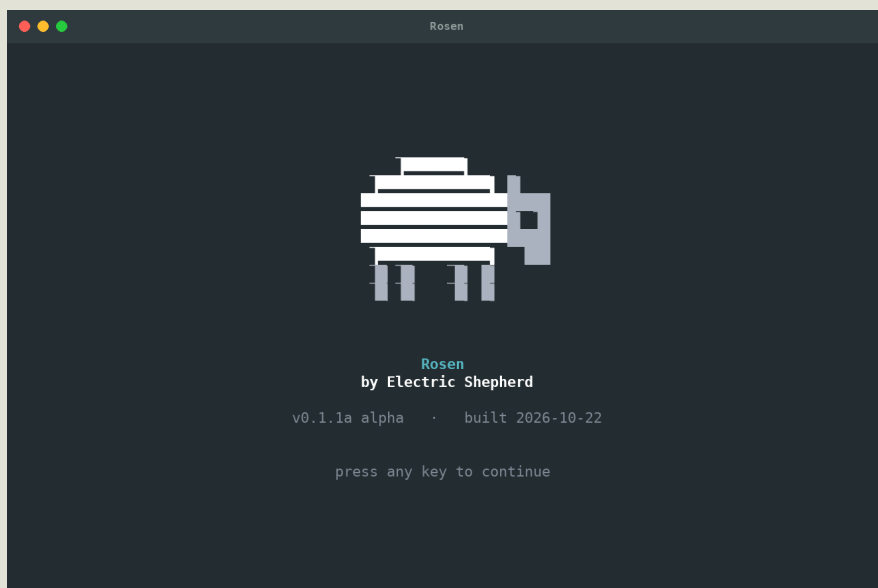


ELECTRIC SHEPHERD

Rosen User Manual

Version 0.1.1a (alpha) · Built 2026-10-22



Electric Shepherd

A chat client for your language models, in the terminal · Linux and macOS

Contents

1. Welcome to Rosen	5
1.1 What Rosen does	5
1.2 What you need	6
1.3 Installing and starting	6
1.4 The splash screen	7
1.5 About the version number	8
1.6 Conventions in this manual	8
1.7 The terms of use	8
2. A Tour of the Screen	9
2.1 The pages	9
2.2 The sidebar layout	9
2.3 The top-bar layout	10
2.4 Lists, views and columns	11
2.5 Tabs	12
2.6 Dialogs	13
2.7 Help	14
2.8 Keys that work everywhere	15
3. The Mouse	17
3.1 What a click does	17
3.2 The wheel	18
3.3 Turning the mouse off	18
4. Quick Start	20
4.1 1. Start Rosen	20
4.2 2. Tell Rosen where your server is	20
4.3 3. Choose a model	22
4.4 4. Say something	22
4.5 5. What next	23
5. Chat	24
5.1 Writing a message	24
5.2 Reading replies	25
5.3 When something goes wrong	26
5.4 Asking again	27
5.5 Several chats	27
5.6 This chat's server and model	28
5.7 What the model is told	28
5.8 Keys on the Chat page	29
5.9 Commands in a message	29
6. Sessions	31
6.1 The table	31
6.2 Views	32
6.3 Keys	32
6.4 Searching, filtering and sorting	33
6.5 Reading, editing and cloning	35
6.6 Deleting	36
6.7 Chats that cannot be read	37
7. Servers	38
7.1 The table	38
7.2 Adding and editing a server	39
7.3 Cloning and looking	39
7.4 Testing	40
7.5 Choosing a server	40
7.6 Keys	40

7.7 Servers on https://	41
7.8 Moving servers between computers	41
8. Models	42
8.1 The list	42
8.2 Choosing	43
8.3 Typing a model name	44
8.4 Several models, several servers	44
9. Code and the Shell	45
9.1 Code blocks	45
9.2 The code menu	45
9.3 Running code	46
9.4 The shell	47
9.5 What is kept	48
10. Export, Import and Attach	49
10.1 The file browser	49
10.2 Exporting a chat	50
10.3 Exporting everything	53
10.4 Importing	53
10.5 Attaching files to a message	54
10.6 What is never done	55
11. Summary	56
11.1 What it counts	56
11.2 Keys	57
12. Checks	58
12.1 What it checks	58
12.2 Keys	59
12.3 The alert badge	59
13. Settings	60
13.1 On the screen	60
13.2 Sent to the model	63
13.3 The files at the foot of the page	64
14. Looks	65
14.1 The looks	66
14.2 The monochrome and classic looks	67
14.3 Reading them	69
14.4 Choosing a look	70
14.5 Color depth and terminals	72
15. Replies You Are Not Looking At	73
15.1 What counts as out of sight	73
15.2 What you see and hear	73
15.3 Choosing	74
15.4 Several chats at once	74
16. The Command Line	76
16.1 Synopsis	76
16.2 Options	76
16.3 Looking at a page as text	77
16.4 Environment	78
16.5 Exit status	79
17. Your Data	80
17.1 Where the files are	80
17.2 How your data is protected	81
17.3 If something goes wrong inside Rosen	81
17.4 Backing up and moving	82
17.5 What Rosen sends, and where	82

17.6 Coming from llm-tui	82
18. Trying Rosen Without a Server	83
18.1 What the sample data holds	83
18.2 The echo server	84
18.3 What does not work in demo mode	84
19. When Something Goes Wrong	85
19.1 A server does not answer	85
19.2 HTTPS and certificates	86
19.3 A reply goes wrong	86
19.4 The screen	86
19.5 Starting, files and locks	87
19.6 The terminal is in a mess	87
19.7 Reporting a problem	88
20. Limits and Known Issues	89
20.1 Limits	89
20.2 What Rosen does not do	89
20.3 Terminals	90
20.4 Known issues	90
Appendix A. Key Reference	91
A.1 On every page	91
A.2 The Chat page	92
A.3 Writing a message	92
A.4 The Sessions, Servers and Models tables	93
A.5 Summary, Checks and Settings	94
A.6 Forms, menus and questions	94
A.7 The file browser	94
A.8 Code blocks and the text viewer	95
A.9 The mouse	95
Appendix B. Coming from llm-tui	97
B.1 Bringing your data across	97
B.2 Keys that moved	97
B.3 Commands that changed	98
B.4 Command line and environment	98
B.5 What is better, and what you might notice	98
Appendix C. Standards and Conformance	100
C.1 Command line	100
C.2 What it asks the system	101
C.3 Extensions	101
C.4 What it does not do	102
C.5 How it is checked	102
Appendix D. Release Notes	103
D.1 v0.1.1a (alpha), 22 October 2026	103
D.2 v0.1a (alpha), 21 October 2026	104
Appendix E. Glossary	107

1. Welcome to Rosen

Rosen is a chat client for language models that lives in your terminal. You point it at a model server (Ollama, llama.cpp, LM Studio, vLLM or anything else that speaks the OpenAI chat API), type a message, and read the reply as it arrives. Every chat is saved as it goes, you can keep several going at once with different models, and when a model writes code you can read it, copy it, save it or run it without leaving the window.

It is one small program with no dependencies. It talks to your servers itself, keeps everything in plain files that you can read and back up, and never sends anything anywhere except to the servers you add.

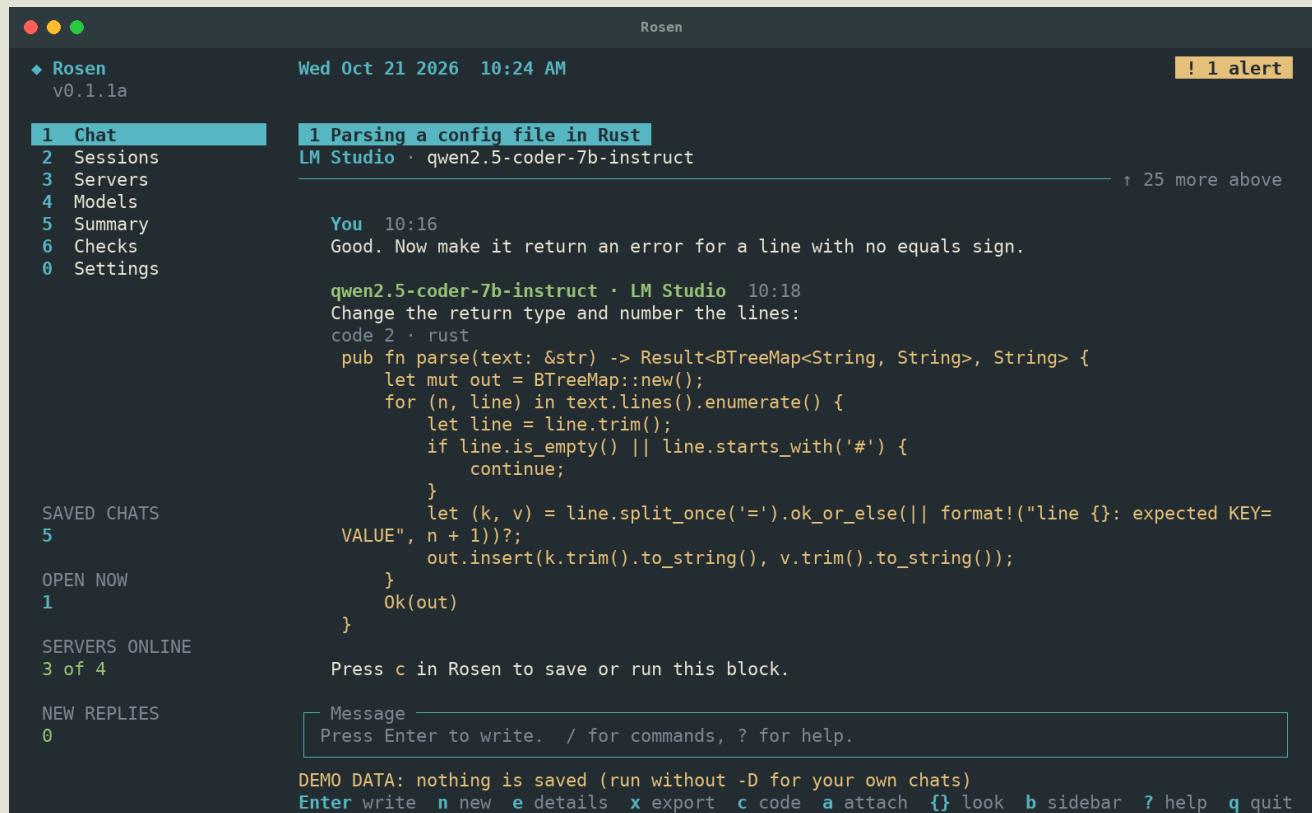


Figure 1. The Chat page, with the sample chats that rosen -D loads. The sidebar lists the six pages and keeps the counts that matter in view.

1.1 What Rosen does

- **Chat** with a model and watch the reply arrive word by word. Press **Esc** to stop one that is going the wrong way. Your message is never lost: if a reply fails or is cut short, what arrived is kept and the message is still there to send again.
- **Several chats at once.** Each chat is a tab with its own server and model, so you can ask a small fast model and a large slow one side by side. Up to nine chats can be open at a time (Chapter 5).
- **Several servers.** Add every program that runs a model on your computer or your network, test each one, and see which models it offers (Chapters 7 and 8).
- **Code you can use.** Code blocks in replies are found for you. Read one, copy it, save it to a file or run it with a time limit, and open a shell that already knows where the code is (Chapter 9).

- **Files in and out.** Export a chat as Markdown, plain text, HTML or a chat file; import a saved chat; attach text files to a message or load a Markdown or text file as the start of a new chat. A built-in file browser does the choosing (Chapter 10).
- **A heads-up when a reply arrives.** If a model finishes while you are on another page, in another chat or in another window, the tab is marked and the terminal rings its bell (Chapter 15).
- **Looks.** Fifteen color looks, from full color and quiet dark palettes to green or amber on black, stepped through with `{` and `}`; a start-up screen with a small flock of sheep, and a clock if you want one (Chapters 13 and 14).
- **The keys and the mouse of Inventory Central.** The same page keys, the same list keys, the same forms with `[ Save ]` and `[ Cancel ]`, and clicks and the wheel where you want them (Chapter 3).
- **Help that knows where you are.** Press `?` on any page or in any dialog (Chapter 2).

Note. Rosen is an alpha release. It has been tested against its own built-in echo server and against test servers that behave like Ollama and the OpenAI API; it has not been tried against every server on the market. Keep a copy of your data (section 17.1) until you are comfortable with it.

1.2 What you need

- **Linux or macOS** and any terminal that understands ANSI escape codes (practically every modern terminal does). Native Windows is not supported; use WSL.
- A terminal at least **50 columns by 12 rows**. At **100 columns by 24 rows** or more you get the sidebar layout; smaller windows use a top bar (Chapter 2).
- A **Rust toolchain** (1.97 or newer) to build it from source. Rosen uses only the Rust standard library, so building works offline.
- The standard Unix commands `stty` (Rosen uses it to control the terminal), `date` (for your local time), `sh`, `kill` and `ps` (to stop a code block that ran out of time and to tell whether another copy is running). Every Linux and macOS system has them.
- For `https://` servers, the **curl** program (section 7.7). Servers on `http://` addresses need nothing extra.
- A **model server** to talk to. If you have none yet, Rosen still starts and has a built-in echo server for finding your way around (Chapter 18).

1.3 Installing and starting

Rosen ships as a source tarball and a test build for Linux. To build it yourself:

```
tar -xzf rosen-0.1.1a.tar.gz
cd rosen-0.1.1a
cargo build --release          # no network access needed (no dependencies)
./target/release/rosen -D      # look around with sample data (nothing is saved)
./target/release/rosen         # your own chats
```

The program is called `rosen`. To run it from anywhere, copy `target/release/rosen` to a folder on your `PATH`, for example `~/local/bin`. The first time you run it without `-D` it starts with two servers already

listed (Ollama on this computer and an OpenAI-compatible server on port 8080), ready for you to edit (Chapter 4 walks through it).

Tip. Run `rosen -D` first. It loads five sample chats, four sample servers and a built-in echo server that answers every message, so you can click through every page. Nothing you do in demo mode is saved, and a reminder is shown on the message line.

1.4 The splash screen

Every time Rosen starts it shows a splash screen: a small flock of sheep, the words "Rosen by Electric Shepherd", the version and the build date. It stays until you press a key.

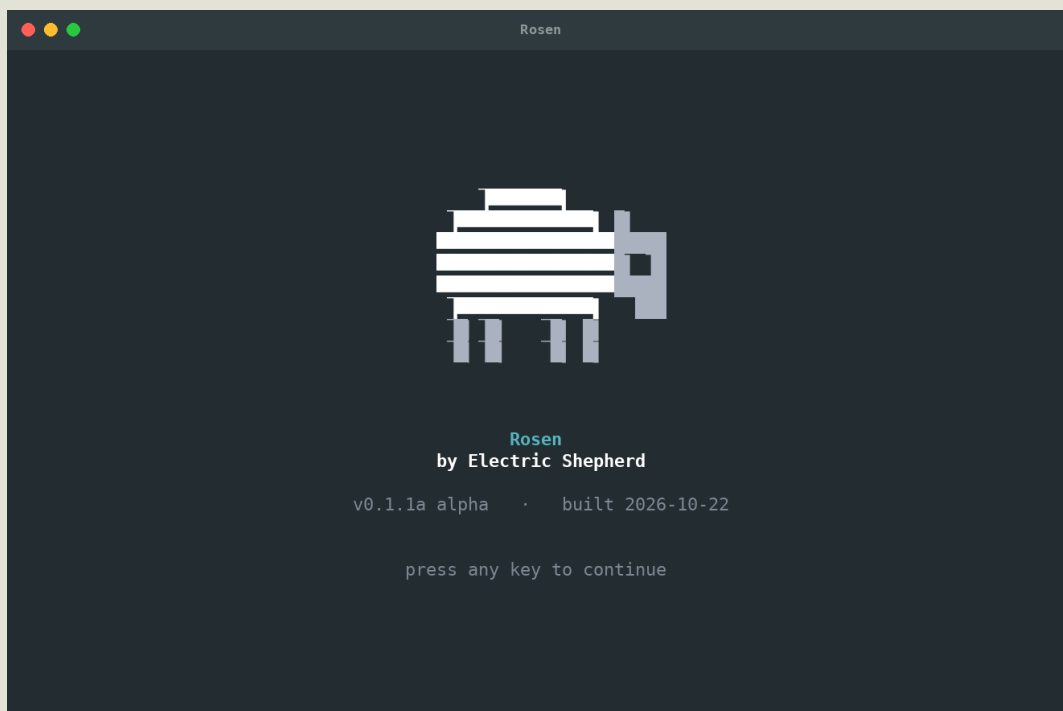


Figure 2. The start-up splash. The version (v0.1.1a, an alpha) and the date the program was built are shown below the name.

To start without it, use `-n` (or `--no-splash`), or set `ROSEN_NO_SPLASH=1`, or choose **Off** (or the plain **Classic** title box) for the **Splash screen** setting (section 13.1). `rosen -v` prints the same name and version on one line:

```
$ rosen -v
Rosen v0.1.1a (alpha, built 2026-10-22)
```

`rosen -L` prints the terms of use (the licence), whose first line names the product, the version and the company (section 1.7).

1.5 About the version number

The short version reads `v0.1.1a`: major version 0, minor version 1, patch 1, and the letter **a** for alpha (**b** will mean beta). Every iteration counts up by one, from `v0.1a` through `v0.1.1a`, and so on to `v0.1.16a`, after which the minor version rolls over to `v0.2.0a`. A final release has no letter (`v1.0`). The Cargo version, `0.1.1-alpha`, says the same thing. The build date is stamped in when the program is compiled, in UTC, and honours `SOURCE_DATE_EPOCH` so that the same source always builds the same program.

1.6 Conventions in this manual

- Keys to press are shown like `a` , `Enter` , `Esc` or `Ctrl-S` . Capital letters mean Shift, so `N` is different from `n` . `Alt-1` means hold Alt and press 1.
- Text in this typeface is something you type, a file name, or a command.
- Screenshots are real screens of the program, drawn from its own screen renderer with the sample data that `rosen -D` loads; the date in them is Wednesday, October 21, 2026. Most are captured at 120 columns so that every column shows. Your terminal may show a slightly narrower or wider layout (Chapter 2).
- Tip, Note and Careful boxes flag shortcuts, background, and things that can lose or change data.

1.7 The terms of use

Rosen is provided free of charge for personal or internal business use, as a binary; it may not be decompiled, sold, rented, leased or sublicensed, and it comes with no warranty. The full text is built into the program and shipped as the file `LICENCE`. To read it, run:

```
rosen -L
```

(`--licence` and `--license` work too.) The first line names the product, the version and the company, so a copy of the output says which release it belongs to. Rosen is a product of **Electric Shepherd**.

2. A Tour of the Screen

Rosen has six pages and a settings page, a handful of dialogs, and one set of keys that works everywhere. This chapter shows what is on the screen and how to move around; the chapters after it take the pages one at a time. The layout, the keys and the forms follow Inventory Central, the sibling program from Electric Shepherd, so if you know one you already know the other.

2.1 The pages

Every page has a number. Press the number to go there, or **Tab** and **Shift-Tab** (or **Right** and **Left**, or **]** and **[**) to step through them. The number keys work on every page; the one place they type a digit instead is a message, a form or a search while you are typing into it (see the end of this chapter).

Key	Page	What it is for
1	Chat	The conversation: read replies, write messages, keep several chats in tabs (Chapter 5).
2	Sessions	Every saved chat, with views, search, sorting, edit, clone, delete, export and import (Chapter 6).
3	Servers	The programs that run models: add, edit, clone, test and choose them (Chapter 7).
4	Models	The models a server offers, one view for each server (Chapter 8).
5	Summary	Counts from your saved chats: how many, how much was said, which servers and models answered (Chapter 11).
6	Checks	Things worth knowing: files others can read, servers that do not answer, chats that could not be saved (Chapter 12).
0	Settings	Colors, clock, start screen and page, sidebar, mouse, new-reply alert and the numbers sent to the model (Chapter 13).

2.2 The sidebar layout

In a window of **100 columns by 24 rows or more** the pages are listed down the left in a sidebar, and the rest of the window belongs to the page. The sidebar has the same background as the page and no line between them; the current page is the highlighted name.

From top to bottom the window holds:

- **The sidebar.** The product name and version, the list of pages with their number keys, and four counts: **Saved chats**, **Open now** (chats open in tabs), **Servers online** (how many answered the last time they were tried; a server counts only after it has been tested or used) and **New replies** (replies that finished while you were not looking; Chapter 15).
- **The title line.** The date, and the time if the clock is on (section 13.1), and at the right the **alert badge**, ! 2 alerts, whenever the Checks page has something that needs attention.
- **The page.** A table, a document or the conversation (section 2.4 describes the tables).

```

Rosen
Wed Oct 21 2026 10:24 AM
! 1 alert

1 Chat
2 Sessions
3 Servers
4 Models
5 Summary
6 Checks
0 Settings

Original order o sorts by the current column
4 servers

> *
  Name      API      Address      Default model      State
Ollama (this computer) Ollama http://loca... llama3.2:3b online (4)
LM Studio      OpenAI http://loca... qwen2.5-coder-7b-instr... online (2)
Workstation (llama.cpp) OpenAI http://192.... mistral-7b-instruct online (1)
Old laptop (switched of... OpenAI http://down... not reachable

SAVED CHATS
5

OPEN NOW
1

SERVERS ONLINE
3 of 4

NEW REPLIES
0

3 of 4 servers online - * marks the server new chats start on
DEMO DATA: nothing is saved (run without -D for your own chats)
Enter edit a add c clone d del u use t test {} look b sidebar ? help q quit

```

Figure 3. The Servers page in the sidebar layout. The sidebar names the pages and keeps four counts in view; the date and the alert badge are above the page; the key hints for this page are on the last line.

- **The bottom three lines.** The **information line** (what the page is showing, or a note about the data), the **message line** (the program tells you what it just did, in green for good news, red for trouble and the accent color for plain information), and the **key hints** for the page. The hints always end with {} look b sidebar ? help q quit ; in a narrow window the end is trimmed to ? help q quit and the first five hints for the page are always kept whole.

b switches the sidebar off and on again, from any page; the choice is kept (the **Sidebar menu** setting, section 13.1). With the sidebar off, a wide window uses the top bar instead.

Note. A message that is too long for the line ends with an ellipsis (...) rather than being cut off in the middle of a word. Nothing is hidden for good: the Checks page and the help pages say the same things at length.

2.3 The top-bar layout

A smaller window, from **50 columns by 12 rows** up to just under 100 by 24, keeps the whole width for the page. The pages are named along the top, with their number key marked in the rule under each name, and the counts are on one line below the date. The product name and version appear at the right of the message line.

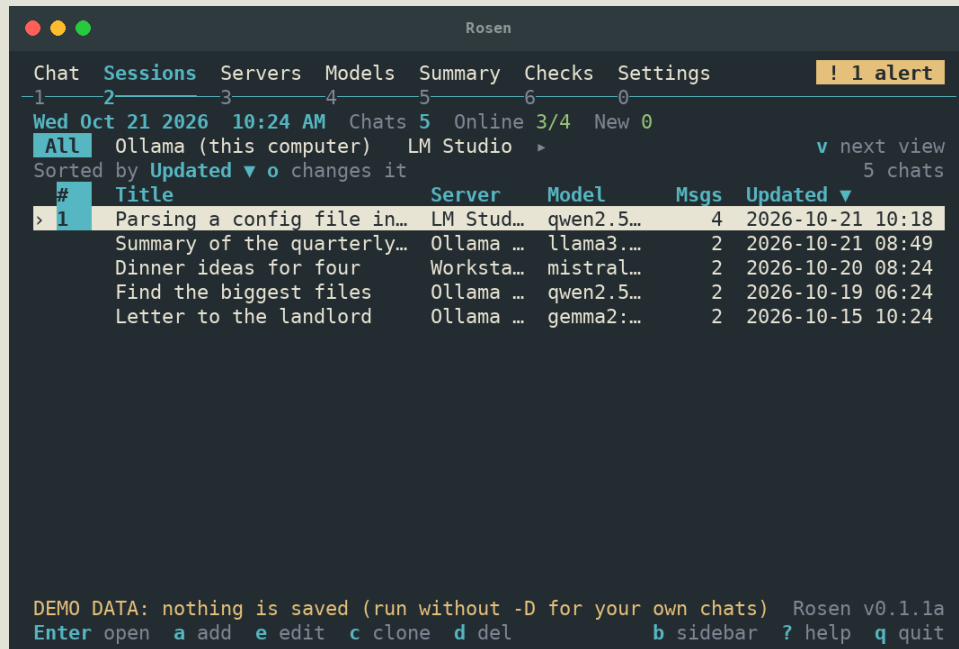


Figure 4. The Sessions page in an ordinary 80 by 24 window.

Below 50 by 12 there is no room to draw anything useful and Rosen says so, and tells you the size it needs; make the window larger and it redraws by itself.

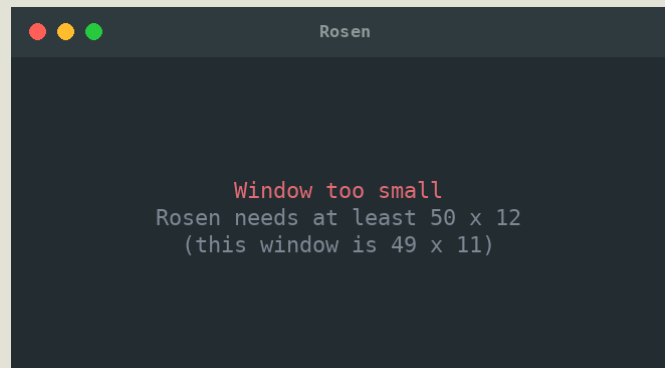


Figure 5. The smallest windows only say that they are too small.

2.4 Lists, views and columns

Sessions, Servers and Models are tables, and they are laid out the same way:

1. **The view bar.** On Sessions and Models the first line names the **views** of the page: All and one for each server on Sessions, one for each server on Models. The current one is highlighted and the right edge says v next view. **v** and **v** step through them; a small arrow at the end of the bar says there are more off to the right.
2. **The sorted-by line.** Sorted by Updated ▼ o changes it, with a count at the right (5 chats). A search or a filter appears here too, as Search: "rust" 1 of 5 shown.
3. **The headings.** The **current column's** heading is highlighted. **h** and **l** move the current column; **o** sorts by it and **f** filters on it. A mark after a heading (▲ or ▼) says which way it is sorted.

4. **The rows.** The highlighted row has a `>` in front and the selection color; in the current column its cell is shown in full, not shortened.

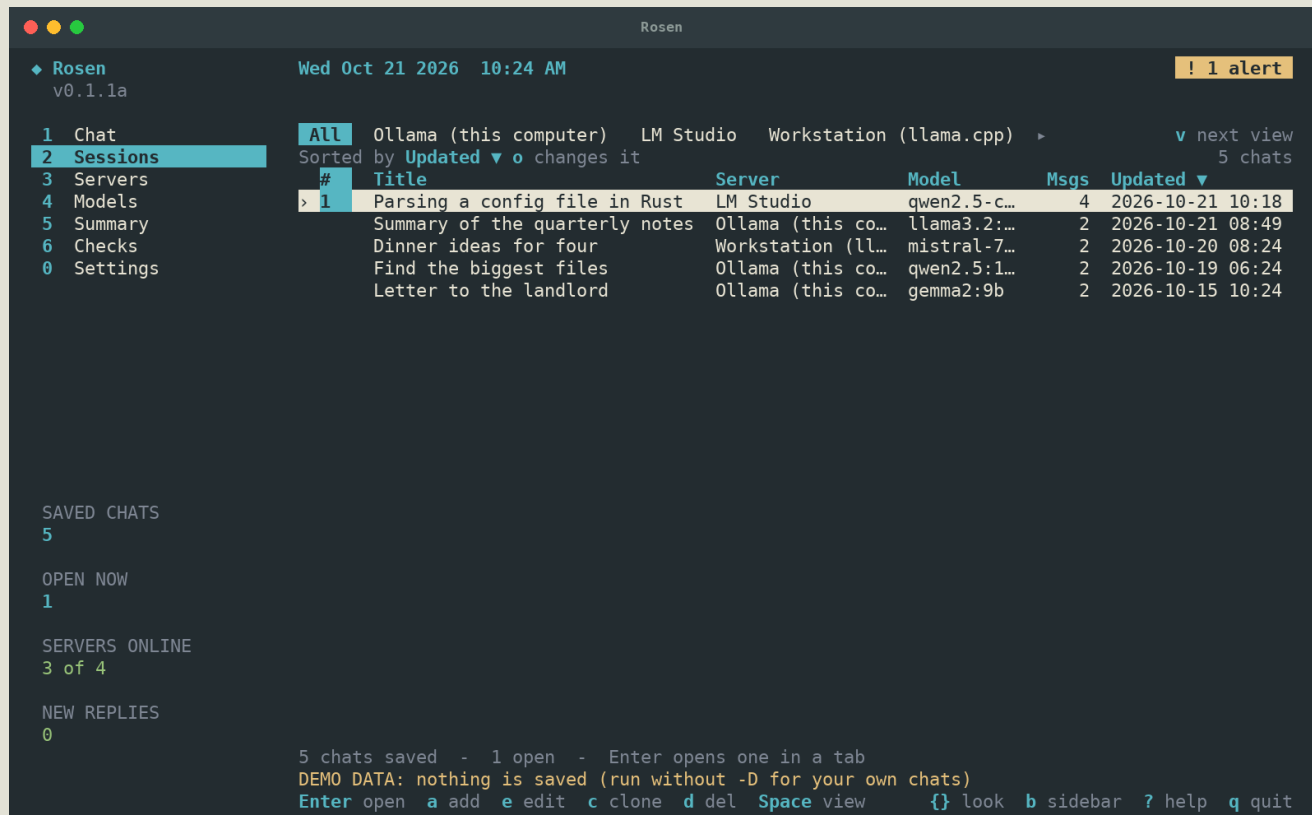


Figure 6. The Sessions page: the view bar, the sorted-by line, the headings and the rows.

