en/img/log-viewer.jpg)

You can search for specific text and filter by date.

Getting support

If the problem persists:

- **GitHub Issues**: github.com/paolodalprato/ollama-easy-gui/issues (https://github.com/paolodalprato/ollama-easy-gui/issues) to report bugs or ask for help
- **GitHub Discussions**: for general questions and discussions

When asking for help, include: - Description of the problem - Operating system and version - Error messages (from logs or console) - Steps to reproduce the problem

Notes for macOS and Linux

This manual was written for Windows, but Ollama Easy GUI should also work on macOS and Linux. Here are the main differences.

macOS

Installing prerequisites

Ollama: Download from ollama.ai (https://ollama.ai), the procedure is similar to Windows.

Git: Usually already installed. Verify with:

```
git --version
```

If not present, install Xcode Command Line Tools:

```
xcode-select --install
```

Node.js: Download from nodejs.org (https://nodejs.org) or install via Homebrew:

```
brew install node
```

Differences in operation

- Use **Terminal** instead of Command Prompt
- Paths use `/` instead of `\`
- The application folder could be `~/Applications/ollama-easy-gui`

Starting the application

```
cd ~/Applications/ollama-easy-gui
npm start
```

Linux

Installing prerequisites

Ollama: Follow instructions on ollama.ai (https://ollama.ai) for your distribution.

Git: Install via package manager:

```
# Ubuntu/Debian
sudo apt install git

# Fedora
sudo dnf install git

# Arch
sudo pacman -S git
```

Node.js: Install via package manager or use nvm:

```
# Ubuntu/Debian
sudo apt install nodejs npm

# Or with nvm (recommended)
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.0/install.sh | bash
nvm install --lts
```

Differences in operation

- Use the **terminal** of your distribution
- Paths use `/`
- Permissions might require `sudo` for some operations

Starting the application

```
cd ~/ollama-easy-gui
npm start
```

Common differences

Aspect	Windows	macOS/Linux
Terminal	Command Prompt or PowerShell	Terminal
Paths	`C:\folder\file`	`/folder/file`
Home user	`%USERPROFILE%`	`~` or `$HOME`
Running in background	Icon in notification area	Background process

Known problems

Ollama not reachable

On some Linux configurations, Ollama might listen only on localhost. Verify with:

```
curl http://localhost:11434/api/tags
```

Permission errors

If you get permission errors, make sure your user has access to the installation folder:

```
sudo chown -R $USER:$USER ~/ollama-easy-gui
```

Not tested

The following features have not yet been tested on macOS and Linux:

- Auto-start of Ollama
- .bat file (Windows-specific, would need a .sh version for Unix)
- Some MCP server paths

If you encounter problems, open an issue on GitHub reporting your operating system and the error found.