2.5 Tabs

On the Chat page, every open chat is a **tab** above the conversation. A tab shows its number and its title. A dot (•) after the title means a reply finished there while you were looking at something else. `v` and `V` (or `Alt-n` and `Alt-p`) move to the next and previous chat, and `Alt-1` to `Alt-9` go to one directly (Chapter 5). The chats are the views of the Chat page.

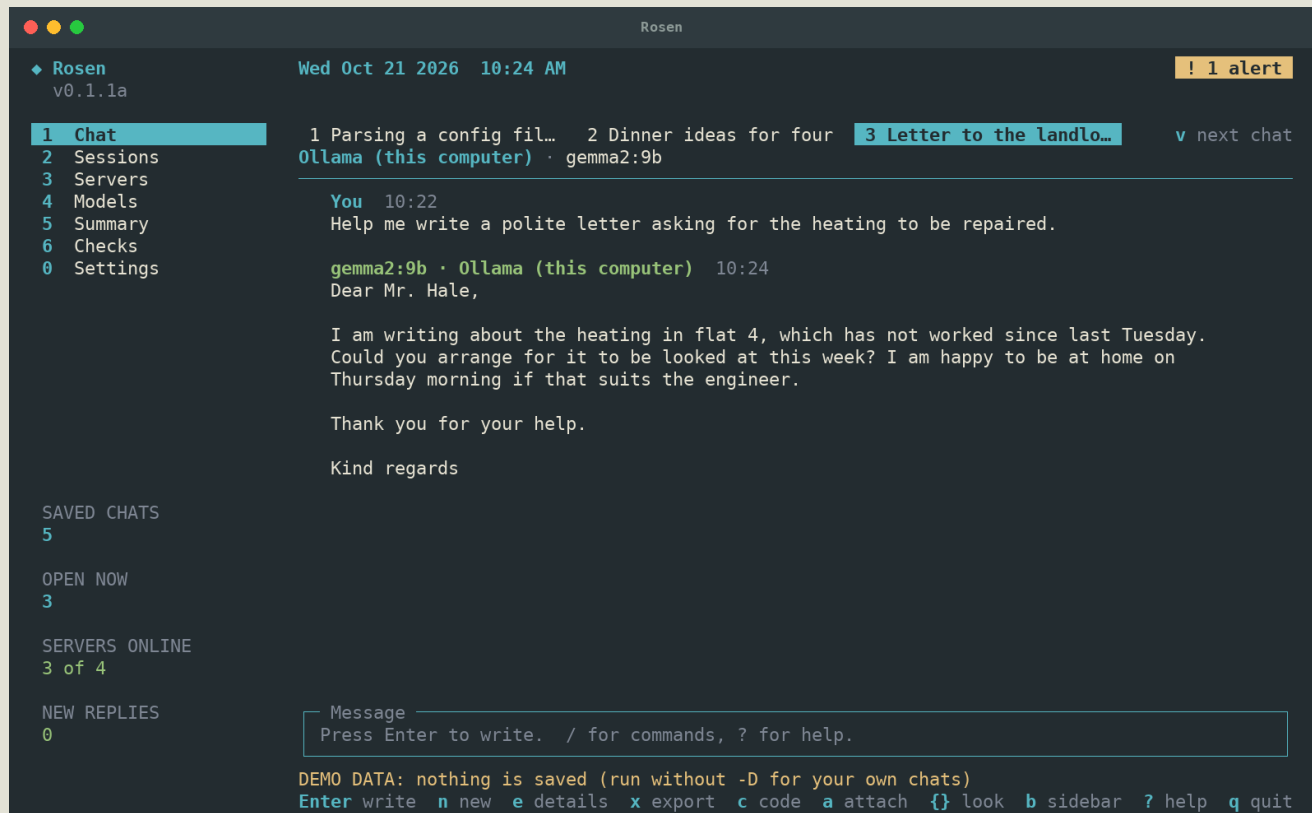


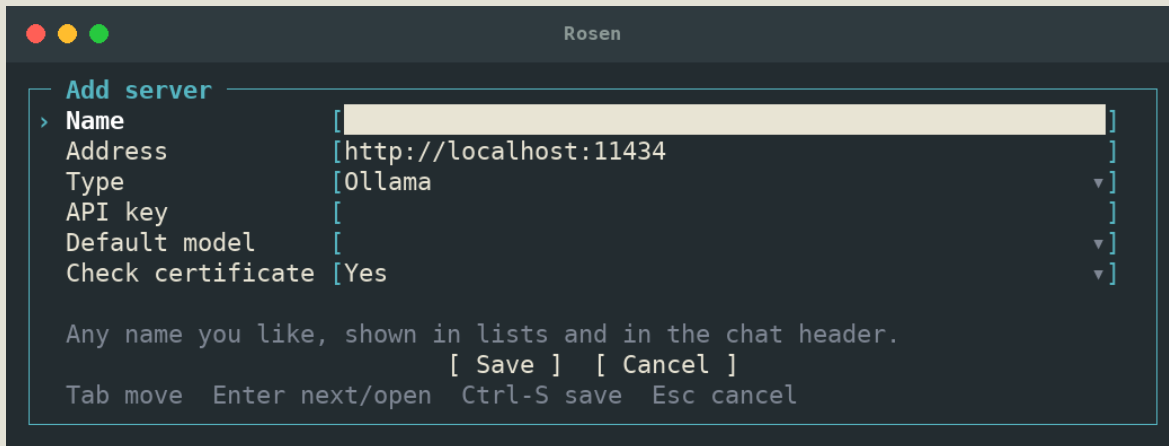
Figure 7. Three chats open at once. Each has its own server and model, shown under the tab bar.

2.6 Dialogs

Whatever needs more than a keystroke opens a **dialog** over the page: a form with a few fields, a menu, a question, the file browser, or a help page. Dialogs have the same few keys everywhere:

- **Tab**, **Down** and **Enter** go to the next field; **Shift-Tab** and **Up** go back. **Enter** on the last text field saves; **Tab** on the last field reaches the [Save] and [Cancel] buttons, which **Enter** or **Space** presses.
- **Ctrl-S** (or **F2**) saves a form from any field.
- On a choice, **Left** and **Right** change it without opening it, and **Enter** opens its menu. Typing in a menu narrows it; **Ctrl-N** and **Ctrl-P** move in it.
- **Esc** closes a menu; a second **Esc** cancels the form. Nothing changes until you save.
- A question is answered with **y** or **Enter** for yes, **n** or **Esc** for no. Every question that could lose something names it.

A field that already holds a value is selected when you reach it: typing replaces the value, and the arrow keys edit it in place. The line under a form explains the field you are on.



Add server

> Name []

Address [http://localhost:11434]

Type [Ollama ▼]

API key []

Default model [▼]

Check certificate [Yes ▼]

Any name you like, shown in lists and in the chat header.

[Save] [Cancel]

Tab move Enter next/open Ctrl-S save Esc cancel

Figure 8. A form: adding a server. The grey line under the fields explains the highlighted one; the buttons are above the key hints.

2.7 Help

Press **?** (or **F1**) on any page for help on **that page**: the keys it understands and what it is for, followed by the keys that work everywhere, the mouse, and, on the Chat page, the commands you can type. In a dialog, **F1** opens help for the dialog: the keys of a form, with the explanation of every field in it; the keys of the file browser; the keys of a menu. **Esc** closes help and puts you back where you were.

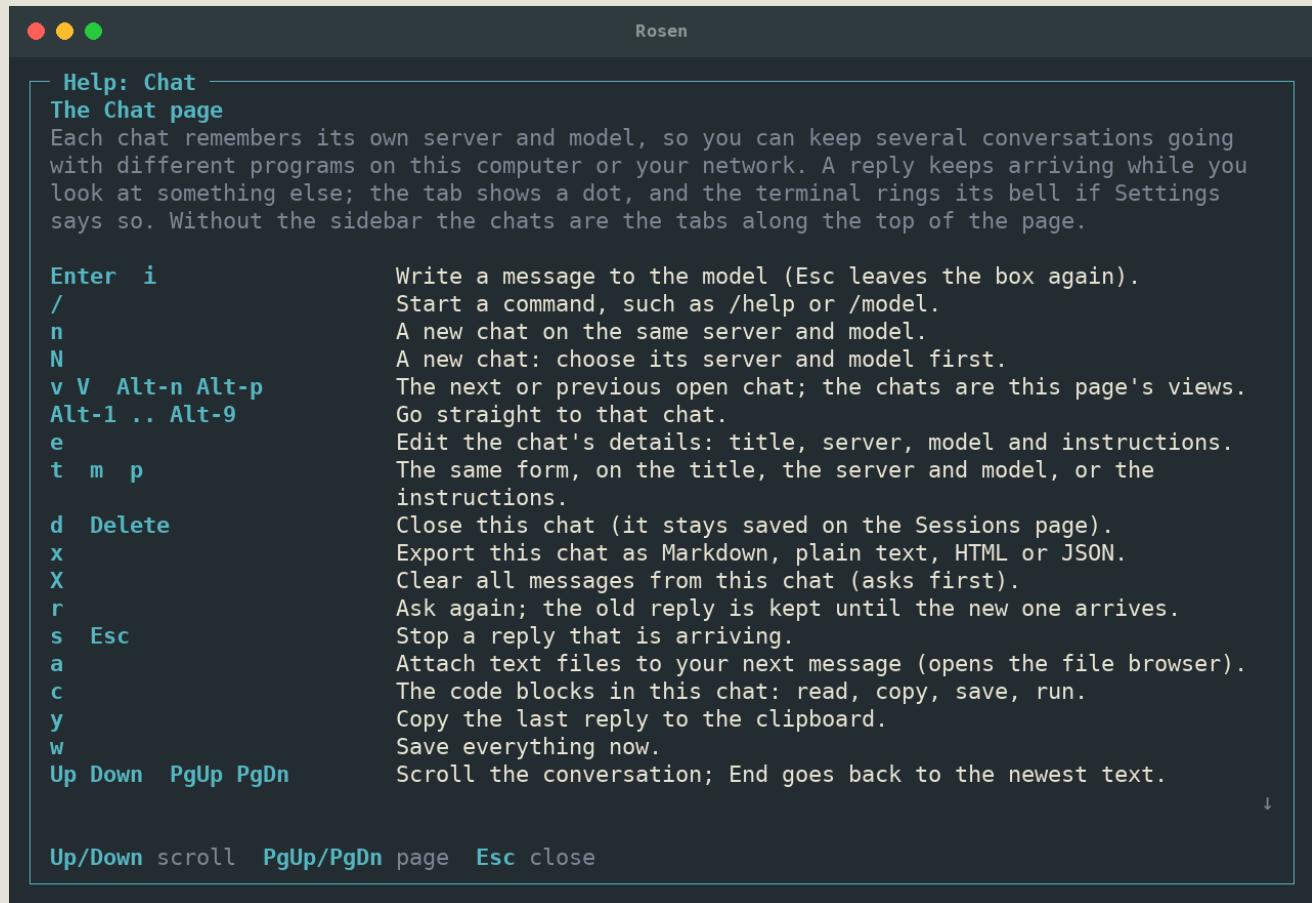


Figure 9. Help for the Chat page. The arrow at the bottom right says there is more below; **PgDn** reads on.

2.8 Keys that work everywhere

Key	What it does
1 to 6 , 0	Go to that page.
Tab , Right ,]	The next page.
Shift-Tab , Left , [	The previous page.
v , V	The next or previous view of the page (on Chat, the next or previous open chat).
b	The sidebar off, or on again.
{ , }	The previous or next look; a message names the one you landed on (Look: winter (3 of 15)).
? , F1	Help for where you are.
! , F2	Open a shell; type <code>exit</code> to come back (Chapter 9).
Ctrl-L	Redraw the screen (after something else has written on it).
q , Ctrl-C	Quit. If a reply is still arriving, Rosen asks first.

When you are typing (a message, a form, a search, a menu you can narrow, the file browser's filter) the letters and digits go into the text; **Esc** leaves, and then the keys above work again. Appendix A lists every key.

Tip. Quitting is always safe: chats are saved as they change, so there is no "save before you quit". If Rosen is stopped from outside (the terminal is closed, or the process is killed), a small helper puts the terminal back in order, and the chats are exactly as they were at the last change (section 17.2).

3. The Mouse

Everything in Rosen can be done from the keyboard. If your terminal reports the mouse, you can also click and use the wheel, and the clicks do exactly what the matching key would have done. The mouse is on by default; this chapter says what it does, and how to turn it off.

3.1 What a click does

Only the **left button** and the **wheel** are used. A click acts on what is drawn at that place on the screen, as it is at that moment.

Click on	What happens
A page name in the sidebar or the top bar	Go to that page, as its number key does.
A row of a table	Highlight it, and make the column you clicked the current column. Click the highlighted row again and it opens, as Enter does: a chat opens in a tab, a server opens its edit form, a model is used.
A heading of a table	Sort by that column; click again to turn the order round.
A name in the view bar (Sessions, Models)	Open that view.
A chat's tab (Chat)	Switch to that chat.
The message box	Start writing.
A key hint on the last line	Press that key. Enter open presses Enter ; ? help opens help.
A line of Checks or Settings	Highlight it; click it again to open it.

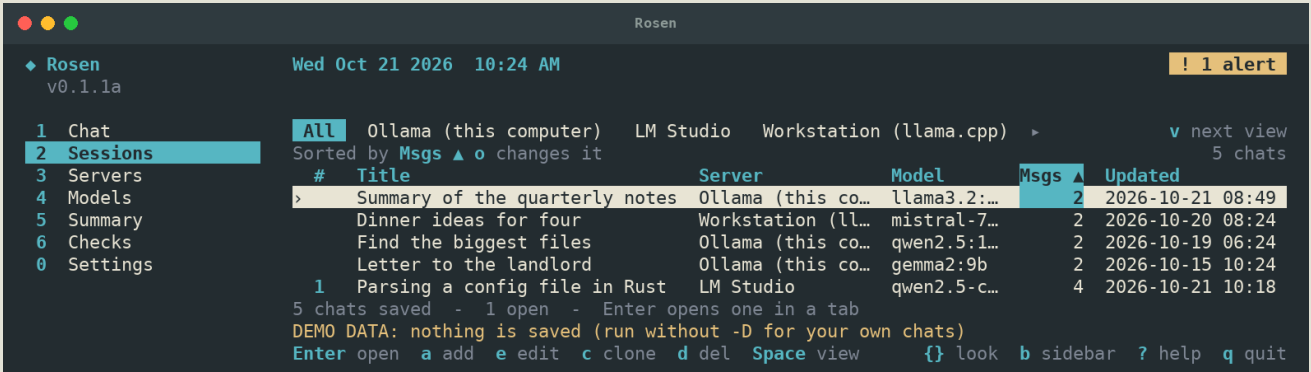


Figure 10. A click on the **Msgs** heading has sorted the chats by it: the line above the table and the mark after the heading say so.

In a form or a menu:

Click on	What happens
A field of a form	Go to that field. A choice opens its menu; a text field puts the cursor in it.
[Save] or [Cancel]	Press the button.
An entry of a form's menu or list of suggestions	Take it.
An entry of a settings menu or the file browser	The first click highlights it (a colors menu previews it), the second takes it.
Outside the dialog	Nothing. A dialog is closed with its buttons or Esc , never by a stray click.

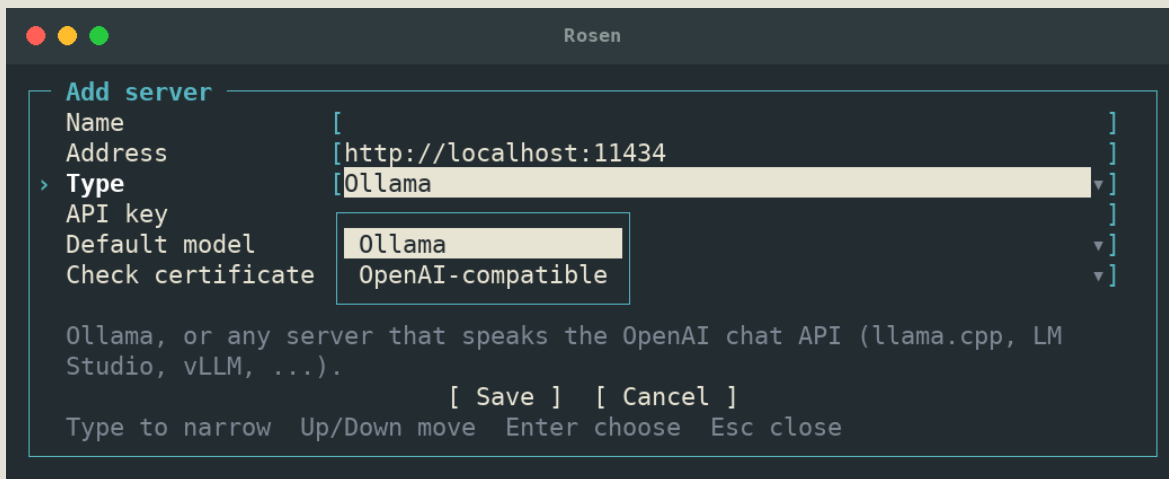


Figure 11. A click on the Type field of a form has opened its menu. Clicking one of the two entries takes it.

3.2 The wheel

The wheel moves **three rows** at a time:

- on the Chat page it scrolls the conversation;
- in a table, and on the Checks and Settings pages, it moves the highlight;
- in Summary, help, a text viewer and the file browser it scrolls;
- in a menu or a list of suggestions it moves the highlight, one entry at a time;
- over a question (yes or no) it does nothing.

3.3 Turning the mouse off

On the Settings page, **Mouse** is **On** or **Off** (section 13.1). It is a real switch, not a promise:

- With **Off**, Rosen never asks the terminal to report the mouse. The terminal keeps its own mouse behavior, so you can select text and copy it as usual.
- With **On**, the terminal sends clicks and wheel turns to Rosen instead. Most terminals let you select text anyway by holding **Shift** (in some, **Alt** or **Option**) while you drag.
- If a stray report arrives while the setting is Off, it is ignored.

- Rosen asks for the mouse again after you come back from a shell, and gives it back when it ends, however it ends: a helper puts the terminal in order even if Rosen is killed.

Note. Rosen asks for plain button reports in the SGR form (CSI ? 1000 h with CSI ? 1006 h), which every current xterm-compatible terminal understands. Dragging, the middle and right buttons and double clicks are not used; "click again" means two separate clicks on the same row.

Tip. If clicking seems to do nothing, check that **Mouse** is **On** (page 0), and that the program is not inside a terminal multiplexer that keeps the mouse for itself.

4. Quick Start

This chapter takes you from a fresh install to a conversation with a model in about five minutes. It assumes you have a model server running somewhere you can reach. If you do not, skim it, then try everything on the built-in echo server (Chapter 18).

4.1 1. Start Rosen

```
rosen
```

Press any key at the splash screen. You land on the **Chat** page with one empty chat. The first time, Rosen has made two servers for you: **Ollama (this computer)** at `http://localhost:11434` and **OpenAI-compatible (this computer)** at `http://localhost:8080`. If one of those is what you run, skip to step 3.

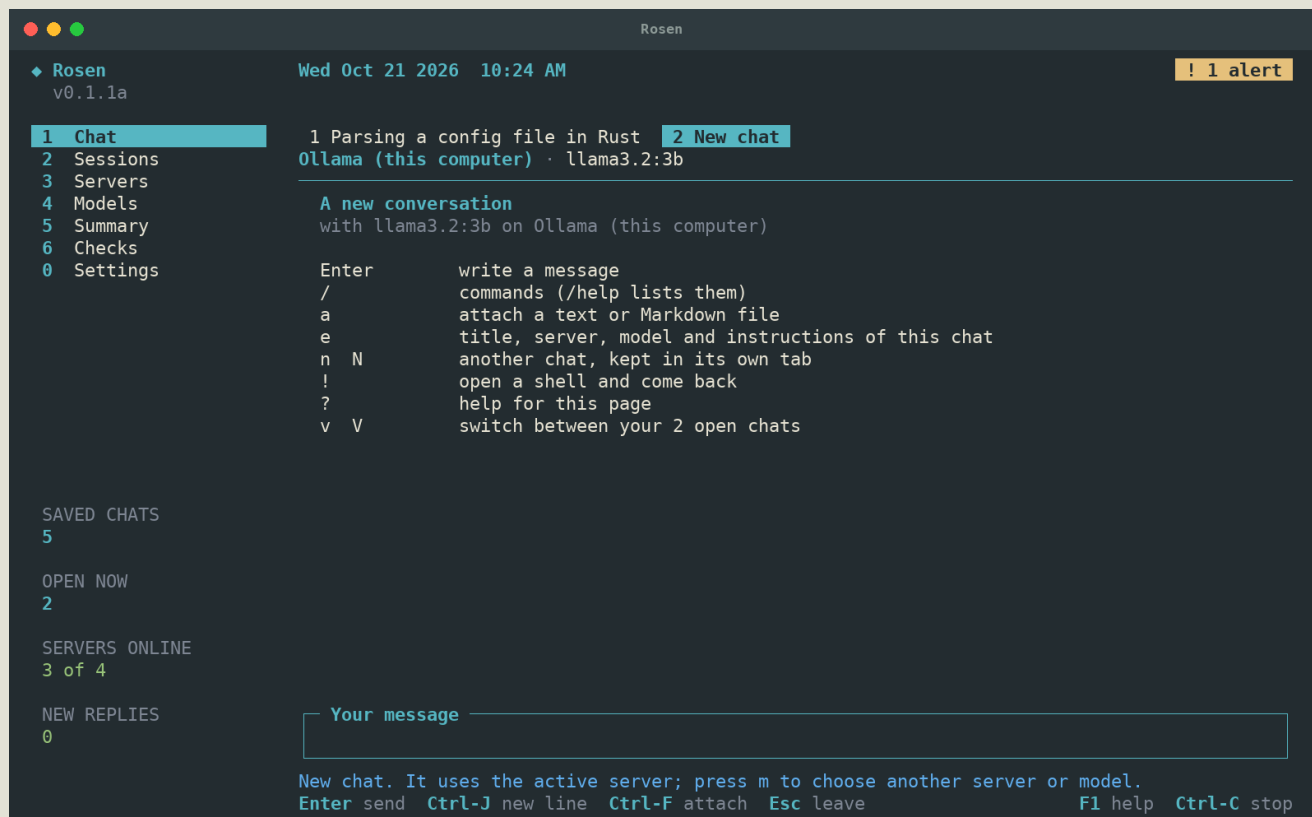
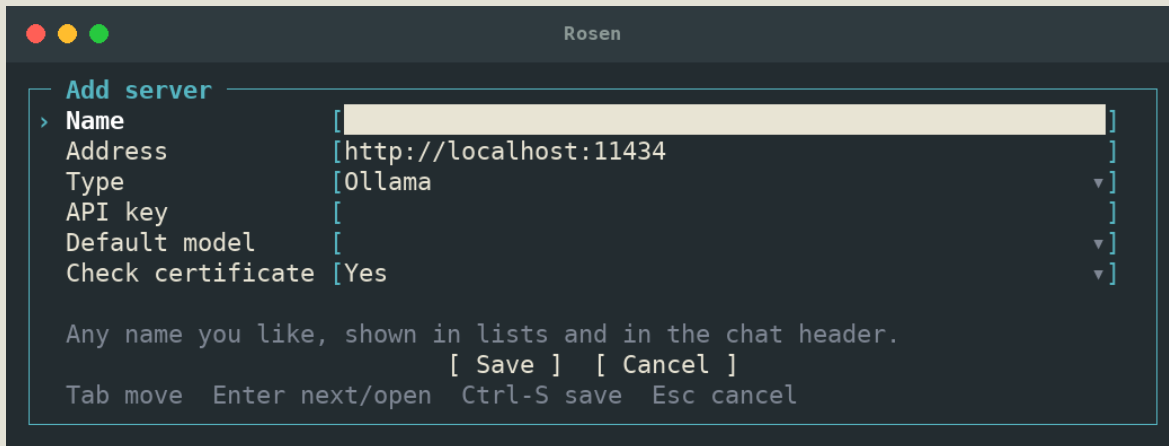


Figure 12. A new chat. The tips list the keys that matter; the box below them is where you write, and the line under it shows what Rosen last did.

4.2 2. Tell Rosen where your server is

Press **3** for the **Servers** page and then **a** to add a server.



Add server

> **Name** []

Address [http://localhost:11434]

Type [Ollama ▼]

API key []

Default model [▼]

Check certificate [Yes ▼]

Any name you like, shown in lists and in the chat header.

[Save] [Cancel]

Tab move Enter next/open Ctrl-S save Esc cancel

Figure 13. The Add server form.

Fill in:

1. **Name:** anything you like, such as `workstation (llama.cpp)`.
2. **Address:** where the server is, for example `http://192.168.1.40:8080`. If you leave off `http://` it is assumed. Paste the address with or without a trailing `/v1` or `/api`; Rosen removes one of them.
3. **Type:** **Ollama** for Ollama, **OpenAI-compatible** for everything else (llama.cpp's `llama-server`, LM Studio, vLLM, LocalAI and most others).
4. **API key:** only if the server wants one. It is shown as dots.
5. **Default model:** leave it empty for now; step 3 shows you the list.
6. **Check certificate:** leave it on. It only matters for `https://` addresses (section 7.7).

Before you save, press **ctrl-T** to **test** the connection with what you have typed. Rosen asks the server for its models and then sends it a one-word question; the message line reports what happened. Then press **ctrl-S** to save, or **Tab** to the [Save] button and press **Enter** (or click it).

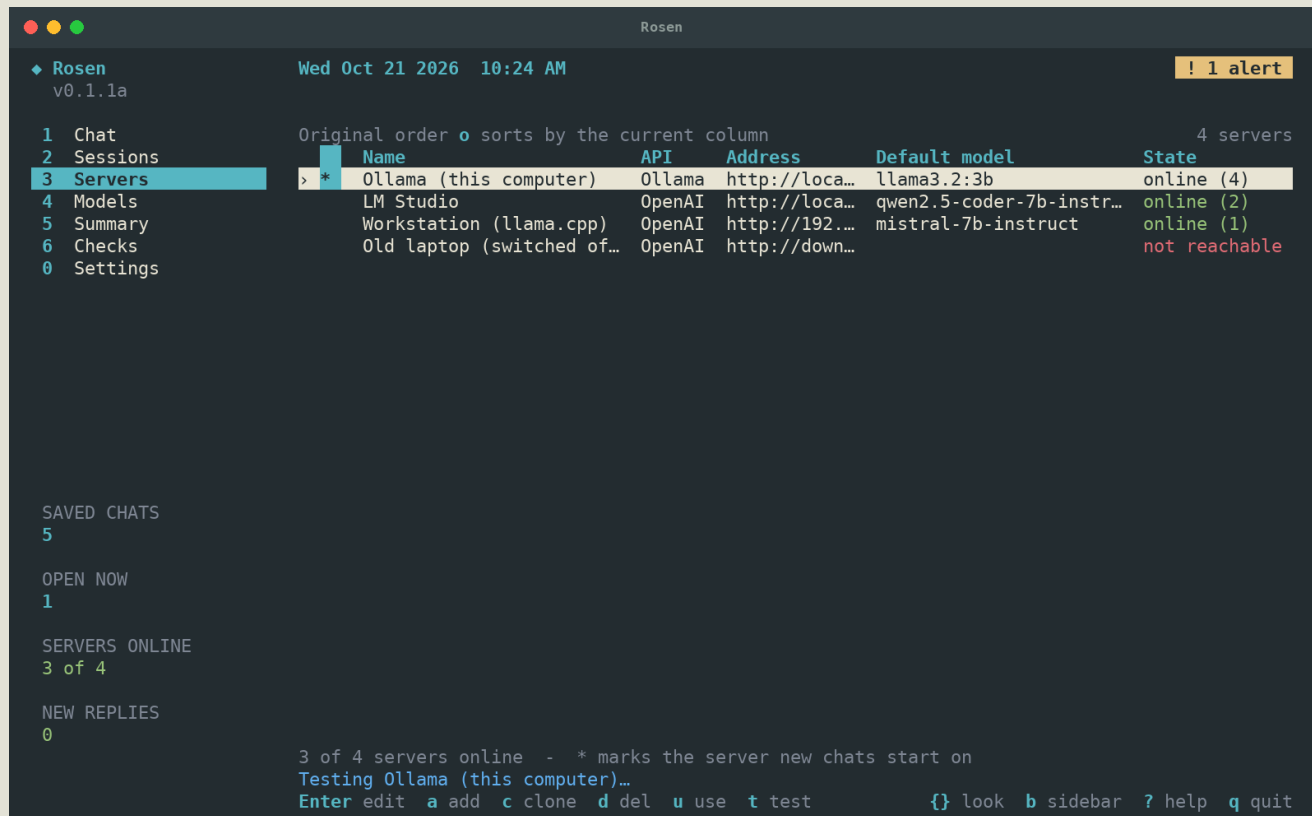


Figure 14. Testing a server from the Servers page (**t**). The message line says what is going on; the State column fills in when it is done.

Tip. If the type is wrong, the test says so in plain words ("This server speaks the OpenAI-compatible API, not Ollama") instead of showing an error from the protocol, and the same goes for the model list and the first message.

4.3 3. Choose a model

Highlight your server and press **u** to use it for the current chat. Then press **4** for the **Models** page, which lists what the server offers, highlight a model and press **Enter**. The chat now uses that server and model; the header above the conversation says which.

To make it the model the server uses for new chats, press **D** on it ("default").

4.4 4. Say something

Press **1** to go back to the chat, **Enter** to open the message box, type your question, and press **Enter** to send. (A chat you start with **n** opens the box for you.) The reply appears as it is written. **Esc** stops a reply that is going the wrong way; what has arrived is kept.

- **Ctrl-J** (or **Alt-Enter**) starts a new line inside a message; pasting keeps its line breaks.
- **Esc** leaves the box without sending, and keeps what you typed.
- **PgUp** and **PgDn** scroll the conversation; **End** jumps to the newest text.

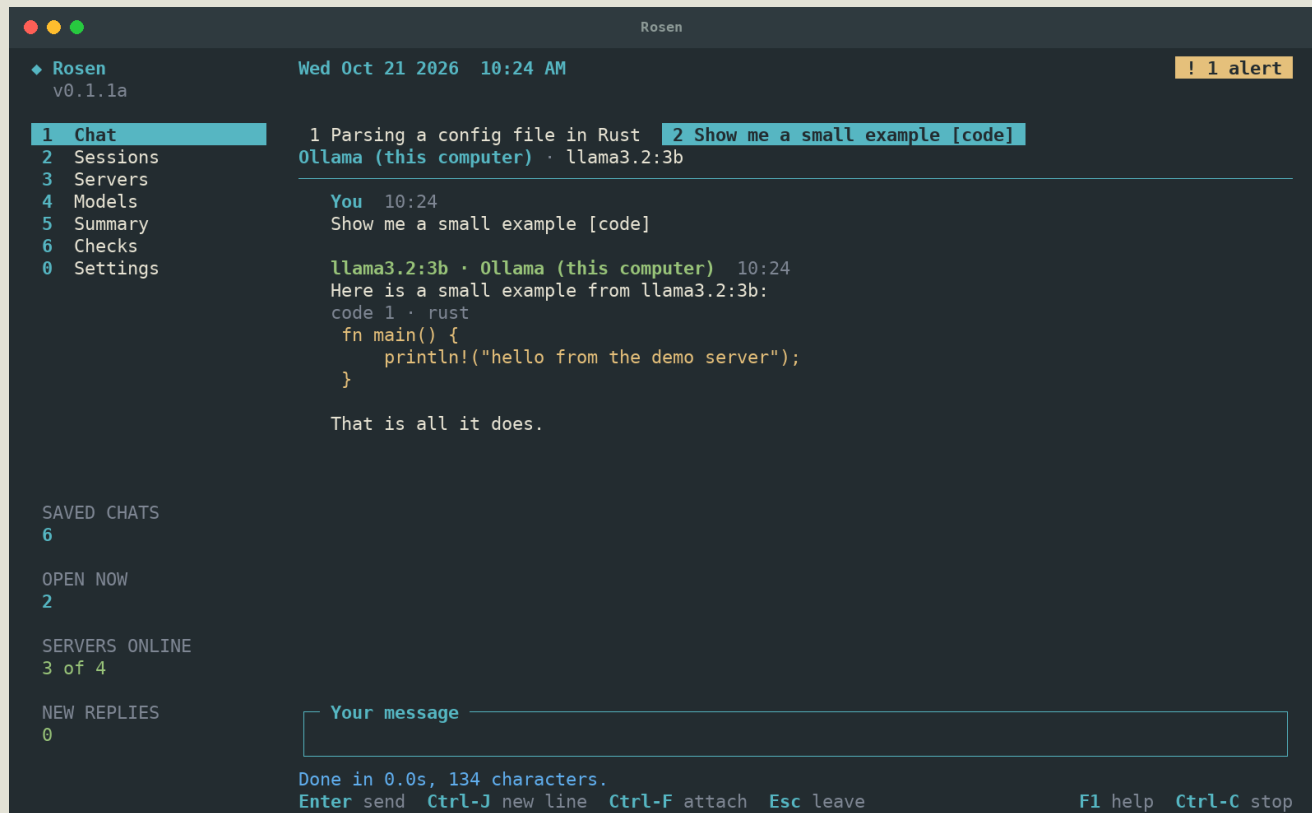


Figure 15. A reply with a code block. Press **c** to read, copy, save or run it.

4.5 5. What next

- Press **n** for another chat on the same server, or **N** to choose the server and model for it first. Switch between chats with **v** and **V**, or with **Alt-1** to **Alt-9**.
- Press **c** to see the code blocks in a reply: read, copy, save or run them.
- Press **a** to attach a text file to your next message; **x** to export the chat as Markdown or text; **e** to change its title, server, model or instructions.
- Press **{** or **}** to try the other looks, **b** to hide the sidebar, and click anywhere you like: the mouse works too (Chapter 3).
- Press **?** on any page for help on that page.

Your chats are saved as soon as they change: there is nothing to remember to do. Chapter 17 says where they are.

5. Chat

The Chat page (**1**) is where you talk to a model. It shows one chat at a time, with a tab for every chat that is open. Each chat remembers its own server and model, its own instructions if you gave it any, and everything that was said.

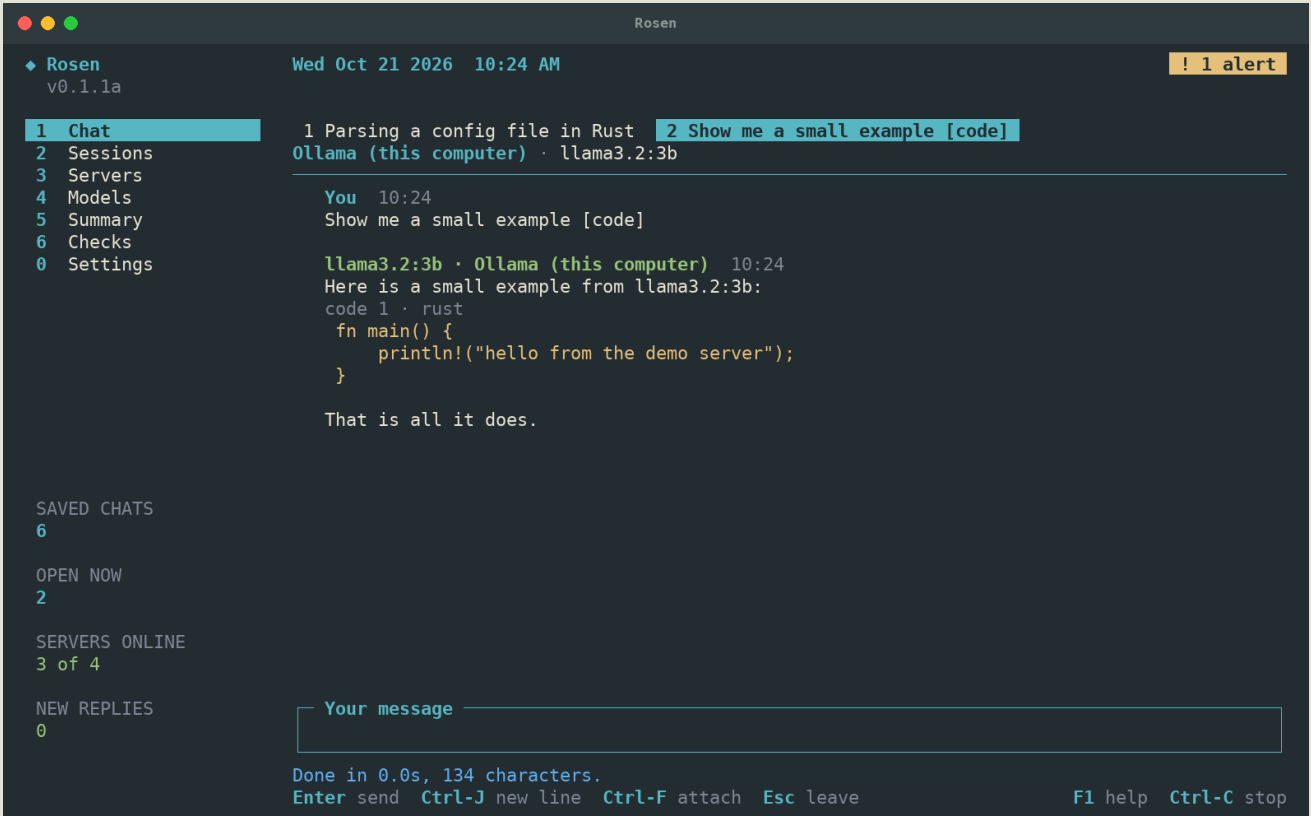


Figure 16. A chat. The tab bar and the header (server and model) are at the top, the conversation fills the middle, and the message box is at the bottom.

5.1 Writing a message

Press **Enter** (or **i**) to open the message box; the box changes from "Message" to "Your message". Type, and press **Enter** to send. A chat you start with **n** opens the box for you. You can also click the box with the mouse.

Key	What it does
Enter	Send the message.
Ctrl-J , Alt-Enter	A new line inside the message.
Esc	Leave the box; the text stays for later.
Ctrl-C	Stop a reply that is arriving; if none is, leave the box.
Ctrl-F	Attach text files to this message (section 10.5).
Ctrl-X	Remove the last attachment.
Ctrl-R	Ask again (section 5.4).

Key	What it does
Ctrl-N	Start a new chat.
F2	Open a shell (Chapter 9).
Up , Down , PgUp , PgDn	Move in the text. At the first or last line they scroll the conversation instead.
Home , End , Ctrl-A , Ctrl-E	The start and end of the line.
Ctrl-W , Ctrl-U , Ctrl-K	Delete a word back, everything back to the start, or everything on to the end.

A block of text pasted into the box arrives whole, with its line breaks, and is not sent until you press **Enter** . The box holds up to 200,000 characters.

A message whose first character is `/` is a command, not text for the model (section 5.9). To send text that really starts with a slash, start it with two: `//usr/bin` is sent as `/usr/bin`. A line that starts with a slash but is not a command is **not** thrown away: Rosen tells you it does not know the command and puts your text back in the box so you can fix it.

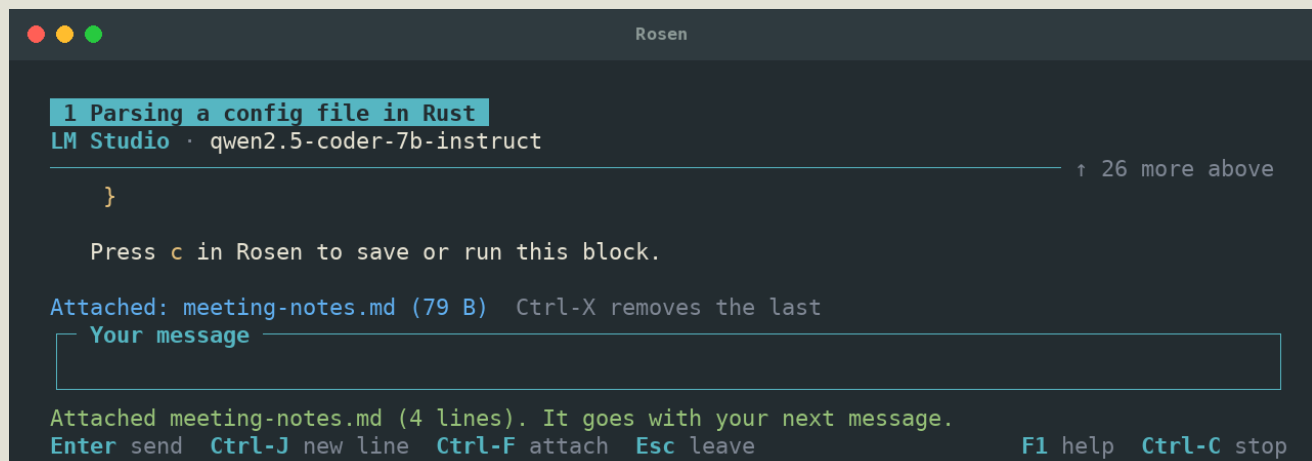


Figure 17. A message with a file attached. The line above the box names it and shows its size; **ctrl-x** removes it.

5.2 Reading replies

A reply is shown under the name of the model and server that wrote it, with the time. While it is arriving the header shows how long it has been going and reminds you that **Esc** stops it.

- **Scrolling.** **Up** and **Down** scroll a line, **PgUp** and **PgDn** a page, **End** goes back to the newest text. When you have scrolled up, the header says how many lines are above.
- **Code blocks** are drawn on a background of their own and numbered: code 2 · rust. Section 9.2 shows what to do with them.
- **Copying.** **y** copies the last reply to the clipboard, and **c** then **y** copies a code block. Copying uses the terminal's clipboard feature (OSC 52); a few terminals leave it off by default, in which case Rosen says so and you can select text with the mouse as usual (turn **Mouse** off in Settings, or hold **shift** while you drag; Chapter 3).
- **The wheel** scrolls the conversation three lines at a time.

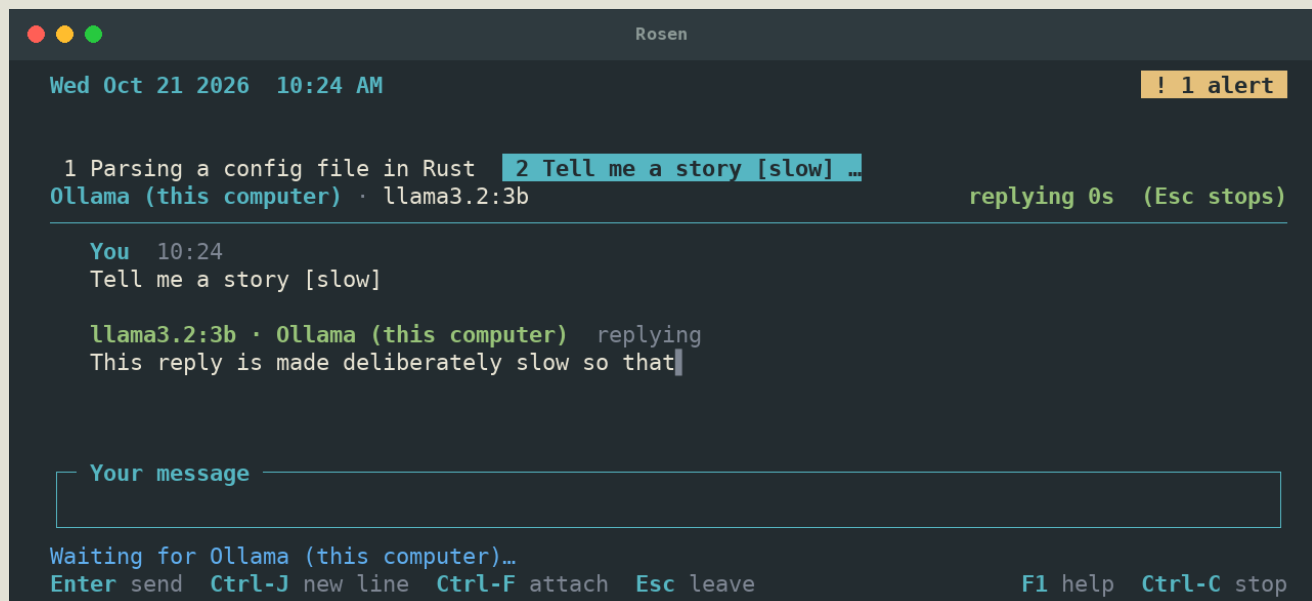


Figure 18. A reply arriving. The tab carries an ellipsis while the chat is busy, and the header says replying 0s (Esc stops).

- **Streaming.** If you would rather see a reply all at once, turn off **Show replies as they arrive** in Settings (section 13.2).

5.3 When something goes wrong

Your message is saved the moment you send it, before anything is asked of the server, and nothing that goes wrong afterwards takes it away.

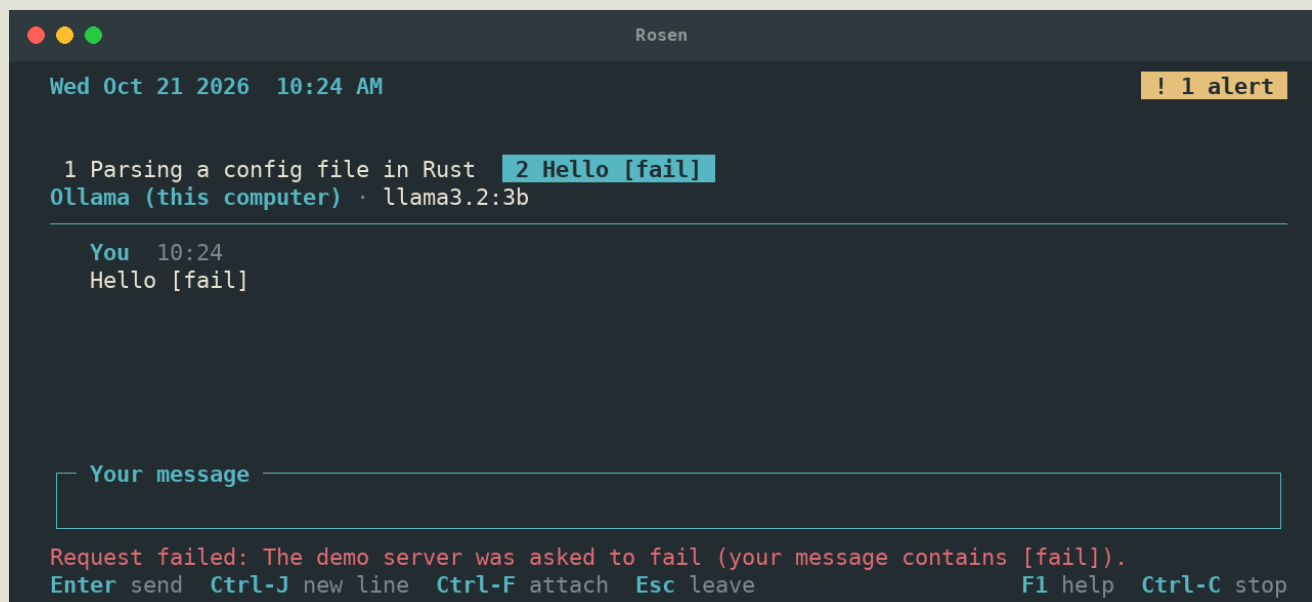


Figure 19. A request that failed. The message line says why in plain words; your message is still in the chat.

- **The server did not answer, or said no.** The message line says what happened ("Could not connect...", "the server answered 404..."). Fix the cause, then press `r` (or `ctrl-R` in the box) to ask again.
- **A reply was cut short.** If the connection drops, or the server stops before it has said it is finished, the reply is marked **stopped** and counts as an error: you are told, and what arrived is kept so you can read it. `r` asks again.
- **You pressed `Esc`.** A reply you stop yourself is kept as far as it got and marked stopped too.
- **No model is chosen.** Rosen says so and tells you to press `m` (section 5.6).

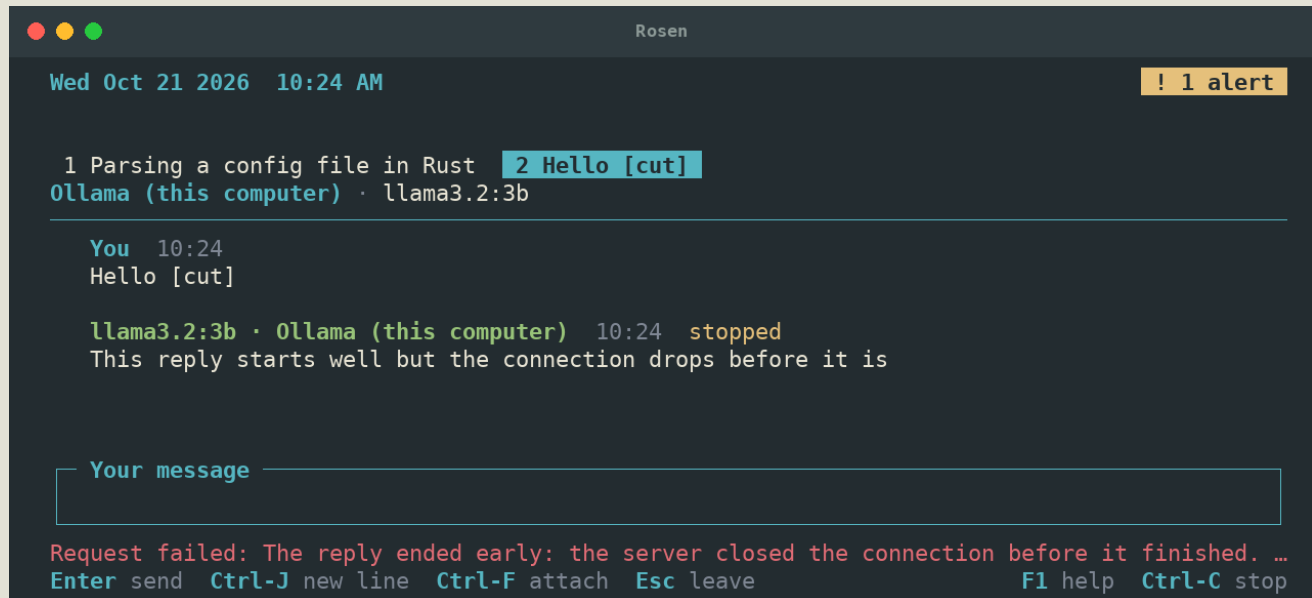


Figure 20. A reply cut short: it is marked stopped, and the part that arrived is kept.

5.4 Asking again

Press `r` (or `ctrl-R` inside the box) to ask the model for another answer to your last message. **The old reply stays** until the new one has arrived in full. If the new one fails, the old one is still there; if it succeeds, it takes the old one's place.

5.5 Several chats

Every chat is a tab, and the chats are the **views** of the Chat page. Press `n` to open a new one that starts on the same server and model as the one you are in, with the message box ready, or `N` to choose the server and model first. `v` and `V` (or `Alt-n` and `Alt-p`) step through the open chats, and `Alt-1` to `Alt-9` go straight to one. You can click a tab too. You can have up to nine open; the Sessions page keeps the rest (Chapter 6).

`d` (or `Delete`) closes the tab. The chat is **not** deleted: it stays on the Sessions page, and opening it there brings it back. If a reply is still arriving, Rosen asks first. `x` is different: it clears every message from this chat, after asking.

A reply keeps arriving in a chat you are not looking at, and several chats can be waiting on several servers at once. When a reply finishes out of sight you are told (Chapter 15).

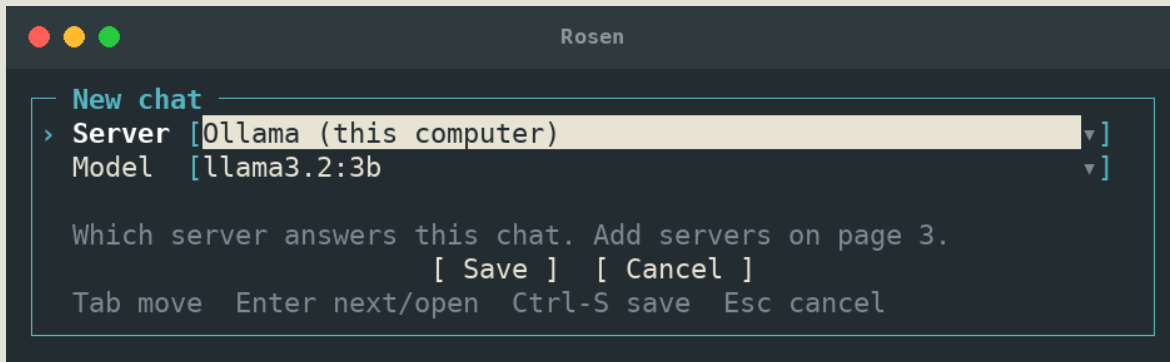


Figure 21. **N**: a new chat on a server of your choice.

5.6 This chat's server and model

Press **e** to edit the details of **this** chat in one form: its **Title**, its **Server**, its **Model** and its **Instructions**. **m** opens the same form with the cursor on the server, **t** on the title and **p** on the instructions. The chat keeps what you choose from then on, even if you change the default on the Servers page later, and when you open the chat again days later it asks the same model.

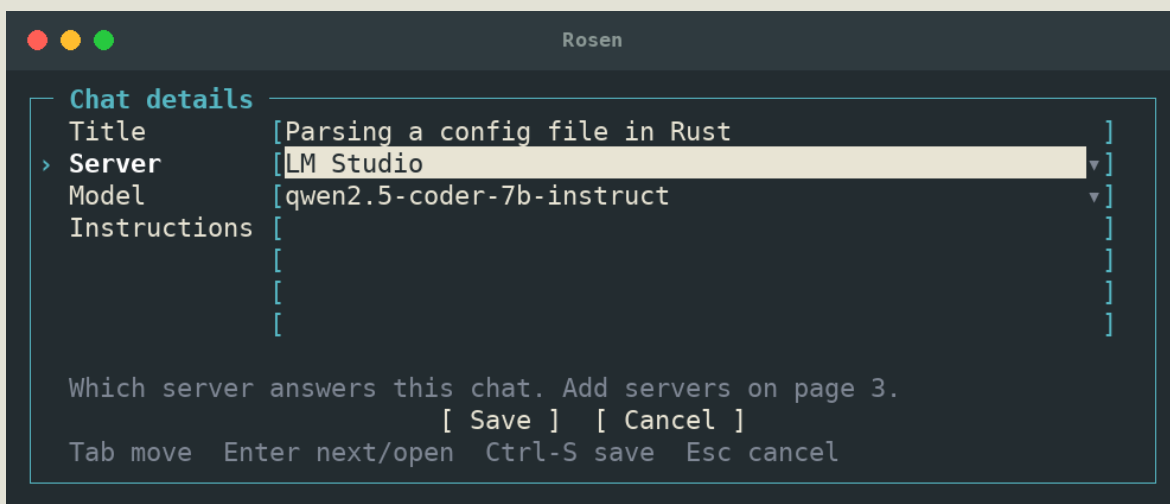


Figure 22. The chat's details. The server is a menu; the model is typed or chosen from that server's list (page 4); choosing another server brings its default model with it.

Choose **Save** with **ctrl-S**, or **Tab** to the button and **Enter**. Nothing changes until you do.

You can change them in the middle of a conversation. The whole conversation (as far as Memory allows, below) is sent to the new model, so it carries on where the last one left off.

5.7 What the model is told

With every message Rosen sends the model:

1. **The instructions** (the "system prompt"): this chat's own, if you set some in the details form (**e** or **p**) or with `/system`, otherwise the default from Settings. You are a helpful assistant. is the default until you change it.
2. **The recent messages**. The last **Memory** messages of the chat (40 by default; 0 sends the whole chat). A chat that is longer than that keeps all its text on screen and on disk, but the model only

sees the latest part. When the chat is longer than that, the header says so (remembers the last 40 of 63 messages).

3. **Your new message**, with any attached files after it, each as [File: name] followed by the text in a code block.

The numbers that shape the reply (Temperature, Top-p, Longest reply, Time limit) come from Settings (section 13.2) and apply to every chat; `/set` changes them from inside a message (section 5.9).

5.8 Keys on the Chat page

These work when the message box is closed:

Key	What it does
<code>Enter</code> , <code>i</code>	Write a message.
<code>/</code>	Start a command.
<code>n</code>	A new chat on the same server and model, ready for typing.
<code>N</code>	A new chat on a server and model you choose.
<code>v</code> , <code>V</code> , <code>Alt-n</code> , <code>Alt-p</code>	The next or previous open chat.
<code>Alt-1</code> to <code>Alt-9</code>	Go straight to that chat.
<code>e</code>	Edit the chat's details: title, server, model and instructions.
<code>t</code> , <code>m</code> , <code>p</code>	The same form, with the cursor on the title, the server and model, or the instructions.
<code>d</code> , <code>Delete</code>	Close this chat (it stays saved).
<code>x</code>	Export this chat.
<code>X</code>	Clear every message from this chat (asks first).
<code>r</code>	Ask again; the old reply is kept until the new one arrives.
<code>s</code> , <code>Esc</code>	Stop a reply that is arriving.
<code>a</code>	Attach text files to the next message.
<code>c</code>	The code blocks in this chat.
<code>y</code>	Copy the last reply.
<code>w</code>	Save everything now (it is saved anyway).
<code>Up</code> , <code>Down</code> , <code>j</code> , <code>k</code> , <code>PgUp</code> , <code>PgDn</code> , <code>Space</code>	Scroll the conversation.
<code>g</code> , <code>G</code> , <code>Home</code> , <code>End</code>	The first message; the newest text.

The page keys of Chapter 2 (`Tab` , `b` , `{` and the rest) work here too. The keys that used a letter of their own in earlier versions have moved to match Inventory Central; Appendix B lists them.

5.9 Commands in a message

Type a command at the start of a message and press `Enter` . Everything a key does has a command too, and a few commands have no key.

Command	What it does
<code>/help</code>	Open help.
<code>/new</code>	A new chat.
<code>/server [name or number]</code>	Choose the server for this chat, by part of its name or by its place in the list (1 is the first server you added). With nothing, opens the menu.
<code>/model [name]</code>	Choose the model for this chat. With no name, opens the menu.
<code>/system [text]</code>	Instructions for this chat. <code>/system</code> - removes them and goes back to the default; with no text it opens the box.
<code>/rename [title]</code>	Rename the chat.
<code>/attach [file]</code>	Attach a text file. With no name, opens the file browser.
<code>/import [file]</code>	Import a saved chat, or load a text or Markdown file (section 10.4).
<code>/export [md txt html json] [file]</code>	Export this chat (section 10.2).
<code>/exportall</code>	Export every saved chat into a folder.
<code>/regen, /stop, /copy</code>	Ask again, stop a reply, copy the last reply.
<code>/code</code>	The code blocks of this chat.
<code>/run [n]</code>	Run code block <i>n</i> and show what it prints (asks first).
<code>/shell, /! [command]</code>	Open a shell, or run one command in it (Chapter 9).
<code>/look [name]</code>	Change the colors to any of the fifteen looks by name (full, teal, winter, harbor, dusk, shepherd, spring, autumn, ember, samhain, white, green, amber, classic-dark, classic-light). With no name, or one it does not know, it lists the names. The look is kept, as if chosen on the Settings page.
<code>/set [name value]</code>	Change temperature, top_p, max_tokens, memory, timeout or stream (on or off).
<code>/clear, /delete, /close</code>	Empty, delete or close the chat (the first two ask).
<code>/chats, /servers, /models, /settings</code>	Go to that page.
<code>/save</code>	Save the chat now.
<code>/quit</code>	Quit.
<code>//text</code>	Send <code>/text</code> as an ordinary message.

Command names are not case sensitive. `/theme` and `/colors` are other names for `/look`, and `/sessions` and `/open` for `/chats`, so what you typed in `llm-tui` still works.

6. Sessions

The Sessions page (**2**) lists every chat Rosen has saved. A chat is saved the moment it changes, so this page is also where you find yesterday's conversation, edit its details, clone it, export it, or delete it for good.

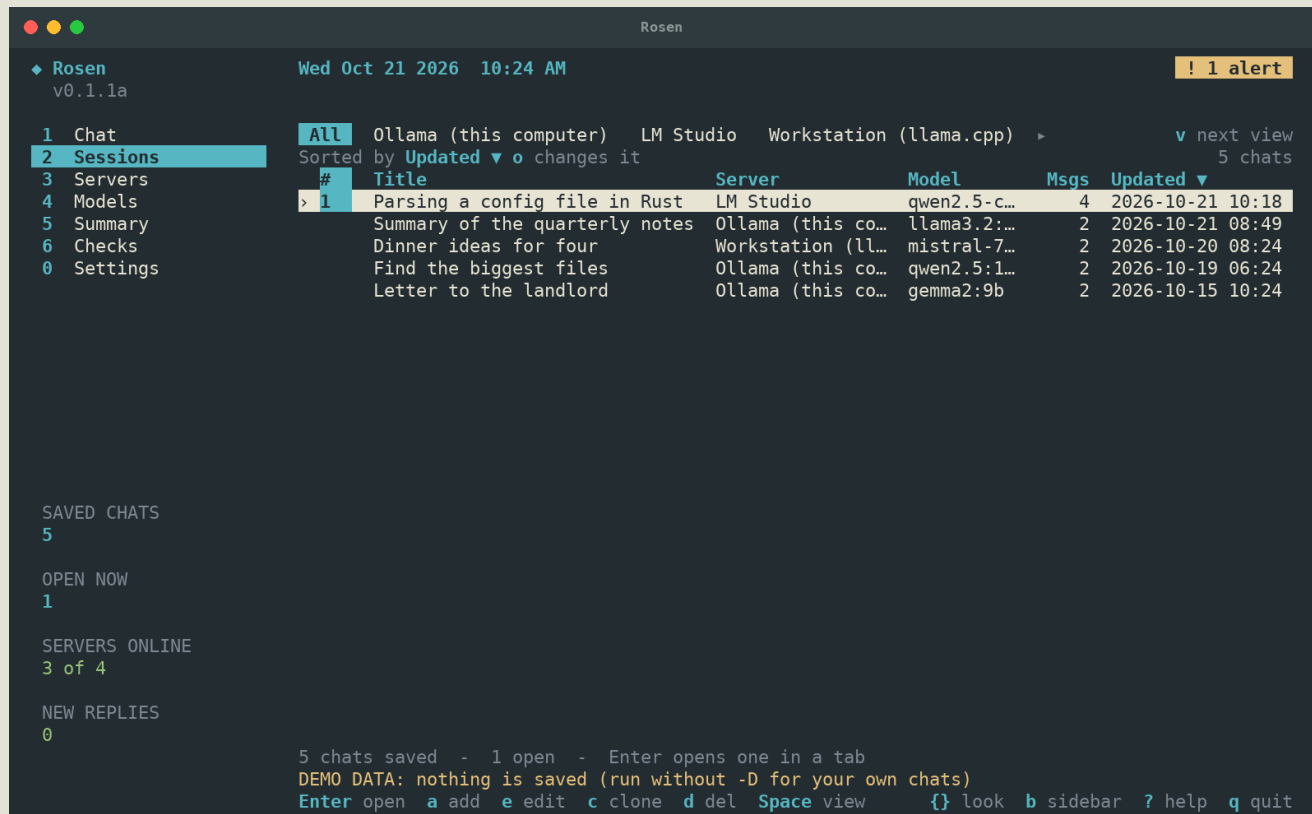


Figure 23. The Sessions page. The view bar is at the top, the sort is shown above the table, and the count is at the right of that line.

6.1 The table

Column	What it shows
#	The number of the tab the chat is open in (1 to 9), or nothing when it is not open.
Title	The chat's title. A new chat is titled from your first message; rename it at any time.
Server	The server the chat uses.
Model	The model the chat uses.
Msgs	How many messages the chat holds, yours and the model's.
Updated	When it last changed.

The highlighted chat, marked with **>**, is the one the keys act on. The current column (the one with the highlighted heading) is shown in full; the others are shortened with ... when they do not fit, so widen the

window or move the current column (`h` and `l`) to see more. When the window is too narrow for every column, a `>` or `<` at the edge of the headings says that more are off to that side.

6.2 Views

The bar above the table names the **views**: **All**, and one for each server. `v` steps to the next view and `V` to the previous, or you can click a name. A server's view shows only the chats that use that server, and the count at the right says how many.

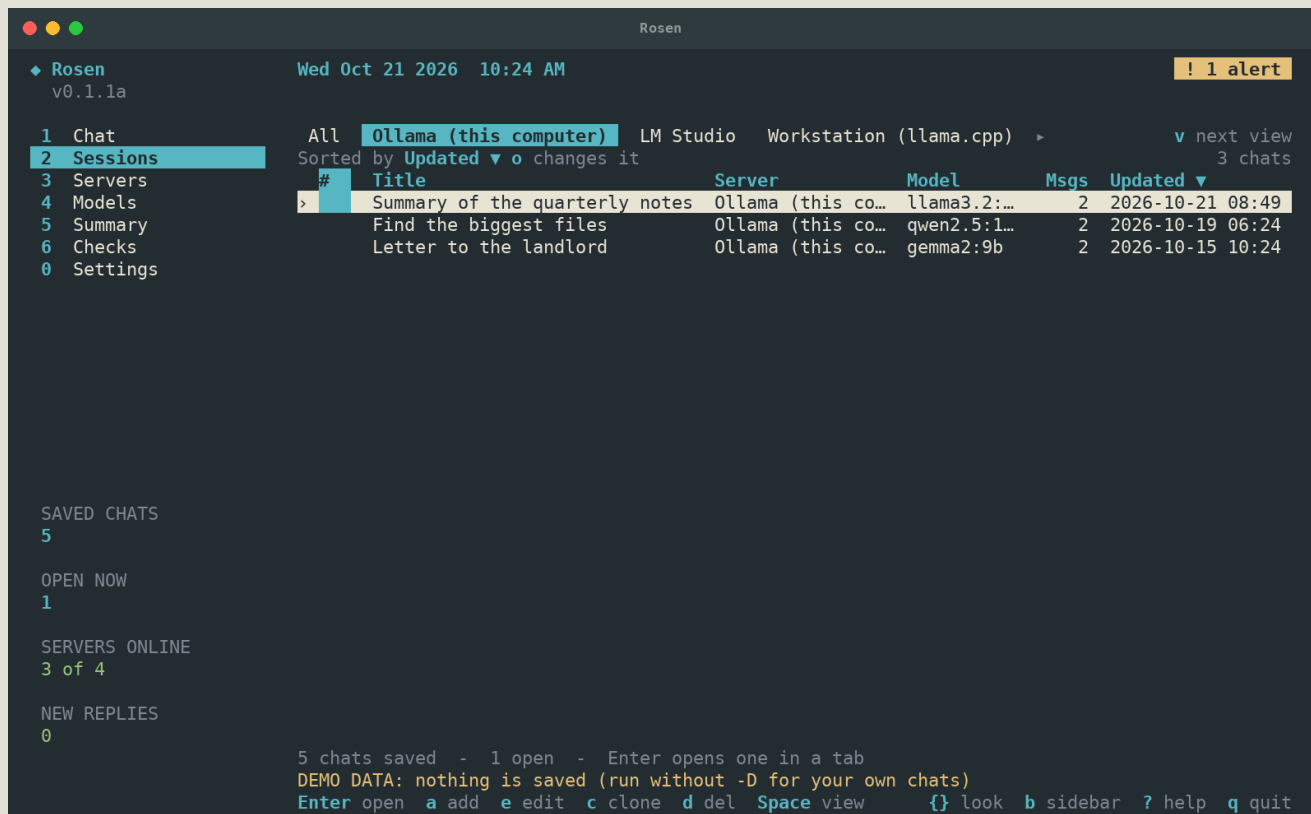


Figure 24. `v` has moved to the view of one server: three of the five chats use it.

A server's view is the quickest way to find "the chats I had with the big model".

6.3 Keys

Key	What it does
<code>Enter</code>	Open the chat in a tab, or switch to it if it is already open.
<code>a</code> , <code>n</code>	Start a new chat.
<code>e</code> , <code>t</code>	Edit the chat's details: title, server, model and instructions (section 5.6).
<code>c</code>	Clone the chat under a new title (below).
<code>Space</code>	Read the whole chat without opening it.
<code>d</code> , <code>Delete</code>	Delete the chat for good. Rosen asks first.
<code>x</code>	Export the highlighted chat (section 10.2).

Key	What it does
X	Export every saved chat into a folder.
i	Import a saved chat, or load a text or Markdown file into a new chat (section 10.4).
v , V	The next or previous view.
Up , Down , j , k , PgUp , PgDn	Move the highlight.
g , G , Home , End	The first and last row.
h , l	Move to the previous or next column.
o , s	Sort by the current column; press again to turn the order round.
O	Choose the column and the direction in a form.
r	Reverse the sort.
Alt-1 to Alt-9	Sort by that column.
/	Search every column.
f	Filter on the current column only.
Esc	Clear the search or filter.
w	Save everything now.

6.4 Searching, filtering and sorting

Press **/** and type. Every word you type must appear somewhere in the row (in the title, the server or the model; capital letters do not matter). The line above the table says what is being shown: Search: "rust" 1 of 5 shown. **Esc** clears it.

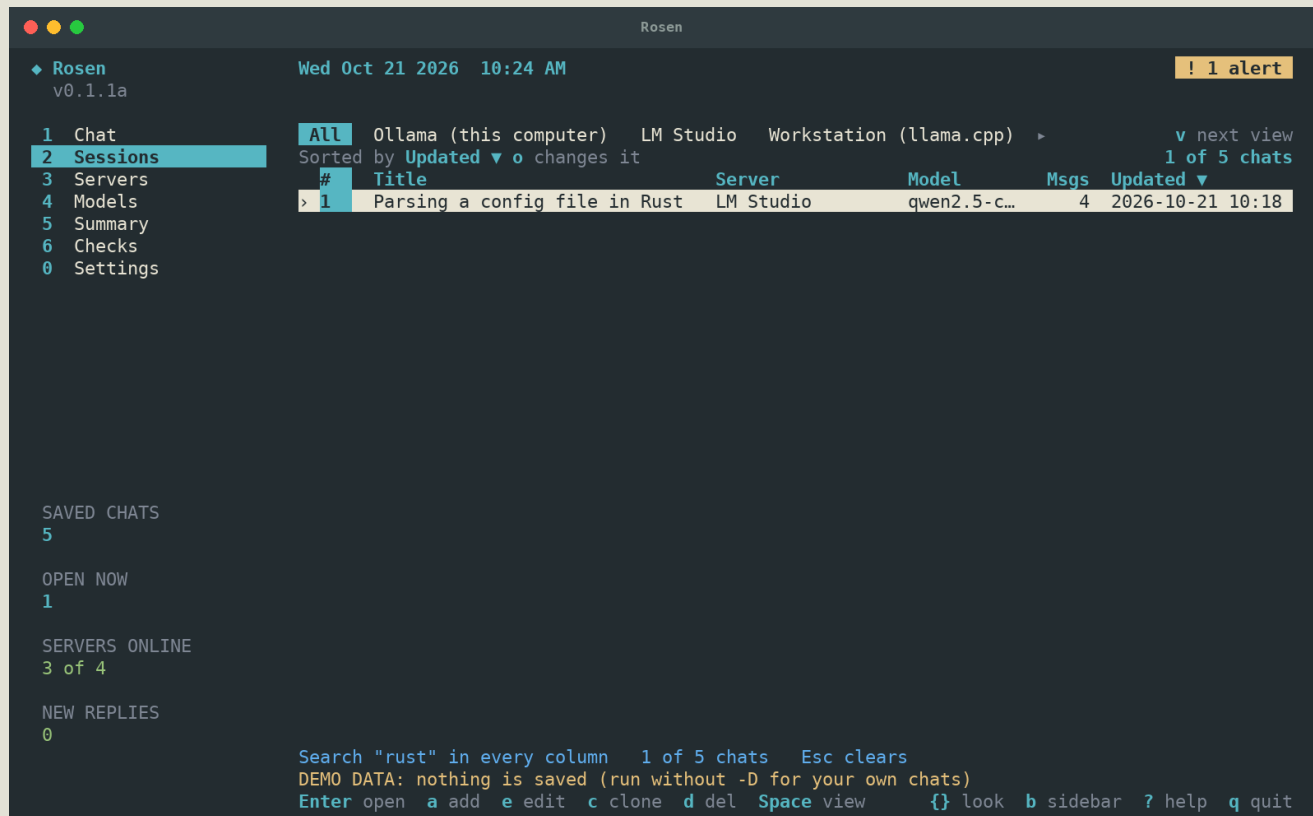
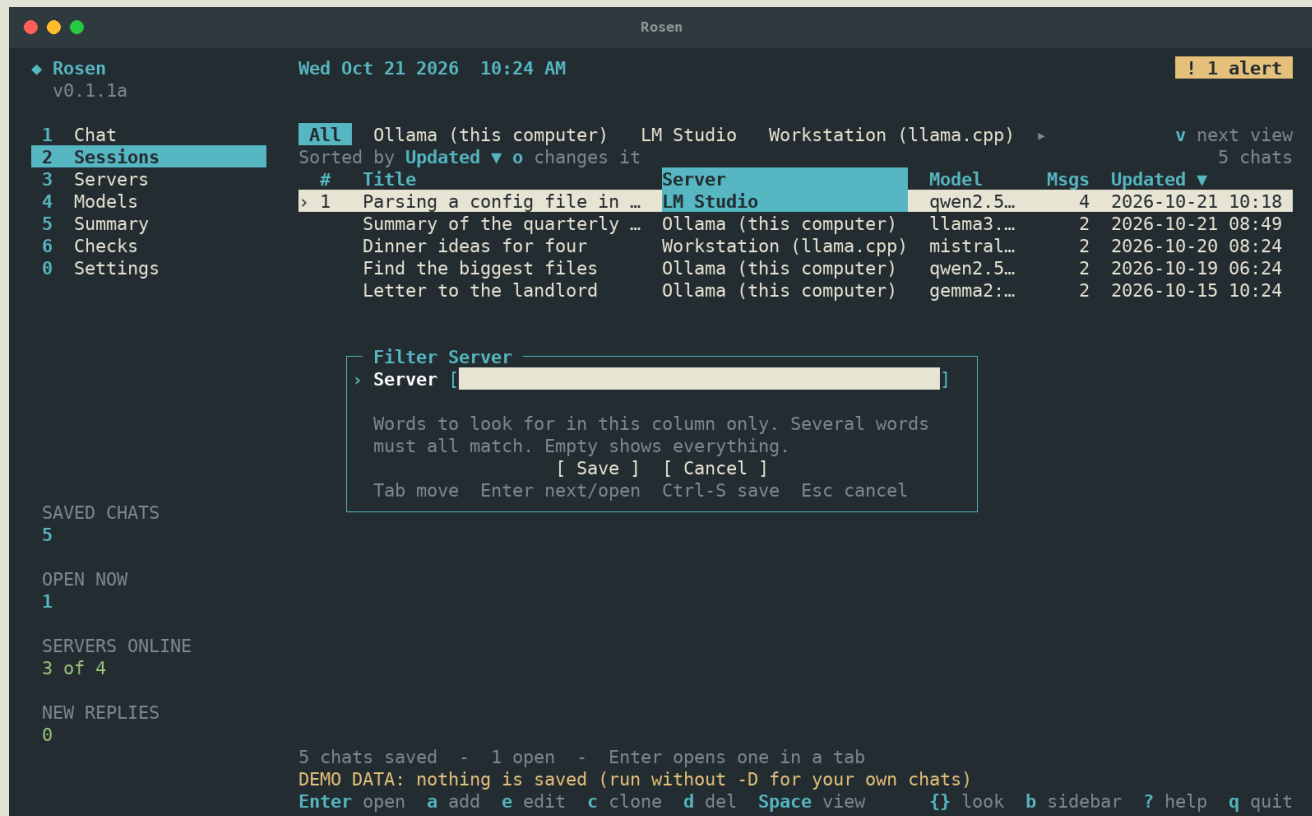
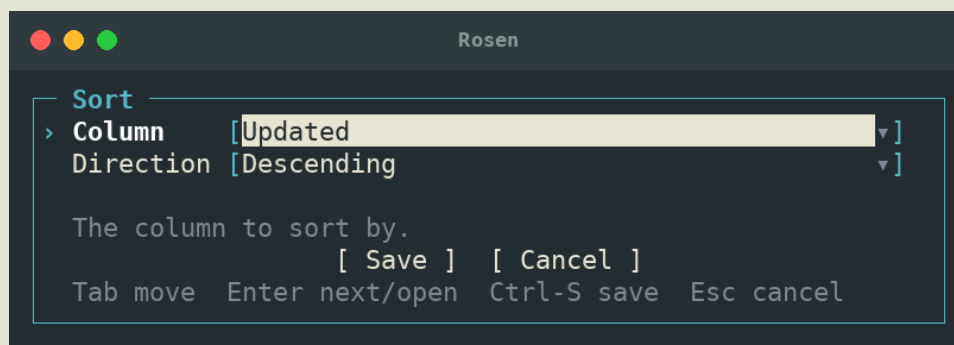


Figure 25. Searching for "rust": one chat of five matches.

f is the same for one column: move to the **Server** column with **l**, press **f**, and type LM, and only the chats whose server contains "LM" are shown. The form is headed with the column's name (Filter Server).

Figure 26. **f** on the Server column.

o sorts by the column the highlight is in, and **o** again turns the order round; **r** reverses it at any time; **Alt-1** to **Alt-9** sort by the first to ninth column directly. **o** opens a form where the column and the direction are fields. The sort is shown above the table: Sorted by Updated ▼ **o** changes it, and the heading of the sorted column carries the same mark. By default the newest chat is first. You can also click a heading to sort by it.

Figure 27. **o**: the column and the direction.

6.5 Reading, editing and cloning

Space shows the chat in a reader without opening it as a tab: its server and model, when it started and was last changed, how many messages, its instructions if it has any, and the whole conversation. **Esc** closes it.

e (or **t**) opens the chat's details form: **Title** (up to 120 characters), **Server**, **Model** and **Instructions**. The title shows on the tab, in this table, and in exports. Saving the form changes the chat and nothing else.

Figure 28. **e** : the details of the highlighted chat.

c clones the chat: a form asks for a **new title** (it starts as Copy of ...) and lets you change the server, model and instructions for the copy. Saving makes a second chat with the same messages, highlights it in the table (the page goes back to the All view if the view you were in would hide it) and says Cloned as "...". It is how you try the same conversation on another model without losing the first.

Figure 29. **c** : the clone starts with a title of its own.

6.6 Deleting

d deletes the chat file after you agree. This is the one place in Rosen where something is removed for good; there is no undo, and no recycle bin.

Careful. A deleted chat is gone. If you might want it later, export it first (**x**); an exported chat file can be imported again.

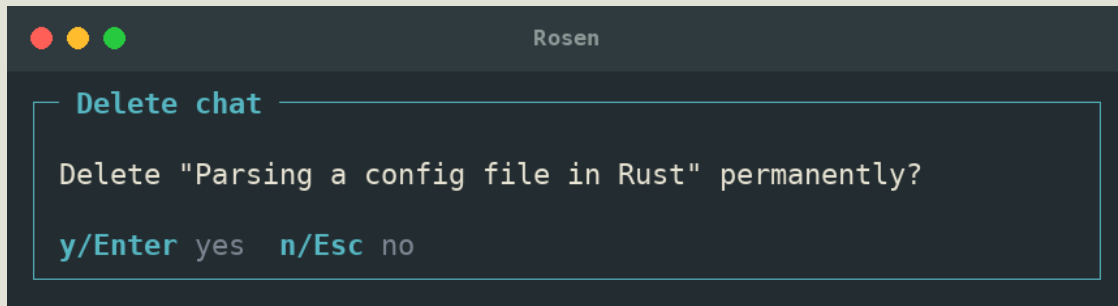


Figure 30. Deleting a chat names it and asks first. `n` or `Esc` leaves it alone.

6.7 Chats that cannot be read

If a chat file cannot be read (it was damaged, or was written by something else), Rosen does **not** hide it, repair it, or write over it. The Sessions page lists it, the Checks page says why (Chapter 12), and the file stays exactly as it was so you can look at it or restore it from a backup. Everything else keeps working.

7. Servers

A **server** is a program that runs a language model and answers over the network: Ollama, llama.cpp's llama-server, LM Studio, vLLM, LocalAI, or anything else that speaks the OpenAI chat API. It can be on this computer or on another one on your network. The Servers page (**3**) is the list of the ones Rosen knows about.

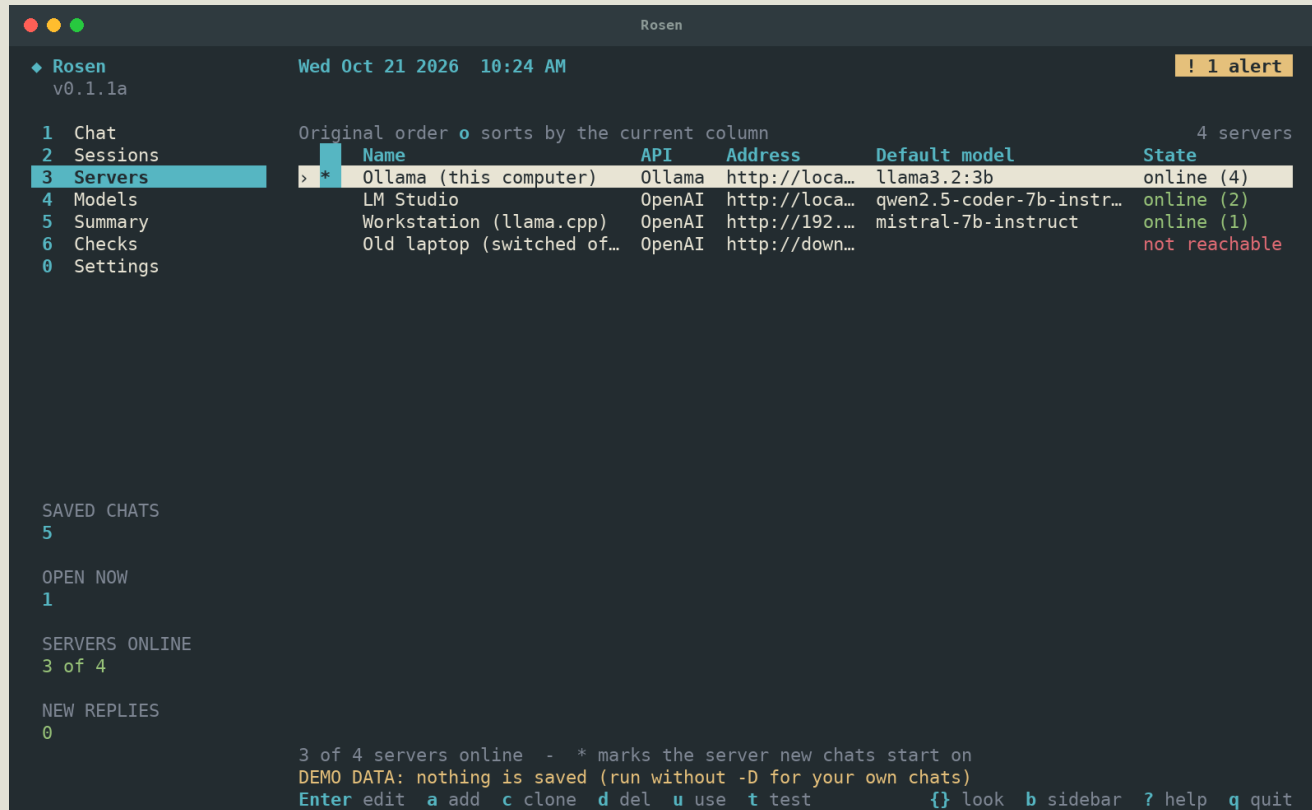


Figure 31. The Servers page. The * marks the server new chats start on. The State column fills in when a server has been tested or used.

7.1 The table

Column	What it shows
*	An asterisk marks the active server: the one new chats start on.
Name	The name you gave it.
API	Ollama or OpenAI: which of the two protocols it speaks.
Address	Where it is.
Default model	The model chats on this server use unless they have chosen their own.
State	not checked until Rosen has tried it; then online (N) where N is the number of models it listed, not reachable, or no model list (it answers but will not list its models).

7.2 Adding and editing a server

Press **a** to add a server and **e** (or **Enter**, or a second click on the row) to edit the highlighted one. Both open the same form (see Figure 13). **c** clones the highlighted server: the form opens with the same address, type, key and model, but with the name empty, because every server needs a name of its own (section 7.3).

Field	What to put in it
Name	Anything you like. It is shown in lists, in the chat header and in exports.
Address	Where the server is, such as <code>http://localhost:11434</code> or <code>http://192.168.1.20:8080</code> . <code>http://</code> is assumed if you leave it off. A trailing <code>/v1</code> or <code>/api</code> is removed (once), because people paste either. Use <code>https://</code> for a server behind TLS (section 7.7).
Type	Ollama , or OpenAI-compatible for llama.cpp, LM Studio, vLLM, LocalAI and most others. It decides which paths Rosen asks: <code>/api/tags</code> and <code>/api/chat</code> for Ollama, <code>/v1/models</code> and <code>/v1/chat/completions</code> for the other.
API key	Only if the server wants one. It is shown as dots, sent as a bearer token, and stored in your data file, which only you can read (section 17.1).
Default model	The model chats use on this server unless they choose their own. The Models page lists what the server offers (Chapter 8).
Check certificate	For <code>https://</code> servers: refuse a certificate that cannot be verified. Leave it on unless the server uses a self-signed certificate you trust.

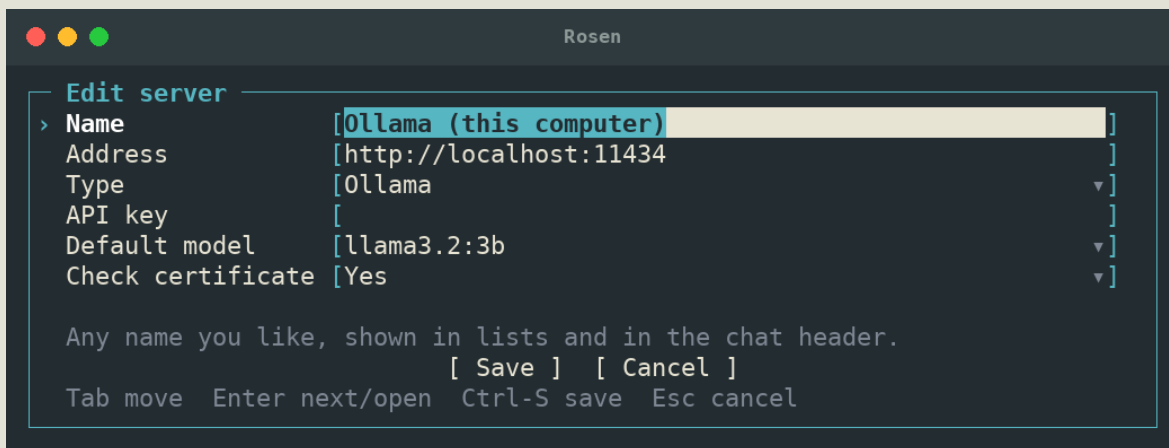


Figure 32. Editing a server. **Ctrl-T** tests the connection with what is typed, before you save.

Ctrl-T in the form tests the connection with what you have typed, without saving. **Ctrl-S** saves, as does the `[ Save ]` button; an empty name or an address that makes no sense is refused with a message under the field. If the Type is wrong, the message names the right one and where to change it.

7.3 Cloning and looking

c is the quick way to a second server that is nearly the same: a second model server on another port, the same server under another name, or one with a different default model. The form opens with the values of the highlighted server and an empty **Name**; saving without a name is refused (The server needs a name.).

Space shows everything that is known about the highlighted server in a reader: its name, address, type, default model, whether a key is set (never the key itself), whether the certificate is checked, its state, whether new chats start there, the last problem it had, and the models it listed.

7.4 Testing

Press **t** on a server (or **ctrl-T** in its form) to test it. Rosen asks the server for its models and then sends it one short question ("Reply with the single word OK.") with a limit of 16 tokens, and reports on the message line: LM Studio works: qwen2.5-coder-7b-instruct answered: OK, or what went wrong, in plain words: nothing is listening at that address, the connection was refused, the server answered with an error, the certificate could not be verified, curl is not installed.

R asks again for the list of models without sending a question. (It is a capital letter because **r** reverses the sort, as it does in Inventory Central.) The State column and the **Servers online** count in the sidebar are updated from these tries.

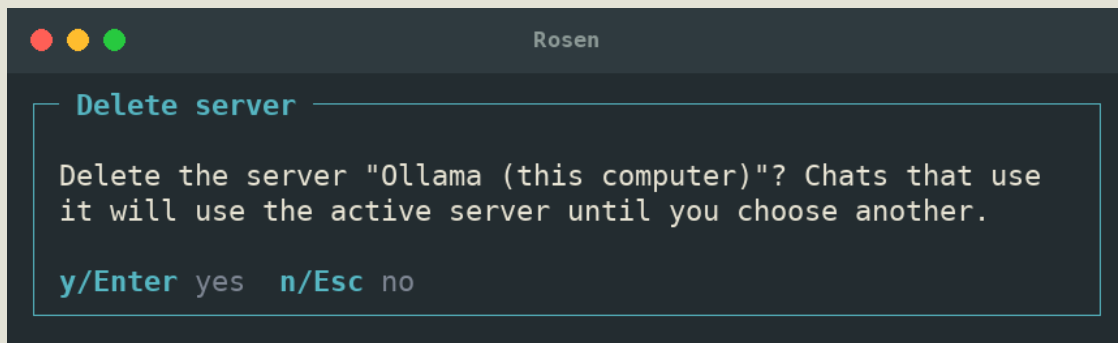


Figure 33. Deleting a server asks first. Chats that used it carry on with the active server until you choose another.

7.5 Choosing a server

u uses the highlighted server for the current chat, together with the server's default model, and, if that moves the chat to another server, makes it the **active** server too, so that new chats start there. **n** starts a new chat on it straight away. **d** deletes a server (the last one cannot be deleted); chats that used it keep working on the active server until you choose another.

7.6 Keys

Key	What it does
Enter , e	Edit the server.
a	Add a server.
c	Clone the server (the name starts empty).
u	Use this server for the current chat.
n	Start a new chat on this server.
t	Test the connection and list its models.
R	Ask the server for its models again.
Space	What is known about the server.

Key	What it does
d , Delete	Delete the server (asks first).
Up , Down , j , k , PgUp , PgDn , g , G	Move.
h , l	Move to the previous or next column.
o , s , O , r , Alt-1 to Alt-9	Sort by the current column, choose how to sort, reverse, or sort by a column number.
/ , f , Esc	Search every column, filter on the current one, and clear.

7.7 Servers on https://

Addresses that start with `http://` are reached directly by Rosen. Addresses that start with `https://` are reached through the system's **curl** program, because the Rust standard library has no TLS and Rosen has no dependencies. This is a declared extension of Rosen's standards (Appendix C), and it has consequences you should know about:

- **curl must be installed.** If it is not, Rosen says so (HTTPS addresses need the 'curl' command) and the Checks page lists the servers affected. `http://` servers still work.
- **Your key never appears in a process list.** Rosen hands curl its settings on standard input, not on the command line, and curl is started with `-q` so a `.curlrc` of yours cannot change anything.
- **Certificates are checked** unless you turn off **Check certificate** for that server. A server with a self-signed certificate is better fixed by trusting its certificate on your system; turning the check off is for test servers, and the Checks page will keep reminding you.
- **A key over plain http is readable by others on the network.** The Checks page flags a server that has an API key but an `http://` address that is not on this computer. (Servers on `localhost` are not flagged.)

7.8 Moving servers between computers

Servers live in your data file (section 17.1). To use the same list on another computer, copy the file; it holds the keys too, so keep it private.

8. Models

The Models page (**4**) lists the models offered by one server at a time. Its **views** are the servers: the page opens on the view of the server the current chat uses, and **v** and **v** (or a click on a name in the view bar) step to the others.

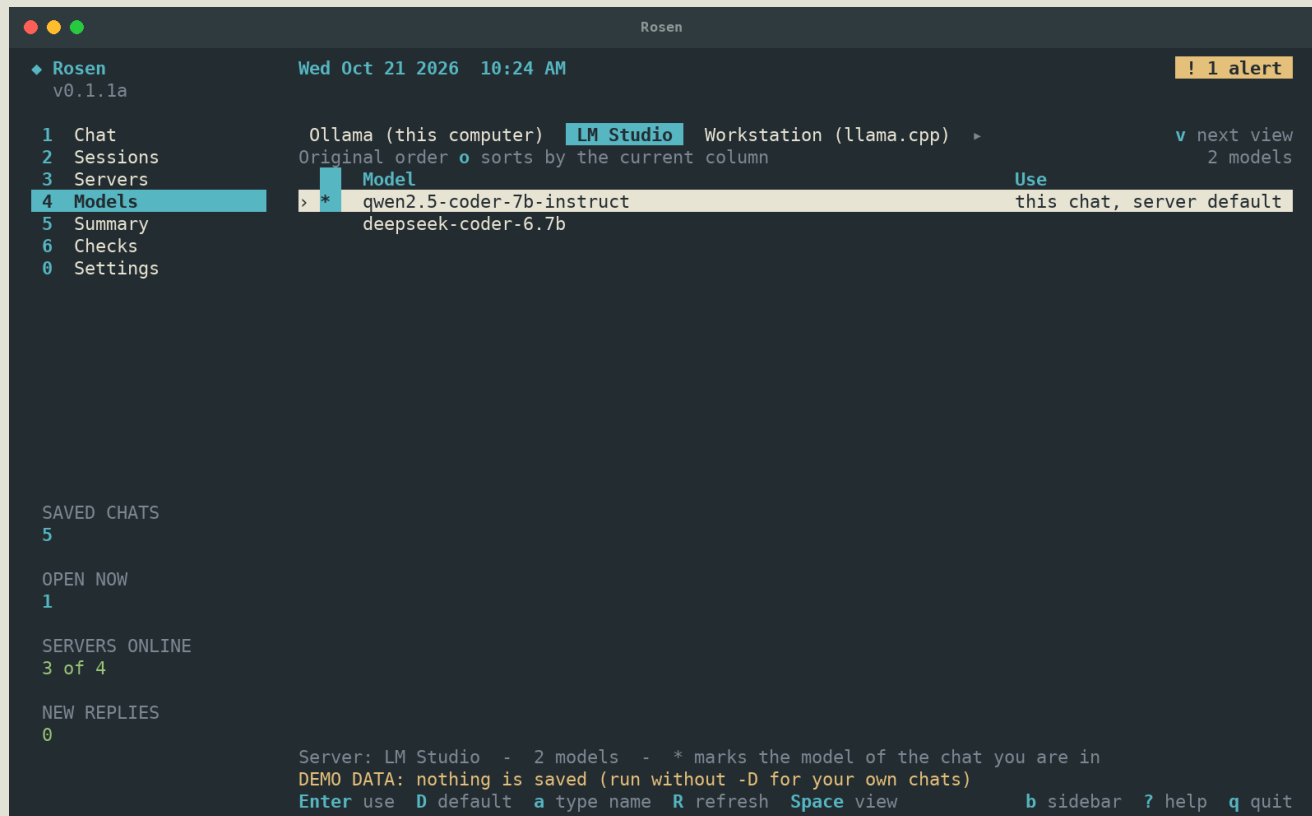


Figure 34. The Models page for a chat on LM Studio. The * marks the model the chat is using now; the Use column says why.

8.1 The list

When you open the page Rosen asks the server for its models, unless it already has an answer. The table has two columns: the model's name, and **Use**, which says where it is in use: `this chat`, `server default`, or both (`this chat, server default`). A model the chat is using that the server did not list (for example one you typed in) is shown dimmed.

If the server cannot be reached the page says so, and so does the State column on the Servers page. When the list arrives while you are looking at the page, the row you had highlighted stays highlighted even if the new rows move around it. If it answers but has no list of models (some OpenAI-style servers do not), the list is empty and you can type the model's name instead (below).

8.2 Choosing

Key	What it does
Enter , e , u	Use this model for the current chat only (on this view's server).
D	Make it the default model of the server, for chats that have not chosen their own.
a	Type a model name the server did not list.
R	Ask the server for its models again.
Space	What is known about the model: its server, whether the server listed it, whether it is the default and whether this chat uses it.
v , V	The next or previous server's view.
/ , f , Esc	Search every column, filter on the current one, and clear.
o , s , O , r , Alt-1 to Alt-3	Sort by the current column, choose how to sort, reverse, or sort by a column number.
h , l	Move between columns.
Up , Down , PgUp , PgDn , g , G	Move.

Using a model on a view of a different server than the chat's moves the chat to that server as well, so the chat and its model always agree.

The same choice is available without leaving the chat: **e** on the Chat page opens the chat's details, with the server and the model (section 5.6).

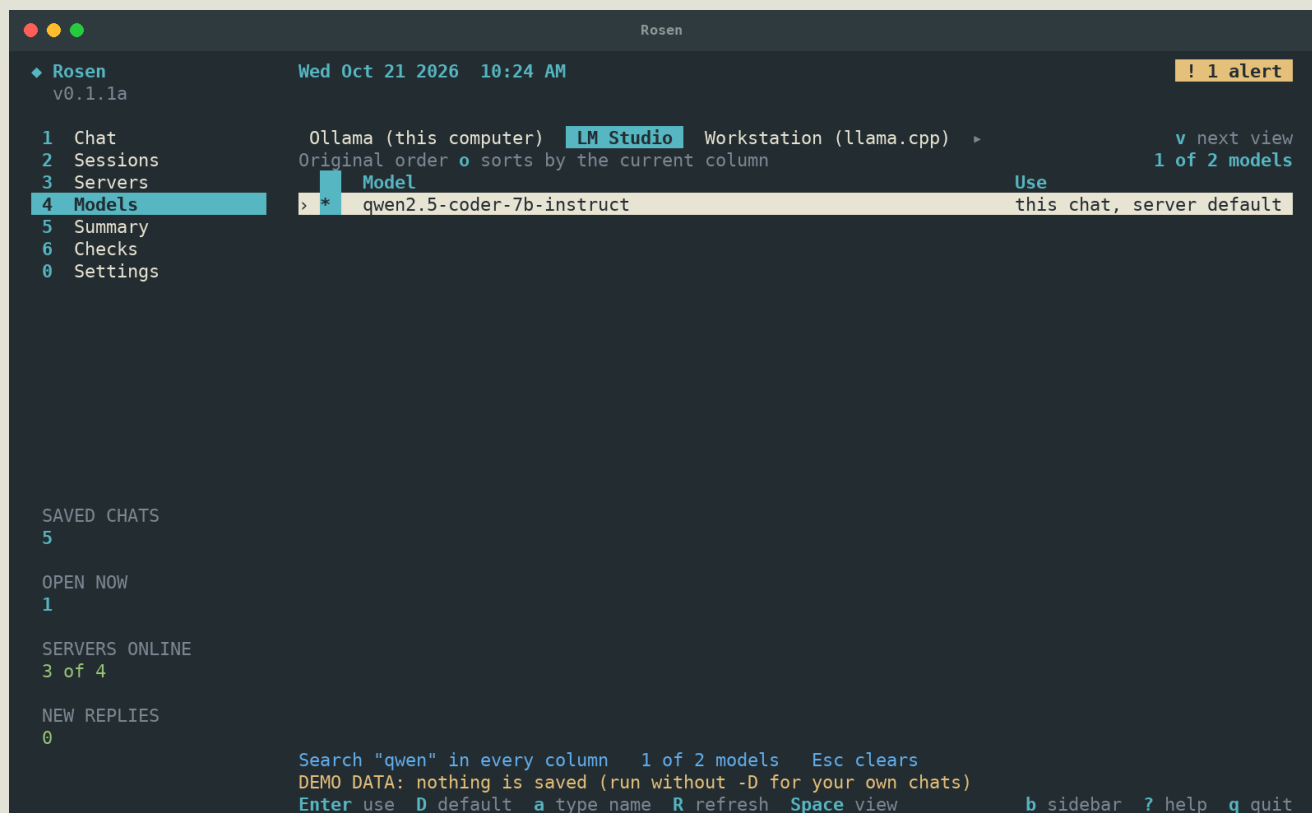


Figure 35. Narrowing a long list: press **/** and type part of a name.

8.3 Typing a model name

Press **a** to type the name of a model the server did not list: for a server that cannot list its models, or one whose model you know is installed but not in the list. Type it exactly as the server knows it (llama3.2:3b, qwen2.5-coder-7b-instruct) and press **Enter**. The chat uses it from then on.

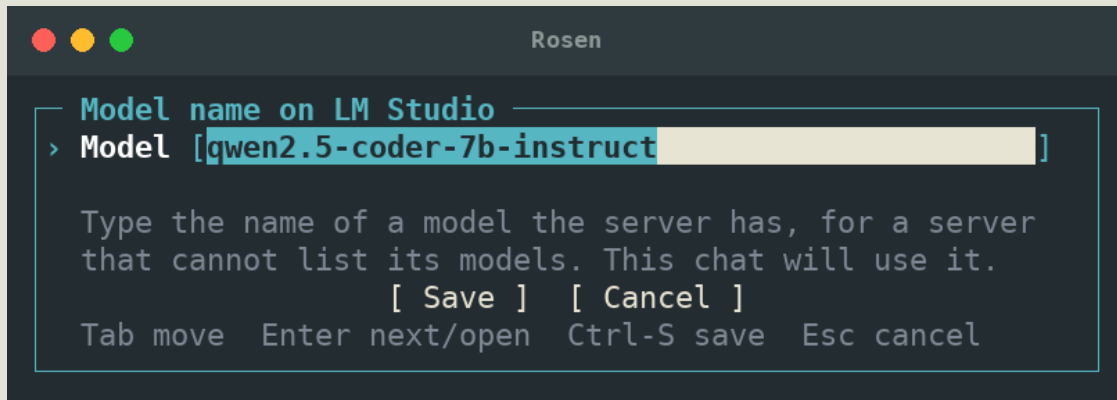


Figure 36. Typing a model name.

If the name is wrong, the server's own error comes back on the next message, shown in plain words, and the chat is not harmed.

8.4 Several models, several servers

Nothing ties a chat to one model for life. You can ask a small, quick model one question, switch to a larger one in the middle of the conversation (the history goes with you, up to the Memory setting), and open another chat on a different server altogether. Each chat remembers its own choice (Chapter 5), and the Summary page counts which models answered how often (Chapter 11).

9. Code and the Shell

Models write code, and then you want to do something with it. Rosen finds the code blocks in a chat for you and lets you read, copy, save or run them, and it can open a shell so you can work in a terminal without leaving Rosen.

9.1 Code blocks

A **code block** is a part of a reply the model fenced with three backticks, optionally followed by a language name (`rust`, `python`, `sh`...). In the conversation each one is drawn on a background of its own and labelled with its number and language: `code 2 · rust`. Numbers count through the whole chat, oldest first.

9.2 The code menu

Press `c` on the Chat page (or type `/code`) to list the code blocks in the chat.

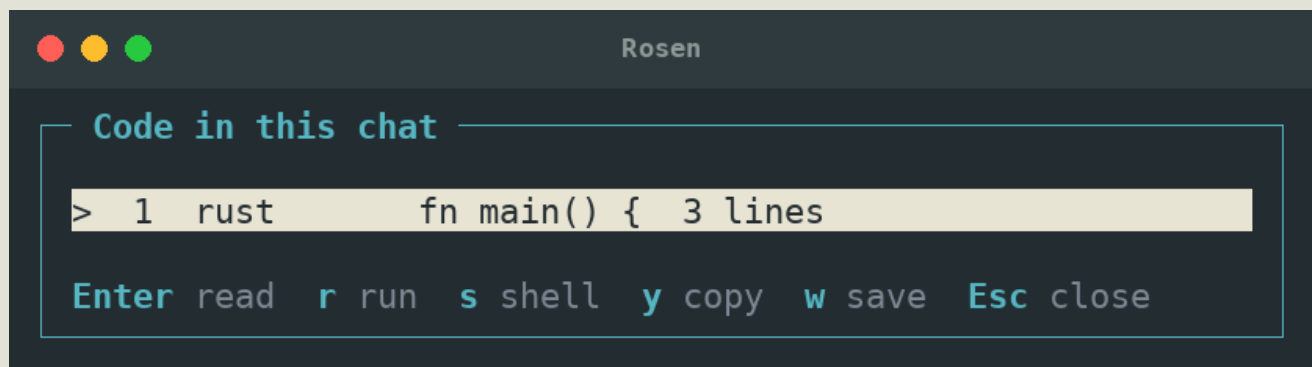


Figure 37. The code blocks in this chat: number, language, the first line, and its length.

Key	What it does
<code>Up</code> , <code>Down</code> , <code>1</code> to <code>9</code>	Pick a block.
<code>Enter</code>	Read it in a viewer.
<code>y</code>	Copy it to the clipboard.
<code>w</code>	Save it to a file (you choose the folder and name).
<code>r</code>	Run it and show what it prints (asks first).
<code>s</code>	Run it in a shell, where you can watch and type (asks first).
<code>Esc</code>	Close the menu.



Figure 38. Reading a block. The keys along the bottom are the same actions.

Saving asks for a file name; the suggestion is `code-N.ext` in the folder you used last, with an extension that fits the language (`rs`, `py`, `sh`...). If the file already exists Rosen asks before replacing it, and a new file is made private (readable only by you; an existing file keeps its mode).

9.3 Running code

Rosen never runs anything by itself. When you press **r** or **s**, it shows what it is about to do and waits for a yes:

> This code was written by the model, not by you. It will run on this computer with your permissions (python3, 12 lines), and you will see what it prints. Read it first (Esc, then c). Run it?

Take that seriously. Code from a model can be wrong, and code that is wrong can delete files. Read it first.

Careful. A program run from Rosen has **your** rights: it can read, change and delete anything you can. Rosen limits how long it runs and how much it prints, but it does not put it in a sandbox.

r runs the code and shows the result in a viewer. The code is written to a private temporary file and run by the right program:

Language	Run with
sh, shell, posix, console, terminal, or none given	sh (no language means sh, and the question says so)
bash	bash if it is installed, otherwise sh
zsh	zsh, if installed
python, python3, py	python3, or python
node, javascript, js	node, if installed
perl, ruby (rb), awk	the program of that name, if installed

Any other language (Rust, C, SQL...) cannot be run by Rosen; it says so and suggests copying or saving the code instead. The run:

- gets **no input** (standard input is closed) and its errors are merged into its output;
- is stopped after **60 seconds**, and the viewer says `stopped after 60 seconds`;
- keeps at most **200,000 bytes** of output and says when it cut it;
- is not waited for once it has finished: if it started something that carries on in the background (`sleep 60 &`), what it printed is shown within a second and the background program is left running;
- shows the program's exit status in the viewer's title: `Output: finished, exit status 0`.

`/run [n]` does the same from the message box (`/run` alone runs the newest block). Only one program runs at a time.

9.4 The shell

Press **!** (or **F2**, or type `/shell`) on any page to **open a shell**. Rosen hands the terminal to your shell (the program in `$SHELL`, or `sh`), and when you type `exit` it takes the terminal back, redraws everything and says `Back from the shell`. Nothing in Rosen changes while you are away, except that a reply that is still arriving carries on in the background.

Two environment variables are set for the shell:

- `ROSEN` holds the version, `v0.1.1a`.
- `ROSEN_CODE` holds the path of a file with the **newest code block** of the chat, written for you. So, after asking for a script, press **!** and then:

```
$ cat "$ROSEN_CODE"          # read it
$ sh "$ROSEN_CODE"           # run it
$ vi "$ROSEN_CODE"           # edit it
$ cp "$ROSEN_CODE" ~/bin/tidy # keep it
```

The shell tells you this when it starts. From the code menu, **s** opens the shell **and runs that block** in it (after asking), so you can watch it and answer its questions; when it ends, Rosen says what its exit status was and waits for **Enter** before taking the screen back.

`/! command` (or `/shell command`) runs one command in the shell the same way and waits for **Enter** when it ends.

Tip. The shell is a real shell on a real terminal: full-screen programs such as editors, `less` and `top` all work, because Rosen steps out of the way completely while it runs.

9.5 What is kept

The temporary files go in a private folder (mode 0700) under `$TMPDIR` (or `/tmp`) named `rosen-` and a number, and are removed when Rosen exits. Nothing you save with **w** is removed: that file is yours.

10. Export, Import and Attach

Rosen reads and writes your own files in three ways: it **exports** a chat as a document, it **imports** a chat or a text file, and it **attaches** text files to a message so the model can read them. All three use the same file browser, and none of them touches a file unless you pick it.

10.1 The file browser

Wherever Rosen needs a file or a folder it opens its own browser, so you never have to remember a path.

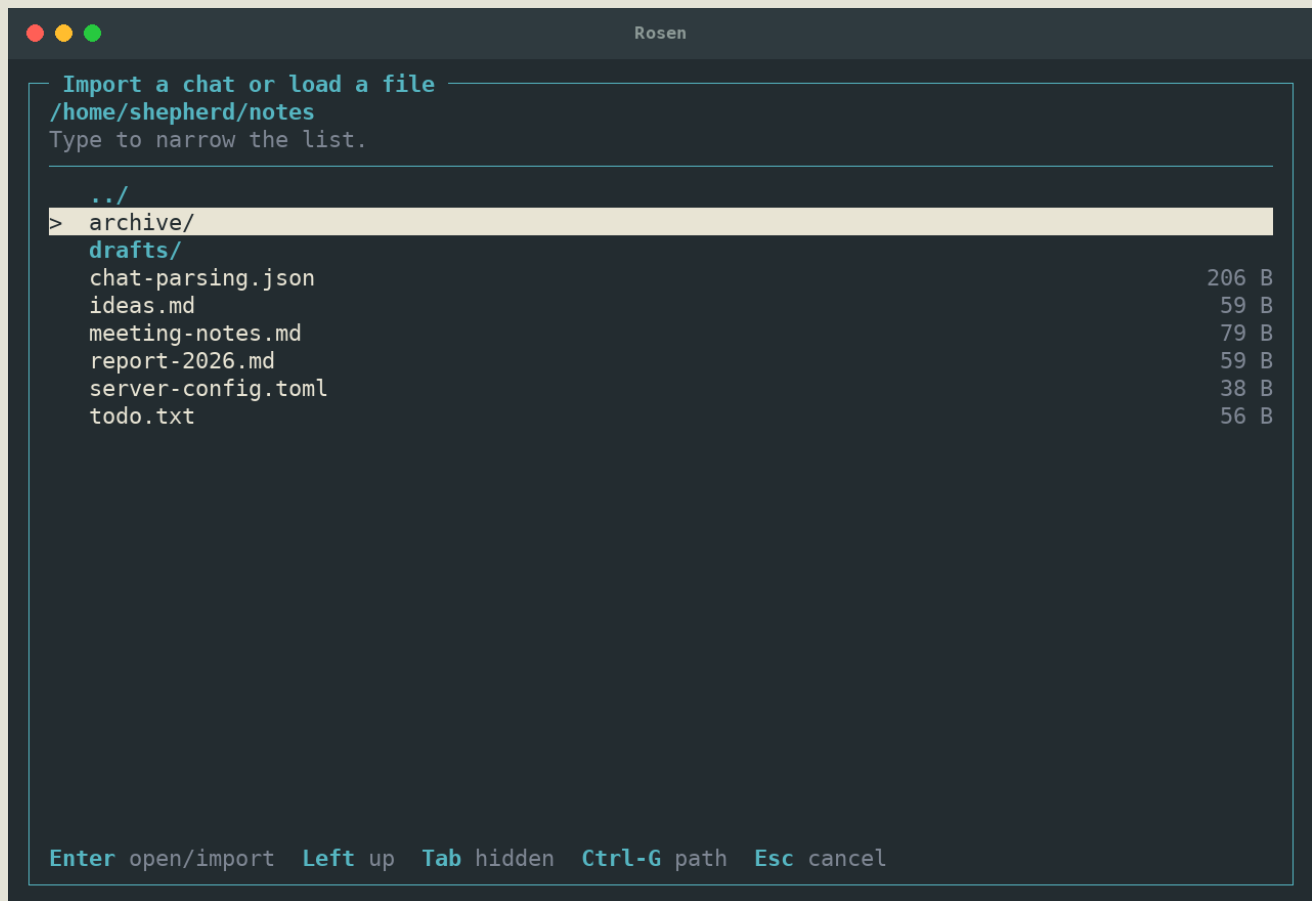


Figure 39. The file browser, here choosing a file to import. The folder is shown at the top; folders come first, then files with their sizes.

Key	What it does
Up , Down , PgUp , PgDn , Home , End	Move the highlight.
Enter	Open a folder, or pick the file (or all the marked files).
Right	Open the highlighted folder.
Left , Backspace	Go up one folder. (Backspace first removes typed letters.)
Space	Mark or unmark a file, where several can be picked.

Key	What it does
<i>letters</i>	Narrow the list to names that contain them. Ctrl-U or Esc clears the letters.
Tab	Show or hide names that start with a dot.
Ctrl-G	Type a path to go to.
Ctrl-S	When choosing a folder: use the folder you are in.
Esc , Ctrl-C	Cancel.

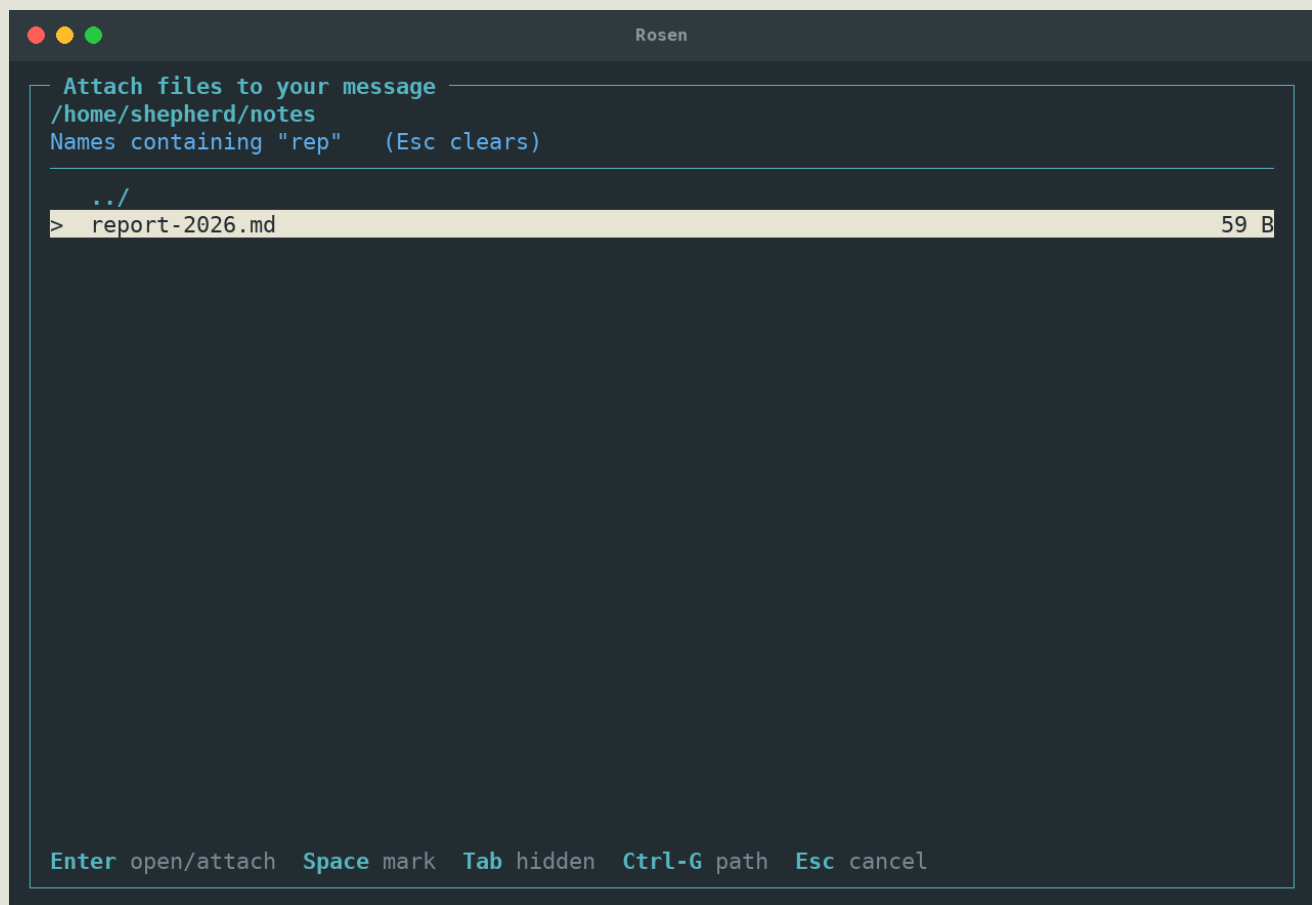


Figure 40. Typing narrows the list: the line under the folder says Names containing "rep".

The browser starts in the folder you used last, or your home folder. It only ever lists names and sizes; it reads a file only when you pick it, and only as text. It refuses a file that is a binary, is not valid UTF-8, or is larger than the limit for what you are doing (below). Names with control characters are shown with question marks, so that a hostile file name cannot draw on your screen.

10.2 Exporting a chat

Press **x** on the Chat page or on the Sessions page (or type `/export`). A form asks for three things:

Field	What it is
Format	Markdown (.md), HTML page (.html), Plain text (.txt), or Chat file (.json), the only one Rosen can import again losslessly.

Field	What it is
Folder	Where the file goes. Ctrl-B opens the browser to choose one. It starts at the folder you used last.
File name	The chat's title, made safe for a file name, without the extension. The extension follows the format, so a Markdown export can never end up named <code>.html</code> .

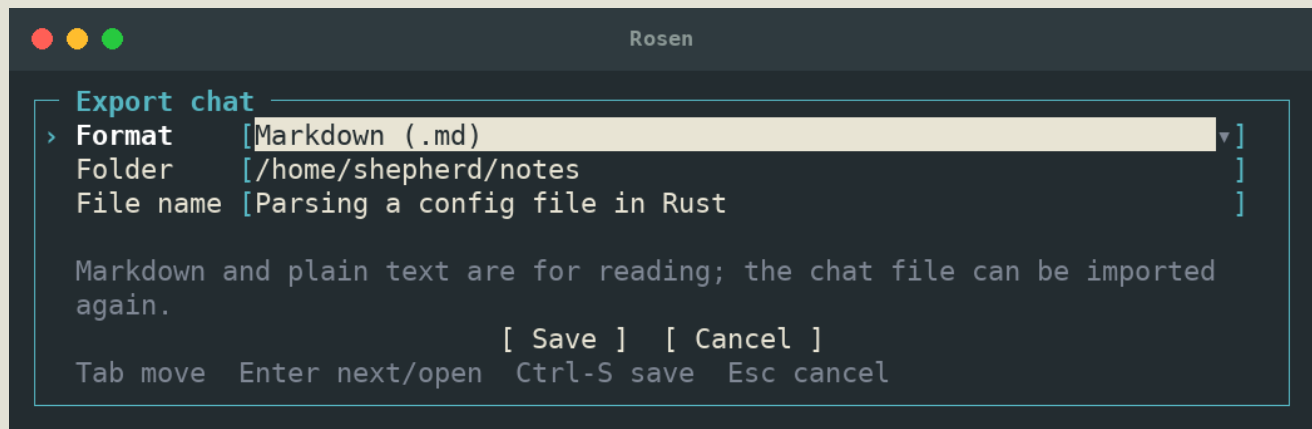


Figure 41. Exporting a chat. The format menu is opened with **Enter** ; **Ctrl-B** browses for the folder.

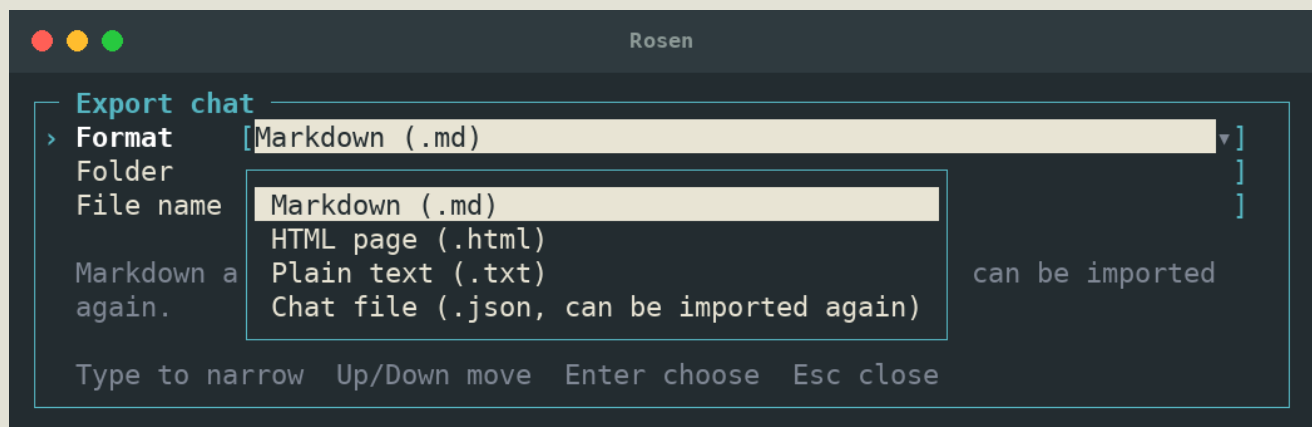


Figure 42. The four formats.

When you save, the file is written and the message line says what happened: wrote `/home/you/exports/greeting.md (380 bytes)`.

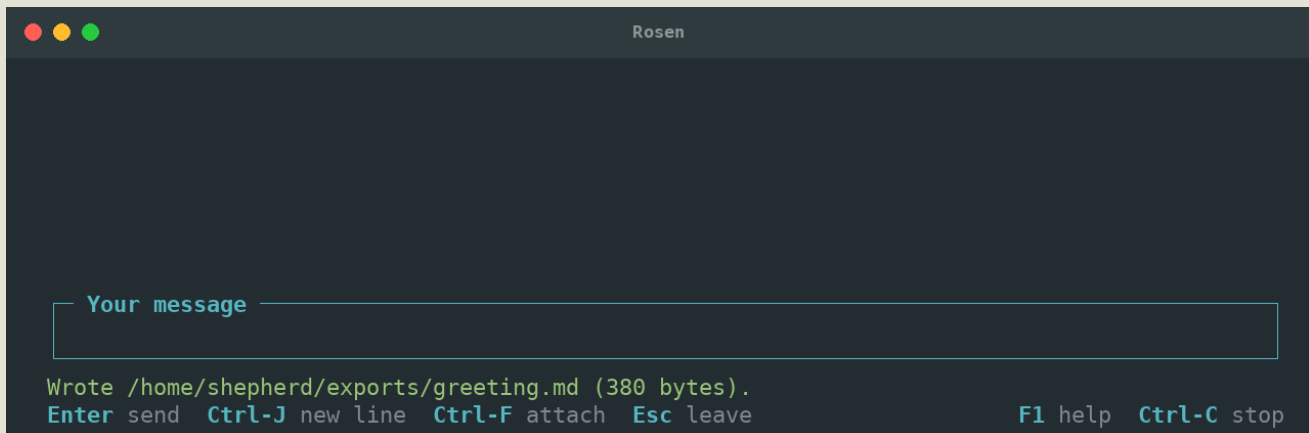


Figure 43. After exporting.

If a file with that name **already exists**, Rosen asks before replacing it, and nothing is touched until you say yes. The one exception is Rosen's own files (the servers and keys, the chats, the settings and their backups): those are never replaced by an export, even if you say yes, and the message says to choose another name. "Export all" likewise refuses the folder where the chats are kept.

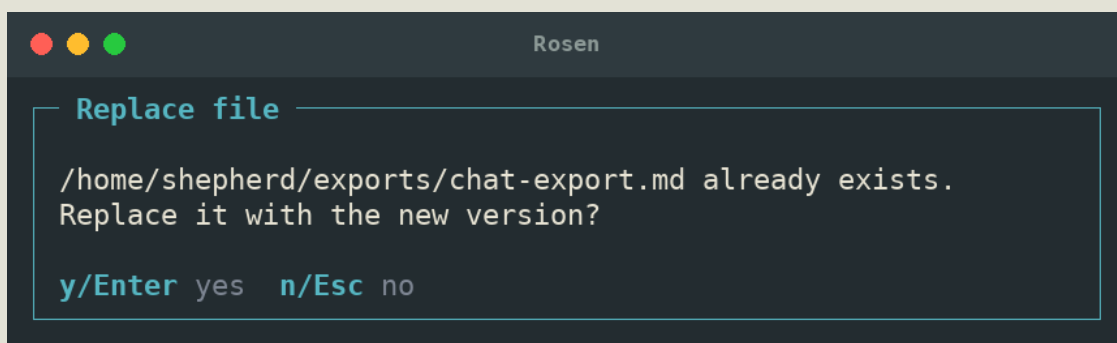


Figure 44. Exporting over an existing file asks first.

A Markdown export looks like this: the title, the date, and then each message under its speaker, with the model and server that answered and the time.

```
# Show me a greeting
Exported 2026-10-21T06:29:54 from Rosen
## You
*2026-10-21T06:29:54*
Show me a greeting
## Assistant (qwen2.5-coder-7b-instruct @ LM Studio)
*2026-10-21T06:29:54*
This is the built-in demo server ...
```

New files are created readable only by you (mode 0600); if you replace a file that exists, it keeps the mode it had.

In the message box, `/export` alone opens the form. `/export md` writes the chat as Markdown, named after its title, into the folder you used last. `/export md ~/notes/chat.md` writes that file; a name with no extension gets the right one, and `/export ~/notes/chat.html` takes the format from the extension. An existing file is asked about first, as always. The format may be `md`, `txt`, `html` or `json`.

10.3 Exporting everything

x on the Sessions page (or `/exportall`) exports **every** saved chat into one folder, one file per chat, in the format you choose, named from the title plus a short code so that two chats with the same title do not collide: Parsing a config file in Rust-a1b2c3.md. The folder is made if it is missing. The message line counts what was written, and if some failed it says how many and why. A chat that cannot be read is skipped and reported, never overwritten.

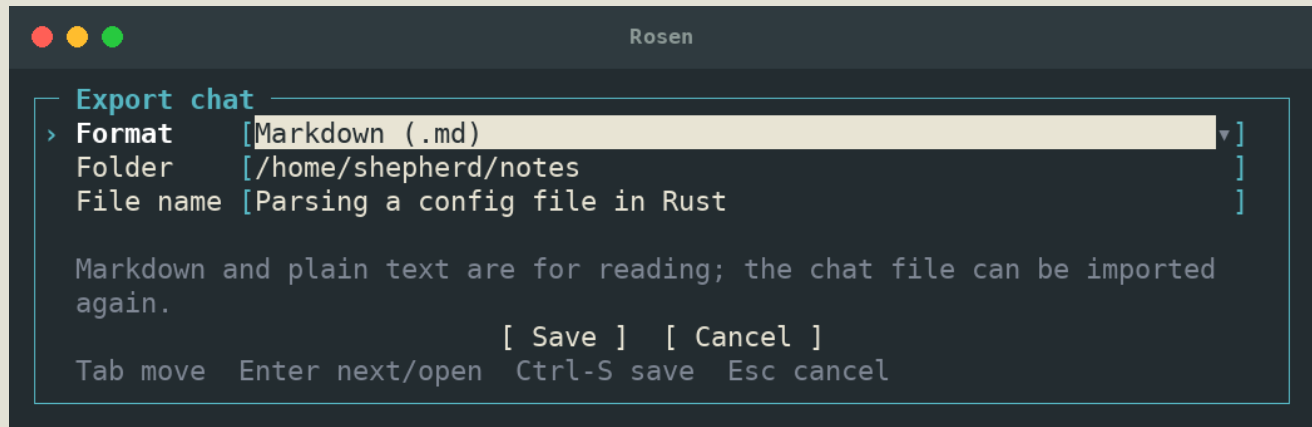


Figure 45. Exporting every saved chat. There is no file name here: each chat names its own file.

10.4 Importing

Press **i** on the Sessions page, or type `/import` (with a path, `/import ~/notes/chat.json`, to skip the browser). Rosen looks at what you picked:

- **A chat file** (.json exported by Rosen, or by llm-tui) becomes a **new chat** straight away: Imported "Parsing a config file in Rust" (2 messages) as a new chat; the file is untouched. Your original file is never changed.
- **Any other text file** (.md, .txt, source code, notes) asks what you want to do with it:

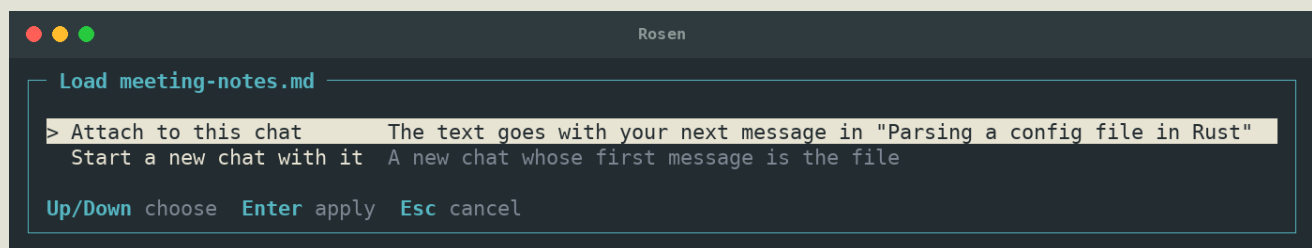


Figure 46. A text or Markdown file: attach it to the current chat, start a new chat with it, or, when it looks like a conversation that Rosen saved, continue it.

Choice	What happens
Attach to this chat	The text goes with your next message in the current chat (see the next section).
Start a new chat with it	A new chat is made whose first message is the file, ready for the model to answer.

Choice	What happens
Continue it as a conversation	Offered when the text is a Markdown or plain-text export Rosen recognises. The messages become a chat you can carry on from where it left off.

The same limits apply as everywhere: a chat file can be up to 20 MB; text that goes to the model as one file is limited to 200,000 bytes (roughly 35,000 words), because the model has to read all of it. The message says so, and tells you what to do, when a file is too big.

10.5 Attaching files to a message

Press **a** on the Chat page (or **ctrl-F** in the message box, or `/attach`) to choose files to send with your **next** message. In the browser, **Space** marks several files and **Enter** attaches the marked ones, or the highlighted one if none are marked.

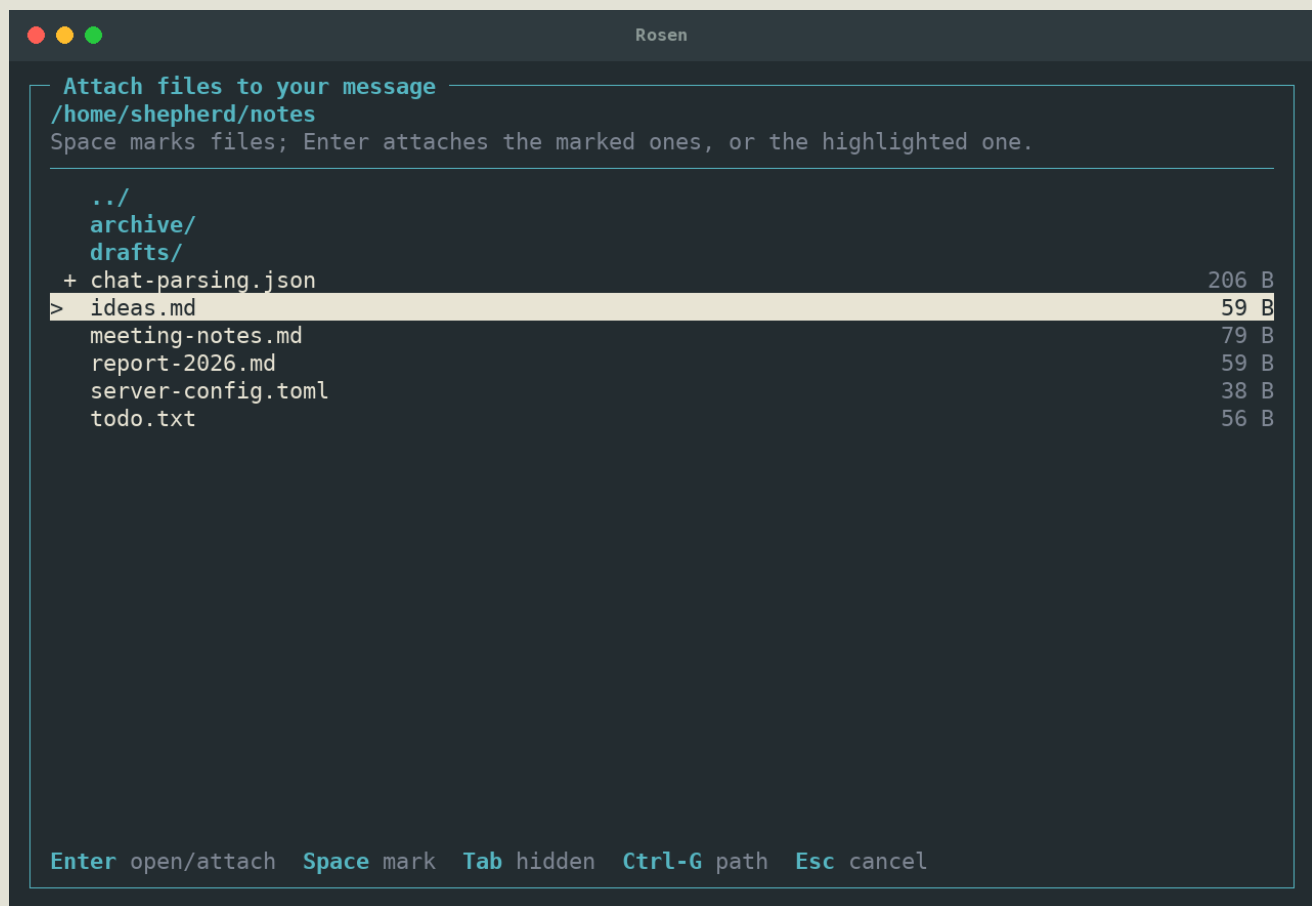


Figure 47. Marking files with **Space**; a + shows the marked ones. **Enter** attaches them.

An attached file is listed above the message box with its size, and **ctrl-X** removes the last one. When you send, each file is added to the message after your own words as `[File: name]` followed by the text of the file inside a code block, so the model sees where one file ends and the next begins.

Up to **10** files can go with one message, each up to **200,000 bytes**, and no more than **600,000 bytes** in all. The file is read once, when you attach it; changes made to it afterwards are not sent.

Tip. Text that is too long for one message is better split into pieces than cut down by hand: attach one piece, ask your question about it, and attach the next.

10.6 What is never done

- No file is read unless you pick it.
- No file is written unless you gave its name, and an existing one is replaced only after you agree.
- In `rosen -D` (sample data) no file is written at all, and the message line says so.

11. Summary

The Summary page (**5**) answers "what have I been doing with all this?" It reads your saved chats and counts them.

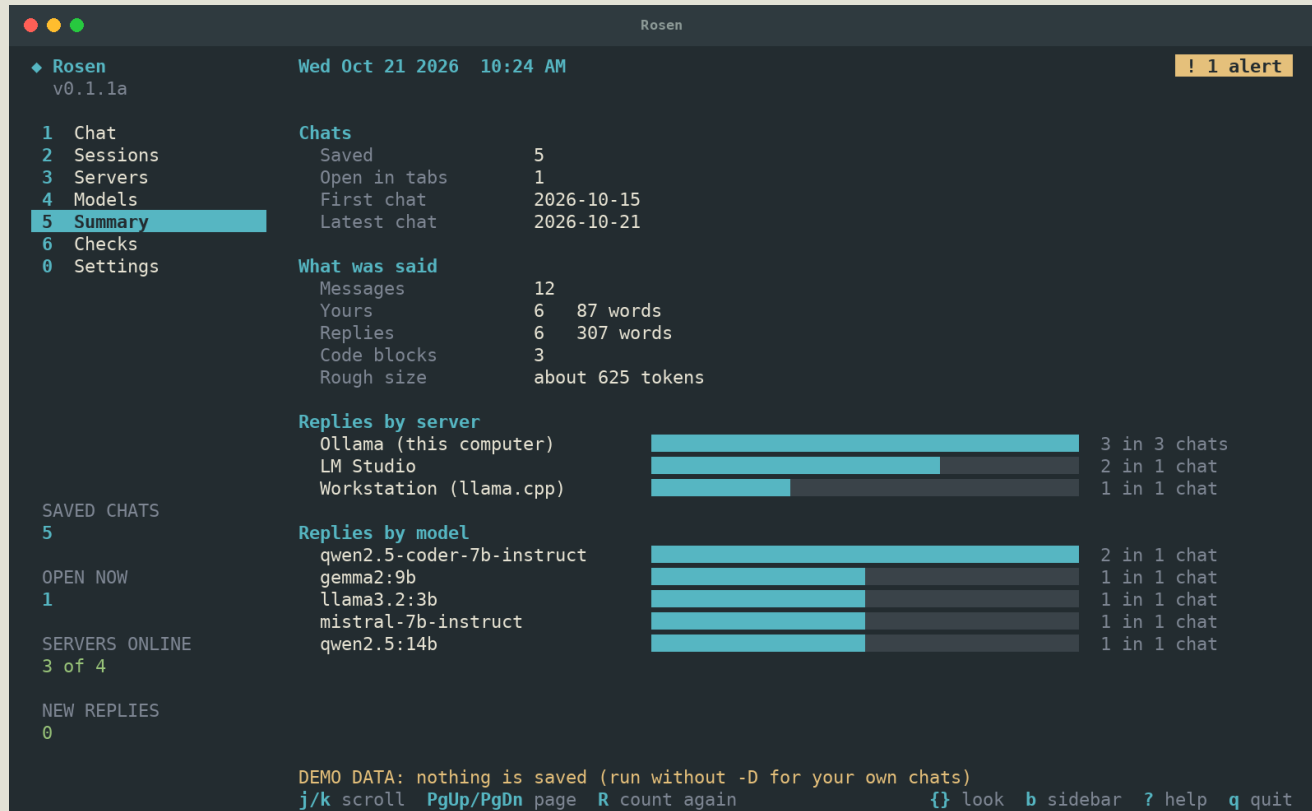


Figure 48. The Summary page for the sample chats. The bars are drawn from the number of replies.

11.1 What it counts

Section	What it shows
Chats	How many chats are saved, how many are open in tabs, the date of the first chat and of the latest one. If any chat file could not be read, it says how many; if there are more than 2,000 chats it counts the newest 2,000 and says it stopped early.
What was said	The number of messages; how many are yours and how many are replies, each with a word count; how many code blocks the chats hold; replies that were stopped before the end; and a rough size in tokens (about four characters to a token), which is only a guide to how much text has gone through.
Replies by server	A bar for each server that has answered, with the number of replies and in how many chats.
Replies by model	The same by model, so you can see which of your models you really use.

A server or model that is no longer in your list is still counted, by the name it had when it answered; replies from before the name was recorded are counted under (not recorded).

11.2 Keys

Key	What it does
Up , Down , j , k	Scroll.
PgUp , PgDn , Space	Scroll a page.
g , G	The top and the bottom.
R	Count again.

The numbers are worked out when you open the page, so they are current then; press **R** to bring them up to date while you are on it. Nothing here is stored: delete a chat and it stops being counted.

12. Checks

The Checks page (**6**) lists things worth knowing, each with a one-line reason and, where something can be done about it, the page to go to. Whenever one of them needs attention, the **alert badge** in the corner of the screen counts it (**! 1 alert**), so you do not have to open the page to find out.

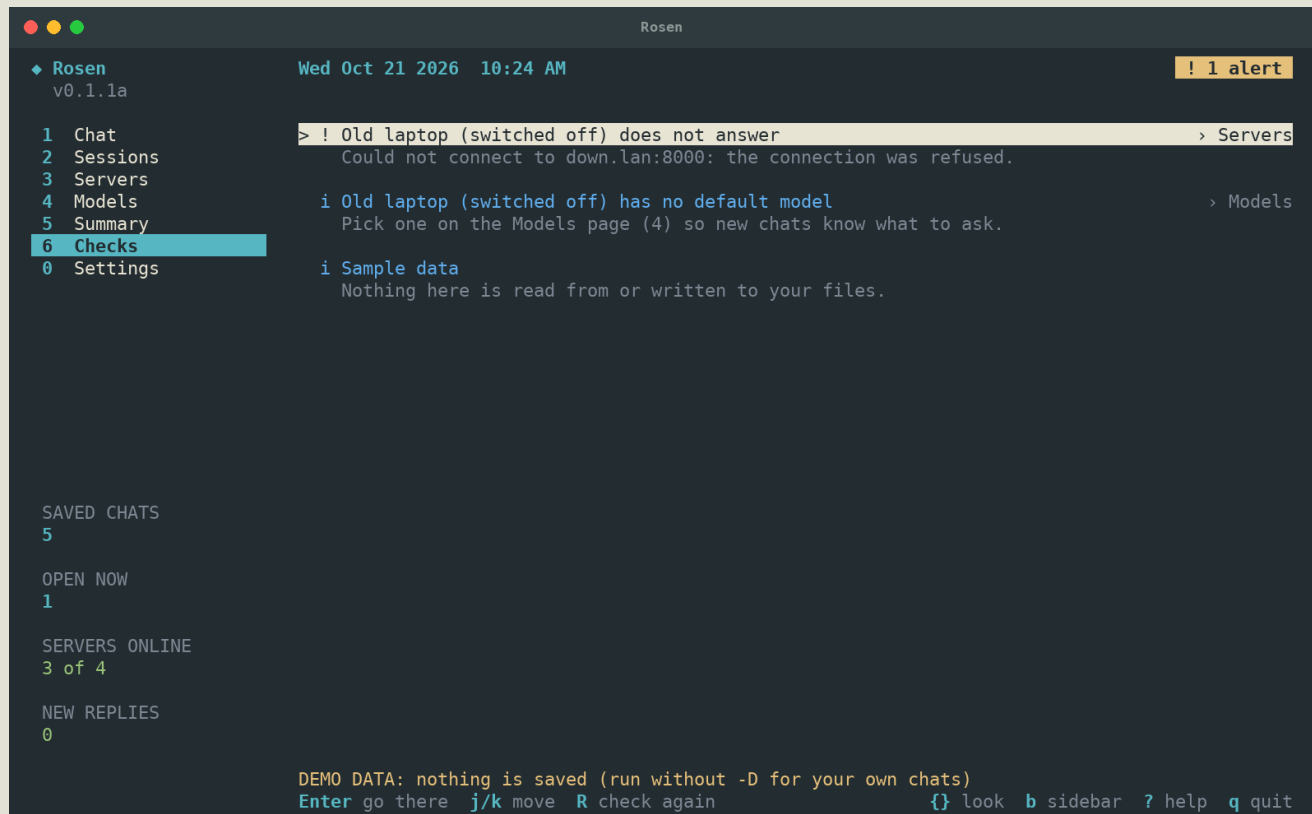


Figure 49. The Checks page. **!** marks something that needs attention, **i** is information. **Enter** goes to the page where it can be dealt with.

12.1 What it checks

Check	What it means
Your data file can be read by other people	The file with your servers and keys has a mode that lets others read it. The message gives the command that fixes it (<code>chmod 600 FILE</code>). When it is private, the page says Data file is private.
Your chats folder can be read by other people	The same for the folder of saved chats (<code>chmod 700</code>).
The backup of your settings can be read by other people	The <code>.bak</code> copy of the data file, which holds the same keys.
A saved chat cannot be read	Names the chat file and why. The file is left as it is and is never overwritten; fix or remove it yourself (section 6.7).

Check	What it means
NOT SAVED	Something could not be written (a full disk, a folder you cannot write to). The message says what. The entry stays, and a red banner on the message line repeats it, until a later save works.
An earlier run left a crash report	A file called <code>crash.log</code> next to your data file says what happened. Please send it if you can.
Screen settings file	Some lines of the file that holds the colors, clock and start-up choices could not be understood. Rosen skipped them and used the defaults for those settings; the message names the line. The file is rewritten, without those lines, the next time you change a screen setting.
NAME does not answer	A server that failed the last time it was tried, with the reason.
NAME has no default model	A server with no model chosen; pick one on the Models page.
NAME: the server's certificate is not checked	An <code>https://</code> server with Check certificate turned off.
NAME: the key is sent without encryption	A server with an API key that is reached over plain <code>http://</code> and is not on this computer.
https servers need the curl program	You have an <code>https://</code> server and curl is not installed (section 7.7).
Sample data	You are in <code>rosen -D</code> ; nothing is read from or written to your files.
No data file yet	A fresh start; the file is made the first time you change a server or a setting.

12.2 Keys

Key	What it does
Up , Down , j , k	Move.
Enter , e	Go to the page where the highlighted check can be dealt with (it is shown at the right of the row).
R	Check again. The page is also rechecked whenever you open it.

12.3 The alert badge

The badge counts the checks marked **!**: those that need attention, not the ones that are only information. It appears at the top right of every page in the sidebar layout, and in the top-bar layout on the line with the page names. When nothing needs attention, there is no badge.

Tip. A fresh Rosen, with your own data file in the right mode and servers that answer, shows no badge. If you see **! 1** alert after your first run, it is almost always a server that was not reachable when you tried it.

13. Settings

The Settings page (`0`) holds how Rosen looks and behaves, and the numbers it sends to the model. Every value has a sensible default, so you may never need to change one. Highlight a setting and press `Enter` (or `Space`, or `e`, or click the highlighted row); a small menu or form opens with the current value and an explanation, and the change takes effect and is saved as soon as you confirm it.

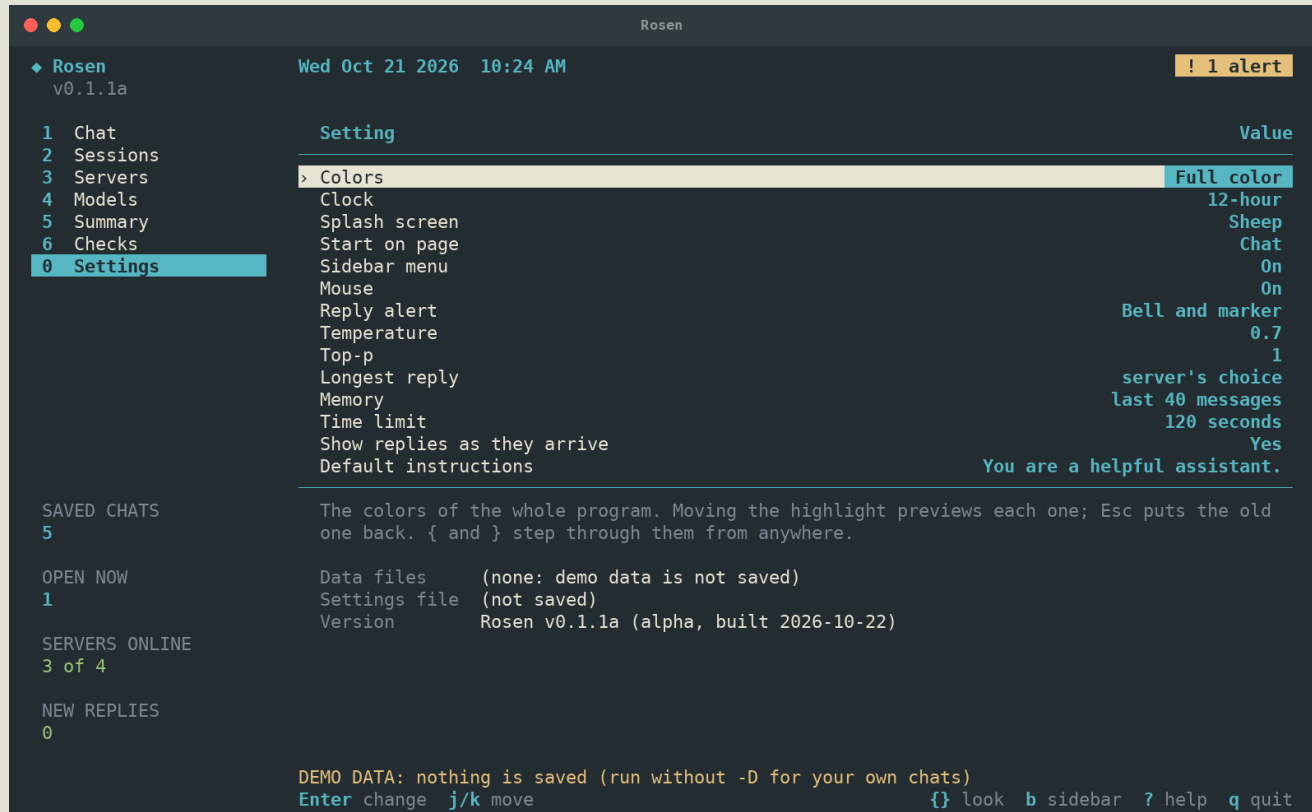


Figure 50. The Settings page. The list has a Setting column and a Value column; the lines under it explain the highlighted setting, name the data folder and the settings file, and give the version.

Settings are in two groups. **On the screen** settings are about your terminal and your habits, and are kept in a small file of their own (section 17.1). **Sent to the model** settings are kept with your servers and apply to every chat.

13.1 On the screen

Setting	Default	What it does
Colors	Full color	The colors of the whole program: fifteen looks (Chapter 14). Moving the highlight in the menu previews each one; <code>Esc</code> puts the old one back. <code>{</code> and <code>}</code> step through the looks from any page.
Clock	12-hour	The time shown beside the date at the top of the screen: Off , 12-hour or 24-hour .

Setting	Default	What it does
Splash screen	Sheep	The picture shown when Rosen starts: Sheep (the flock), Classic (the plain title box) or Off . The sheep needs a window 18 rows tall; a smaller one falls back to the box.
Start on page	Chat	The page shown after the splash screen: Chat, Sessions, Servers, Models, Summary, Checks or Settings. -p PAGE overrides it for one run.
Sidebar menu	On	On lists the pages at the left of a window 100 columns wide and 24 rows tall or more. Off keeps the page names in the top bar, in every size of window. b switches it from any page.
Mouse	On	On : clicks and the wheel work (Chapter 3). Off : Rosen never asks the terminal for the mouse, so the terminal's own selection works, and a click does nothing.
Reply alert	Bell and marker	What happens when a model finishes a reply while you are on another page or chat, or the window is behind another one: Bell and marker , Marker only , or Off (Chapter 15).

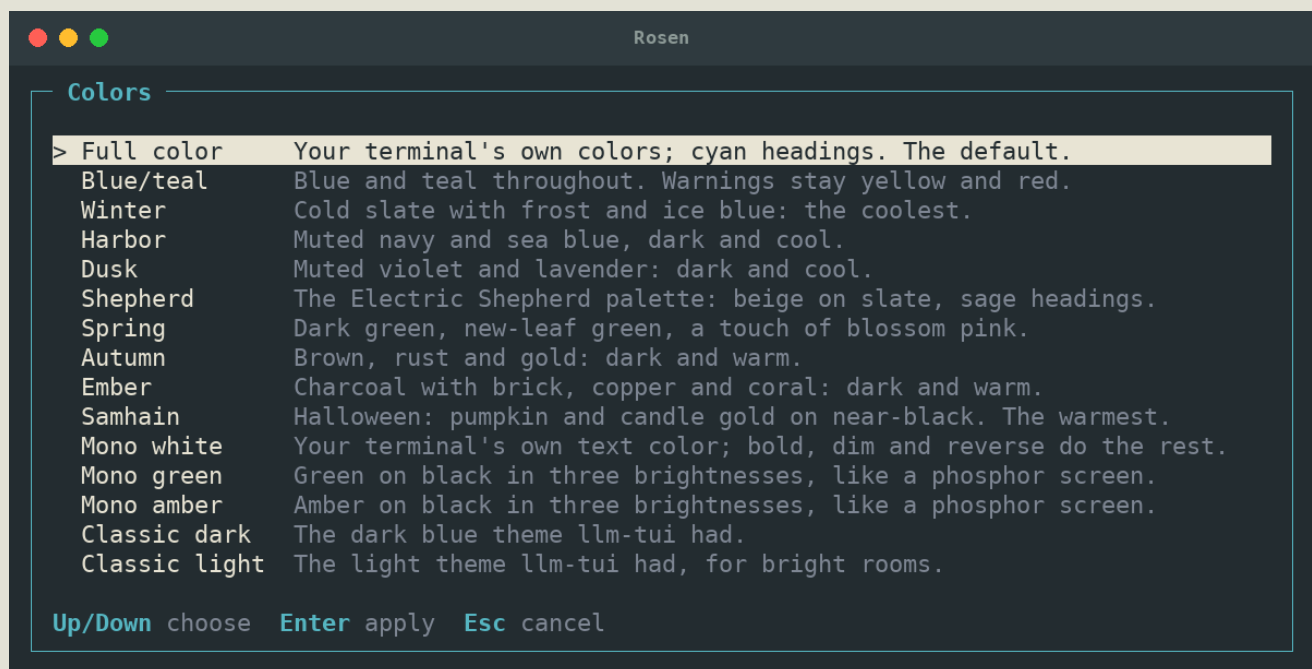


Figure 51. The Colors menu. Each look has a one-line description; the screen behind the menu changes as you move.

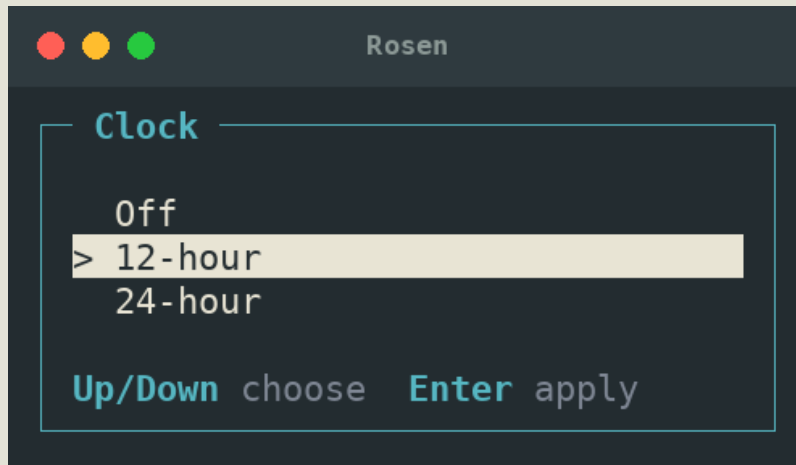


Figure 52. The Clock menu.

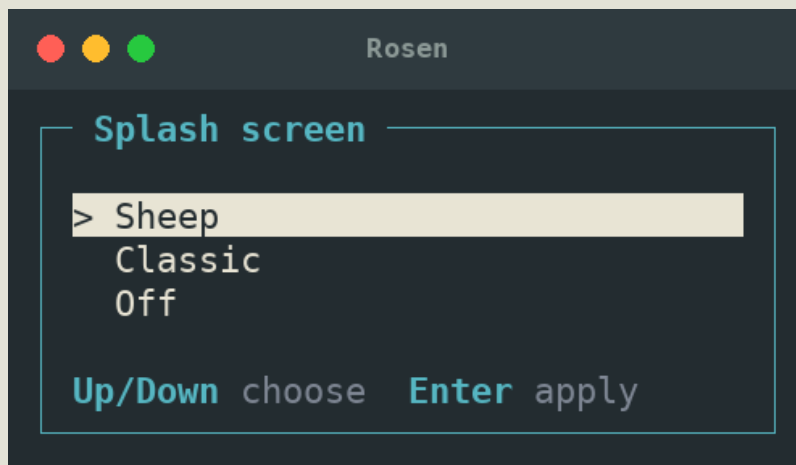


Figure 53. The Splash screen menu.

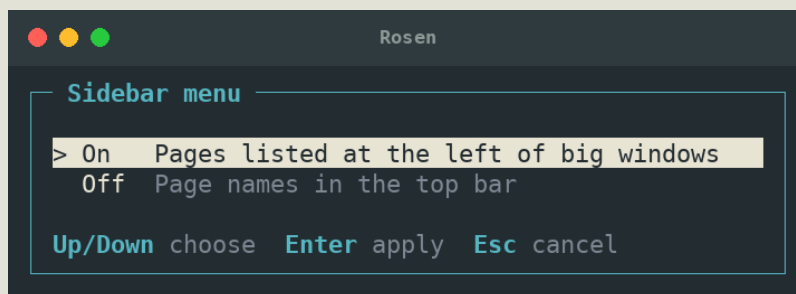


Figure 54. The Sidebar menu setting: On or Off.

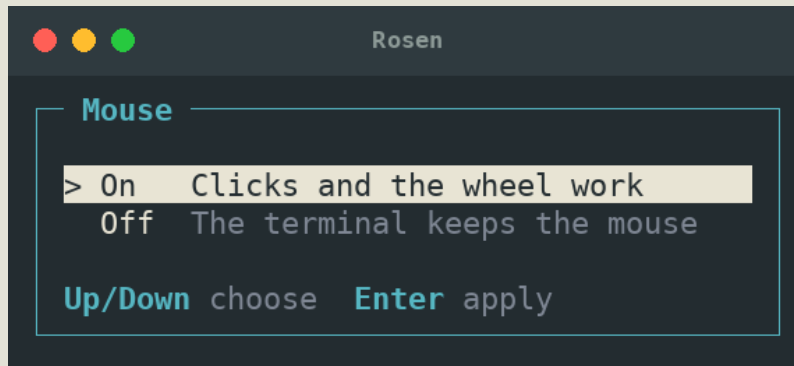


Figure 55. The Mouse setting: On or Off.

13.2 Sent to the model

Setting	Default	Range	What it does
Temperature	0.7	0 to 2	How adventurous the model is: 0 is steady, 2 is wild.
Top-p	1	0 to 1	Limits the choice of words to the likeliest ones: 1 uses them all, 0.1 only the likeliest tenth.
Longest reply	server's choice (0)	0 to 10,000,000	The longest a reply may be, in tokens. 0 lets the server decide.
Memory	last 40 messages	0 to 10,000	How many of the latest messages are sent with each question. 0 sends the whole chat. (Chapter 5.7.)
Time limit	120 seconds	5 to 86,400	How long to wait for a server before giving up. A reply that is still arriving does not count as waiting; the limit is for silence.
Show replies as they arrive	Yes	Yes or No	Show a reply word by word, rather than all at once when it is finished.
Default instructions	You are a helpful assistant.	up to 20,000 characters	Standing instructions sent first with every chat. A chat can have its own (p on the Chat page).

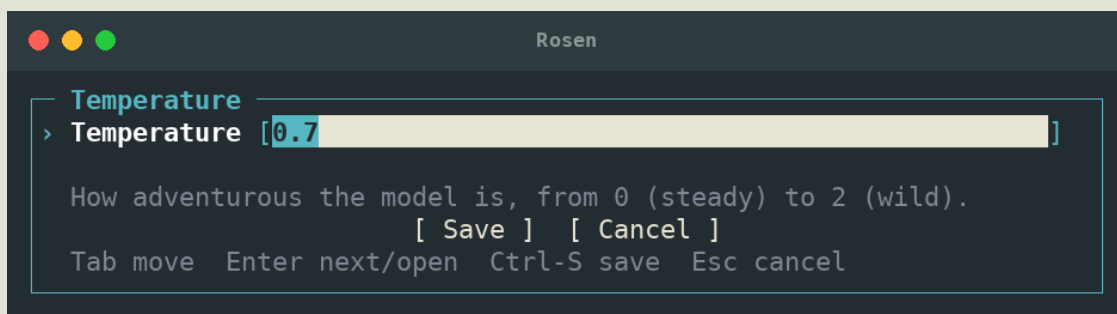


Figure 56. A number setting. The line under the field says what the number means and its range.

If you type a value outside the range, the form says what the range is and keeps the old value. `/set` changes the same numbers from the message box (section 5.9): `/set temperature 0.2`, `/set top_p 0.9`, `/set max_tokens 512`, `/set memory 20`, `/set timeout 300`, `/set stream off`. Temperature and Top-p are moved into their range if you go past it; the others are refused with a message that names the range.

Note. These numbers go to the server with every request. Not every server pays attention to all of them: a server may ignore Top-p, for instance, or fix the length of the reply itself. Rosen sends them and does not second-guess the answer.

13.3 The files at the foot of the page

Under the explanation of the highlighted setting, the page names where its data lives: the **Data files** (the folder with `rosen.json` and the chats), the **Settings file** (the screen settings, section 17.1) and the **Version** of this program, with the date it was built. In demo mode (`-D`) it says that nothing is saved.

14. Looks

Rosen has fifteen **looks**. You choose one on the Settings page (**0** , then **Colors**), step through them from anywhere with **{** and **}** , or pick one for a single run with `-c (rosen -c amber)`, or from the message box with `/look amber`. A look chosen on the Settings page or with **{** and **}** is remembered; one given with `-c` is for that run only.

◆ Rosen v0.1.1a Wed Oct 21 2026 10:24 AM

1 Chat
2 Sessions
3 Servers
4 Models
5 Summary
6 Checks
0 Settings

1 Parsing a config file in Rust
LM Studio · qwen2.5-coder-7b-instruct

You 10:16
Good. Now make it return an error for

qwen2.5-coder-7b-instruct · LM Studio
Change the return type and number the code 2 · rust

```
pub fn parse(text: &str) -> Result<B
let mut out = BTreeMap::new();
for (n, line) in text.lines().en
let line = line.trim();
if line.is_empty() || line.s
continue;
```

Full color

◆ Rosen v0.1.1a Wed Oct 21 2026 10:24 AM

1 Chat
2 Sessions
3 Servers
4 Models
5 Summary
6 Checks
0 Settings

1 Parsing a config file in Rust
LM Studio · qwen2.5-coder-7b-instruct

You 10:16
Good. Now make it return an error for

qwen2.5-coder-7b-instruct · LM Studio
Change the return type and number the code 2 · rust

```
pub fn parse(text: &str) -> Result<B
let mut out = BTreeMap::new();
for (n, line) in text.lines().en
let line = line.trim();
if line.is_empty() || line.s
continue;
```

Blue/teal

◆ Rosen v0.1.1a Wed Oct 21 2026 10:24 AM

1 Chat
2 Sessions
3 Servers
4 Models
5 Summary
6 Checks
0 Settings

1 Parsing a config file in Rust
LM Studio · qwen2.5-coder-7b-instruct

You 10:16
Good. Now make it return an error for

qwen2.5-coder-7b-instruct · LM Studio
Change the return type and number the code 2 · rust

```
pub fn parse(text: &str) -> Result<B
let mut out = BTreeMap::new();
for (n, line) in text.lines().en
let line = line.trim();
if line.is_empty() || line.s
continue;
```

Winter

◆ Rosen v0.1.1a Wed Oct 21 2026 10:24 AM

1 Chat
2 Sessions
3 Servers
4 Models
5 Summary
6 Checks
0 Settings

1 Parsing a config file in Rust
LM Studio · qwen2.5-coder-7b-instruct

You 10:16
Good. Now make it return an error for

qwen2.5-coder-7b-instruct · LM Studio
Change the return type and number the code 2 · rust

```
pub fn parse(text: &str) -> Result<B
let mut out = BTreeMap::new();
for (n, line) in text.lines().en
let line = line.trim();
if line.is_empty() || line.s
continue;
```

Harbor

◆ Rosen v0.1.1a Wed Oct 21 2026 10:24 AM

1 Chat
2 Sessions
3 Servers
4 Models
5 Summary
6 Checks
0 Settings

1 Parsing a config file in Rust
LM Studio · qwen2.5-coder-7b-instruct

You 10:16
Good. Now make it return an error for

qwen2.5-coder-7b-instruct · LM Studio
Change the return type and number the code 2 · rust

```
pub fn parse(text: &str) -> Result<B
let mut out = BTreeMap::new();
for (n, line) in text.lines().en
let line = line.trim();
if line.is_empty() || line.s
continue;
```

Dusk

◆ Rosen v0.1.1a Wed Oct 21 2026 10:24 AM

1 Chat
2 Sessions
3 Servers
4 Models
5 Summary
6 Checks
0 Settings

1 Parsing a config file in Rust
LM Studio · qwen2.5-coder-7b-instruct

You 10:16
Good. Now make it return an error for

qwen2.5-coder-7b-instruct · LM Studio
Change the return type and number the code 2 · rust

```
pub fn parse(text: &str) -> Result<B
let mut out = BTreeMap::new();
for (n, line) in text.lines().en
let line = line.trim();
if line.is_empty() || line.s
continue;
```

Shepherd

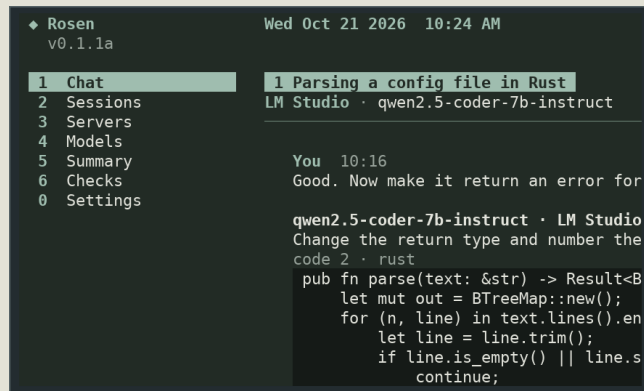
Figure 57. The six cool looks, each shown on the Chat page. Every signal that color gives is also given by brightness, bold or reverse video, so each look stays readable on its own.

14.1 The looks

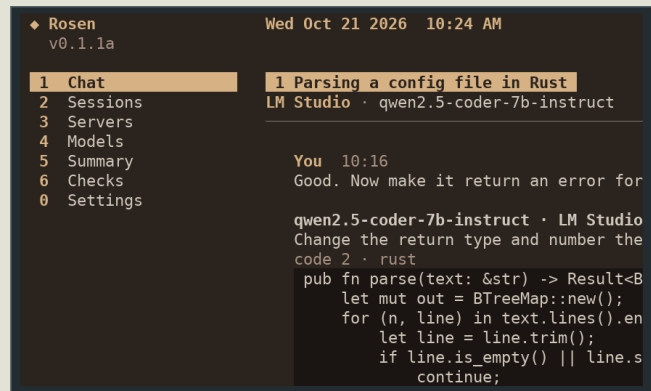
The list is in the order `} steps through it. A message names the one you land on: Look: Winter (3 of 15).`

Look	Name	What it is
Full color	<code>full</code>	The default. Your terminal's own colors; cyan headings, and green, yellow and red carry meaning.
Blue/teal	<code>teal</code>	Blue and teal throughout. Warnings stay yellow and red.
Winter	<code>winter</code>	Cold slate with frost and ice blue: the coolest.
Harbor	<code>harbor</code>	Muted navy and sea blue, dark and cool.
Dusk	<code>dusk</code>	Muted violet and lavender, dark and cool.
Shepherd	<code>shepherd</code>	The Electric Shepherd palette: beige on slate, with sage headings.
Spring	<code>spring</code>	Dark green, new-leaf green and a touch of blossom pink.
Autumn	<code>autumn</code>	Brown, rust and gold: dark and warm.
Ember	<code>ember</code>	Charcoal with brick, copper and coral: dark and warm.
Samhain	<code>samhain</code>	Halloween: pumpkin and candle gold on near-black. The warmest.
Mono white	<code>white</code>	Your terminal's own text color; bold, dim and reverse video do the work color did. It is also what <code>NO_COLOR</code> gives you.
Mono green	<code>green</code>	Green on black in three brightnesses, like a phosphor screen.
Mono amber	<code>amber</code>	Amber on black in three brightnesses.
Classic dark	<code>classic-dark</code>	The dark blue theme <code>llm-tui</code> had, for anyone who liked it.
Classic light	<code>classic-light</code>	The light theme <code>llm-tui</code> had, for bright rooms.

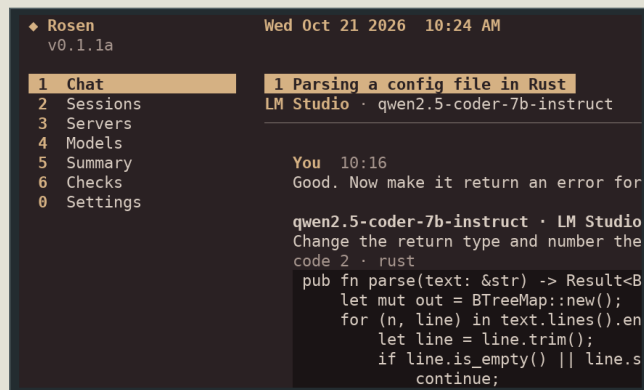
`/look` and `-c` also understand the names `blue`, `dark`, `light` and `paper`, so what you typed in `llm-tui` still works.



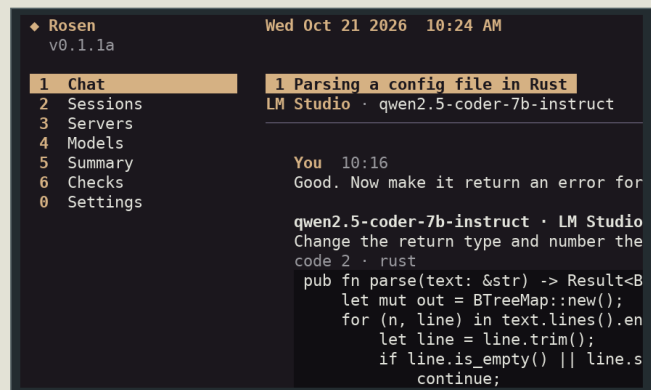
Spring



Autumn



Ember

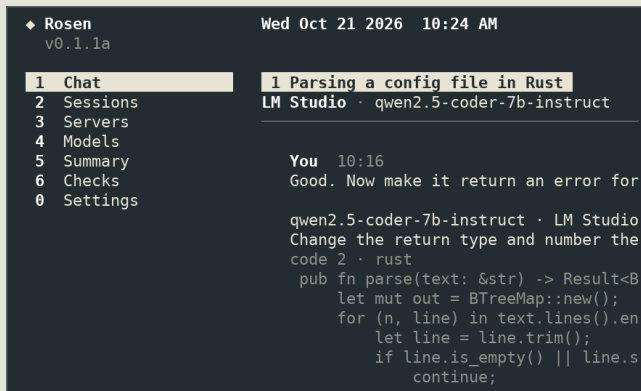


Samhain

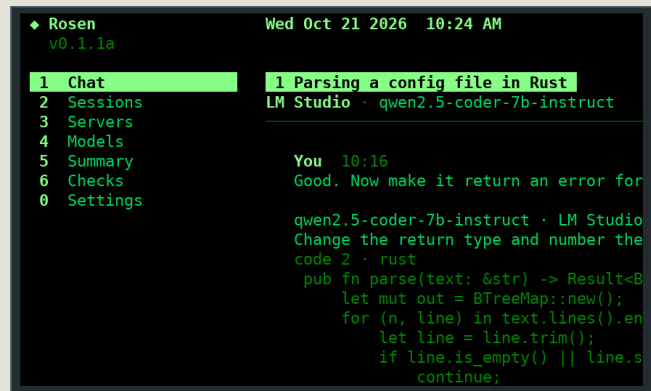
Figure 58. The four warm looks: Spring, Autumn, Ember and Samhain.

14.2 The monochrome and classic looks

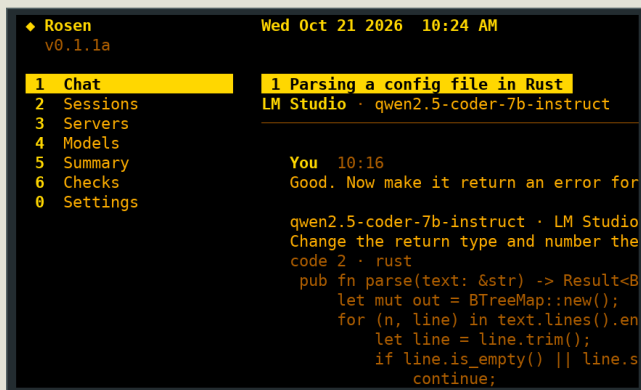
The monochrome looks use three levels of the same color. The brightest is for what you are acting on (the current page, the selected row, the alert badge), the middle for text, and the dimmest for details and borders; reverse video marks the selection. Nothing depends on telling red from green: errors say failed, warnings say what is wrong, and the current page is marked by shape as well as color.



Mono white

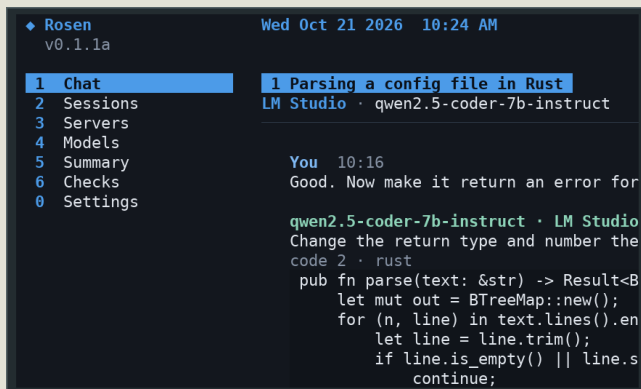


Mono green

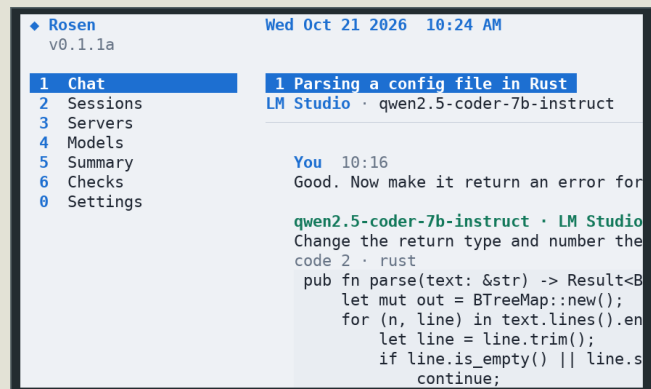


Mono amber

Figure 59. Mono white, Mono green and Mono amber.



Classic dark



Classic light

Figure 60. The two classic looks, kept from llm-tui.

14.3 Reading them

```
llama3.2:3b · Ollama (this computer) 10:24
Here is a small example from llama3.2:3b:
code 1 · rust
fn main() {
    println!("hello from the demo server");
Code in this chat
> 1 rust      fn main() { 3 lines
Enter read r run s shell y copy w save Esc close
```

Full color

```
llama3.2:3b · Ollama (this computer) 10:24
Here is a small example from llama3.2:3b:
code 1 · rust
fn main() {
    println!("hello from the demo server");
Code in this chat
> 1 rust      fn main() { 3 lines
Enter read r run s shell y copy w save Esc close
```

Winter

```
llama3.2:3b · Ollama (this computer) 10:24
Here is a small example from llama3.2:3b:
code 1 · rust
fn main() {
    println!("hello from the demo server");
Code in this chat
> 1 rust      fn main() { 3 lines
Enter read r run s shell y copy w save Esc close
```

Shepherd

```
llama3.2:3b · Ollama (this computer) 10:24
Here is a small example from llama3.2:3b:
code 1 · rust
fn main() {
    println!("hello from the demo server");
Code in this chat
> 1 rust      fn main() { 3 lines
Enter read r run s shell y copy w save Esc close
```

Samhain

```
llama3.2:3b · Ollama (this computer) 10:24
Here is a small example from llama3.2:3b:
code 1 · rust
fn main() {
    println!("hello from the demo server");
Code in this chat
> 1 rust      fn main() { 3 lines
Enter read r run s shell y copy w save Esc close
```

Mono green

```
llama3.2:3b · Ollama (this computer) 10:24
Here is a small example from llama3.2:3b:
code 1 · rust
fn main() {
    println!("hello from the demo server");
Code in this chat
> 1 rust      fn main() { 3 lines
Enter read r run s shell y copy w save Esc close
```

Classic light

Figure 61. A reply with code in six of the looks, with the code menu open on top.

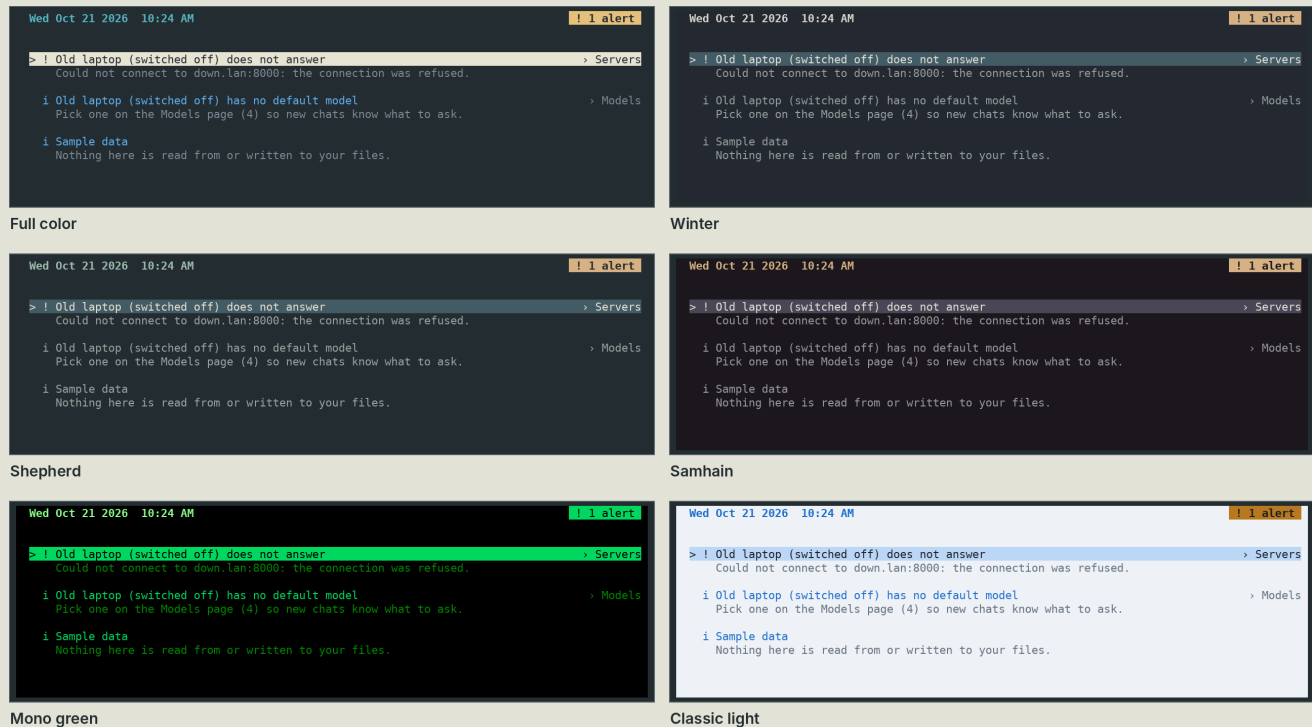


Figure 62. The Checks page in the same six: warnings and trouble keep their own colors, and in the mono looks they are brighter or in reverse.

The contrast rules

The eight muted dark looks (Winter, Harbor, Dusk, Shepherd, Spring, Autumn, Ember and Samhain) are built to the same measures, and a test holds every one of them to those measures, using the WCAG contrast ratio:

- **Text** is at least **7 to 1** against the window, a dialog and a form field.
- **Headings, dim text, and the good, warning and error colors** are at least **4.5 to 1** against the window; headings also against a dialog.
- **Rules** (the lines between things) are at least **3 to 1**.
- Text on the **selected row** is at least 4.5 to 1, and so is the dark text on a filled heading, warning or error label.

So a look that is quiet is still easy to read; none of them trades legibility for mood.

14.4 Choosing a look

Two ways, which end in the same place:

- **From anywhere**, press **}** for the next look or **{** for the previous one; the whole screen changes and a message names it. The choice is saved.
- **On the Settings page**, **Enter** on **Colors** opens a menu of the fifteen looks. As you move the highlight with **Up** and **Down** the whole screen behind the menu changes to that look, so you see it before you choose. **Enter** keeps it and saves it; **Esc** puts the old look back as if nothing had happened.

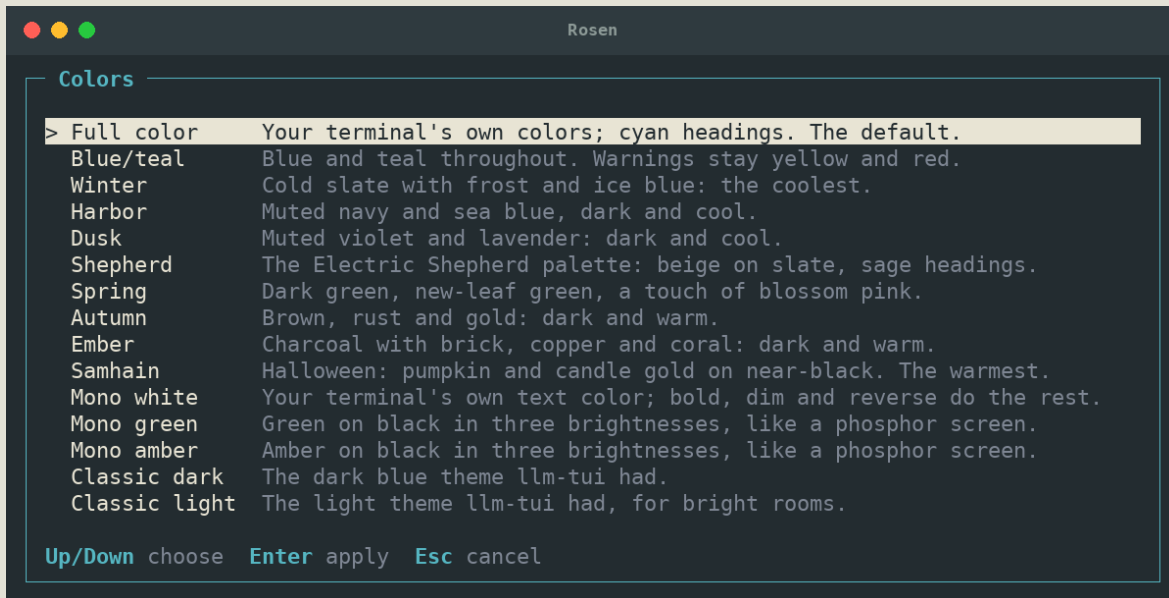


Figure 63. The Colors menu on the Settings page. The screen behind it is already in the highlighted look.



Figure 64. The start-up screen in all fifteen looks.

14.5 Color depth and terminals

- Rosen uses **24-bit color** when your terminal says it has it (the `COLORTERM` variable is `truecolor` or `24bit`), and otherwise the nearest of the 256 standard colors. The mono looks look right in either.
- `NO_COLOR`: if the variable is set to anything that is not empty, Rosen starts in Mono white, whatever the setting says. An explicit `-c look` on the command line wins over it.
- **Background**. Full color, Blue/teal and Mono white leave your terminal's own background alone; the other twelve paint the whole window. Full color is meant for dark terminals: on a light one, choose Classic light or Mono white, or paint your terminal dark.
- **ASCII only**. If your locale is not UTF-8 (or `ROSEN_ASCII` is set), Rosen draws boxes and bars with plain ASCII characters. Everything still works; it just looks plainer.

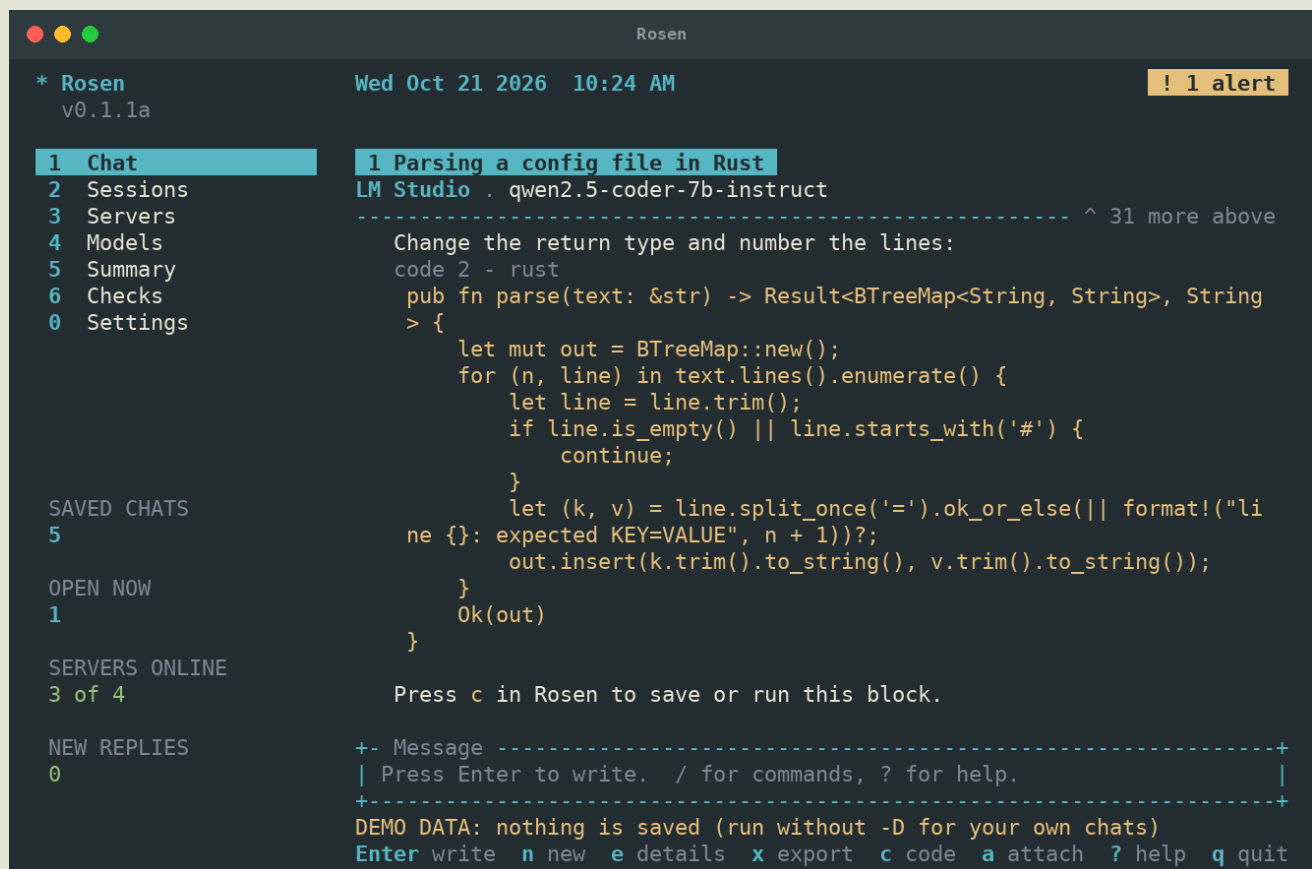


Figure 65. The Chat page with plain ASCII drawing, as it appears when the terminal does not announce UTF-8.

15. Replies You Are Not Looking At

A model can take a while, and you do not have to watch. Rosen keeps every reply arriving while you read another chat, look at another page, or work in another window, and when a reply **ends** out of sight (it finished, or it failed) it tells you.

15.1 What counts as out of sight

A reply is out of sight when any of these is true at the moment it finishes:

- you are on a **different page** (Sessions, Servers, Settings...),
- you are looking at a **different chat** tab,
- the **terminal window is not focused** (you are in another window or another desktop), if your terminal reports focus.

Rosen asks the terminal to say when it gains or loses focus; most modern terminals do. If yours does not, Rosen simply behaves as if the window were always in front.

A reply you stopped yourself does not count: you already know about it. A reply that **fails** does count, because you will want to know that it did not work.

15.2 What you see and hear

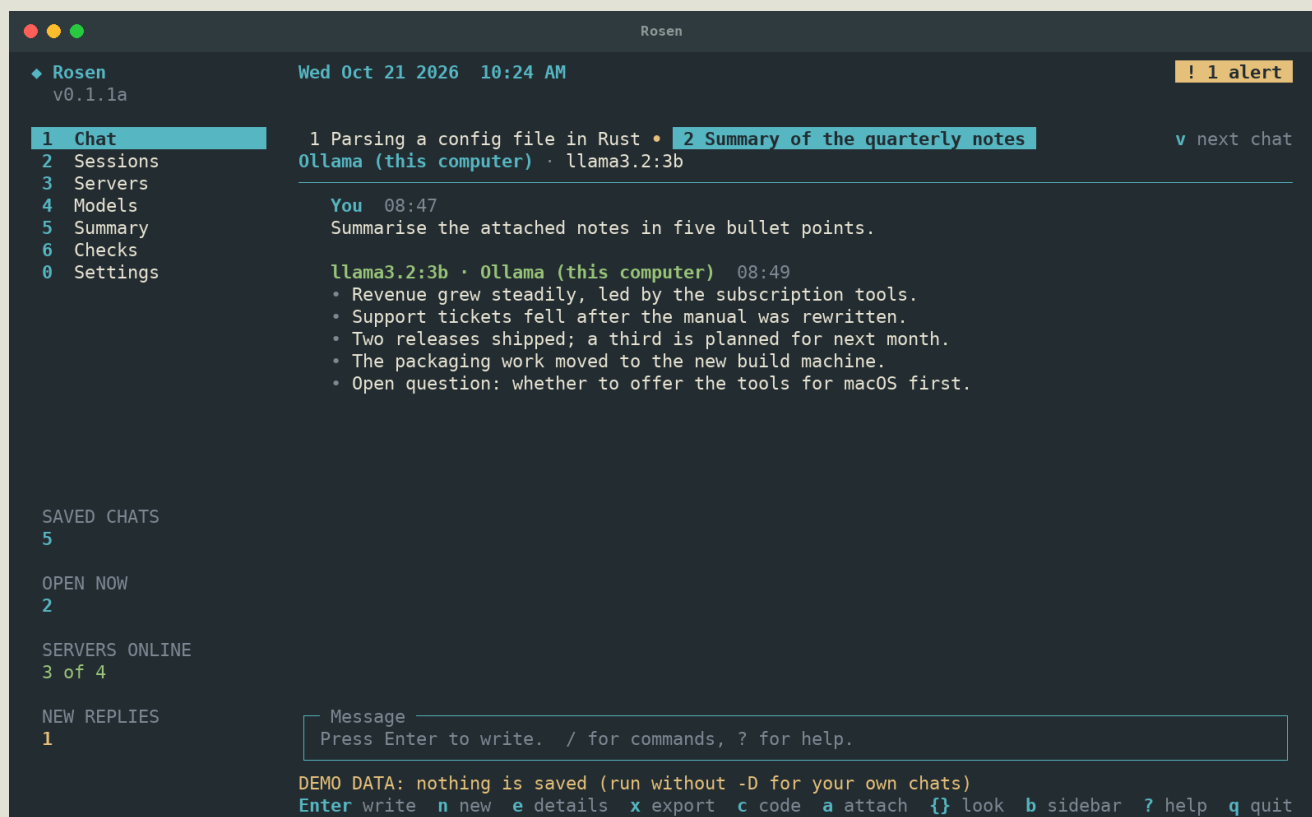


Figure 66. A reply finished in the first chat while the second one was in front. The first tab carries a dot, and New replies in the sidebar counts it.

- **The tab marker.** A dot (•) after the chat's title on its tab.
- **The sidebar count.** `NEW REPLIES` shows how many chats have an unread reply, and `OPEN NOW` how many are open. (In the top-bar layout it is `New 1` on the status line.)
- **The bell.** The terminal's bell (the character a terminal rings or flashes for, depending on how you set it up) is sent once, when the reply finishes.

On the Sessions page the chat's number in the first column takes the same marker color. The marker and the count go away when you look at the chat: when the Chat page is showing that chat, and the window is in front, it is cleared.

15.3 Choosing

The **Reply alert** setting on the Settings page has three values:

Value	What happens
Bell and marker	The marker, the count and the terminal bell. The default.
Marker only	The marker and the count, no sound. For a quiet room.
Off	Nothing at all.

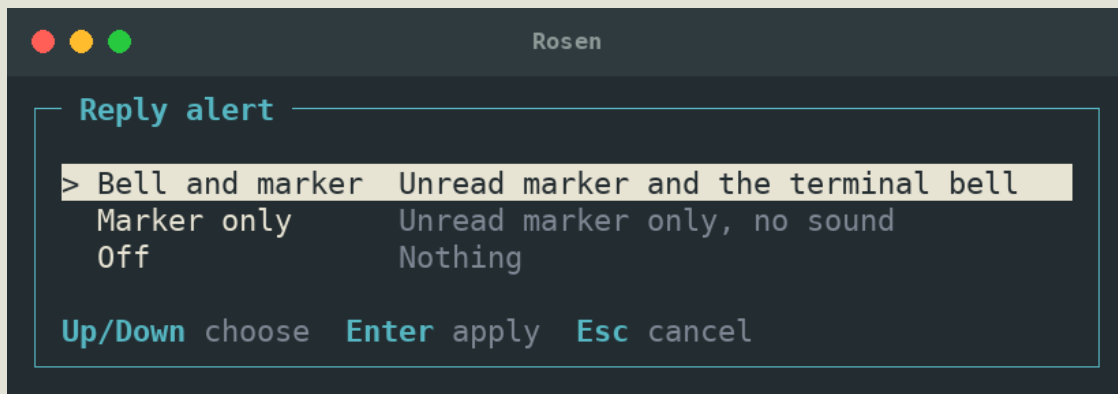


Figure 67. The Reply alert menu.

Tip. Many terminals can turn the bell into a notification, a flash of the window or a sound of your choosing, and some show a badge on the tab or the dock icon. If you want your phone or desktop to tell you too, set that up in the terminal's own preferences: Rosen only rings the bell.

15.4 Several chats at once

This is what makes it comfortable to run several models side by side. Ask a large, slow model something on one tab, open another with `n` and ask a small one, and carry on. Whichever finishes first rings the bell and marks its tab. When you come back to a marked tab the reply is there, complete, and the marker clears.

If you quit while a reply is still arriving, Rosen asks first and keeps what has arrived so far.

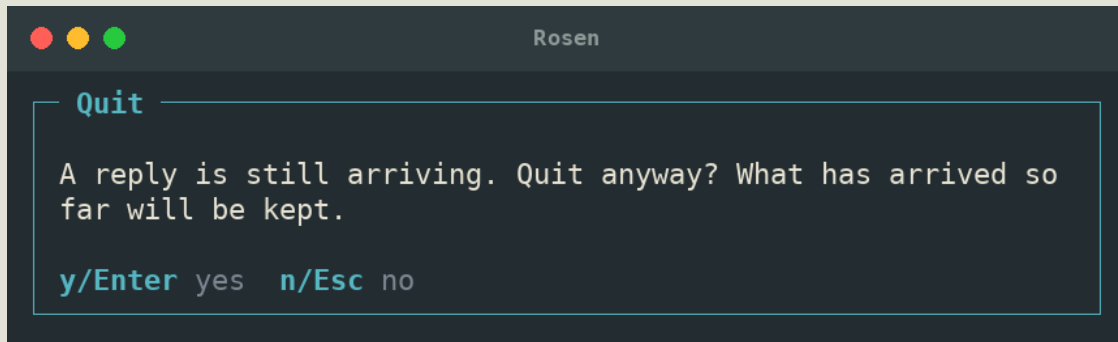


Figure 68. Quitting while a reply is arriving asks first; what has arrived so far is kept.

16. The Command Line

Rosen is an ordinary Unix command. It follows the POSIX utility conventions: single-letter options that may be grouped (`-Dn`), option arguments attached or separate (`-ffile` or `-f file`), `--` to end the options, and the exit statuses 0, 1 and 2. The long forms (`--file`, `--demo...`) are a convenience on top of that, not a replacement.

16.1 Synopsis

```
rosen [-Dn] [-c look] [-p page] [-f file | file]
rosen -d [-n] [-c look] [-p page] [-f file | file | -D] [-s WxH] [-k keys]
rosen -h | -V | -L
```

With no arguments Rosen opens your chats on the terminal you are in. Standard input and output must both be a terminal; otherwise Rosen says so and stops (use `-d`, below, to get a page as text).

16.2 Options

Option	What it does
<code>-f, --file FILE</code>	Use this data file instead of the usual one. The chats are kept in a folder next to it (section 17.1). Different files are entirely separate: their own servers, their own chats, their own lock.
<code>-D, --demo</code>	Start with sample data and a built-in echo server. Nothing is read from or written to your files. It cannot be combined with <code>-f</code> . Chapter 18.
<code>-c, --colors NAME</code>	Use this color look for this run only; the saved setting is not changed. The names are full, teal, winter, harbor, dusk, shepherd, spring, autumn, ember, samhain, white, green, amber, classic-dark and classic-light (Chapter 14). <code>--look</code> is accepted as well.
<code>-p, --page PAGE</code>	Open (or, with <code>-d</code> , print) this page: chat, sessions, servers, models, summary, checks or settings; a page's number key (<code>-p 2</code>) also works, and splash shows the start-up screen. It overrides the Start on page setting for this run.
<code>-n, --no-splash</code>	Skip the start-up screen.
<code>-d, --dump</code>	Print one page as plain text on standard output and exit (section 16.3). Needs no terminal and never writes a file.
<code>-s, --size WxH</code>	With <code>-d</code> : the page size in columns and rows (1 to 1000 each), such as <code>-s 100x30</code> . Without it, the <code>COLUMNS</code> and <code>LINES</code> variables if both are set, otherwise 100 by 32.
<code>-k, --keys KEYS</code>	With <code>-d</code> : press these keys first (section 16.3).
<code>-h, --help</code>	Show a summary of all this and exit.
<code>-V, --version</code>	Show the version and exit: Rosen v0.1.1a (alpha, built 2026-10-22).
<code>-L, --licence</code>	Show the terms of use and exit. The first line names the product, the version and the company. <code>--license</code> works too.

The one operand is the **data file**, so `rosen chats.json` is the same as `rosen -f chats.json`. The page to open or print is chosen with `-p` and defaults to the **Start on page** setting for a program and to chat for `-d`. `-s` and `-k` without `-d` are errors.

Options may follow operands, as in `rosen chats.json -n`. If `POSIXLY_CORRECT` is set, the first operand ends the options, as POSIX says it should.

16.3 Looking at a page as text

`-d` draws one page exactly as it would appear on the terminal, but prints it as plain text. It is meant for scripts, for bug reports, and for checking what Rosen will show without starting it. By default it reads your own data file, read-only; add `-D` to see the sample data instead.

```
rosen -d -D -n -s 80x14 -p sessions
```

prints the Sessions page of the sample data in an 80 by 14 window:

```
Chat Sessions Servers Models Summary Checks Settings      ! 1 alert
-1-----2=====3-----4-----5-----6-----0-----
Wed Oct 21 2026 10:24 AM Chats 5 Online 3/4 New 0
All Ollama (this computer) LM Studio > v next view
Sorted by Updated v o changes it 5 chats
# Title Server Model Msgs Updated v
> 1 Parsing a config file in~ LM Stud~ qwen2.5~ 4 2026-10-21 10:18
Summary of the quarterly~ Ollama ~ llama3.~ 2 2026-10-21 08:49
Dinner ideas for four Worksta~ mistral~ 2 2026-10-20 08:24
Find the biggest files Ollama ~ qwen2.5~ 2 2026-10-19 06:24
Letter to the landlord Ollama ~ gemma2:~ 2 2026-10-15 10:24
DEMO DATA: nothing you do here is saved (run without -D for your own chats).
Enter open a add e edit c clone d del b sidebar ? help q quit
```

The page is drawn with plain ASCII characters whenever the locale is not UTF-8, as it is here; otherwise with the same box-drawing characters you see on screen.

Pressing keys first

`-k` takes a list of keys, separated by spaces, which Rosen presses before it draws the page. It reaches dialogs and states that a page name alone cannot:

```
rosen -d -D -n -s 64x14 -k "3 a text:Studio Tab text:10.0.0.5:8080"
```

presses **3** (the Servers page), **a** (add a server), types Studio, **Tab**, types an address, and prints the form with those values in it:

```

Chat Sess Serv Model Sum Chk Cfg          ! 1 alert
+- Add server -----+
| Name           [Studio           ] |
|> Address       [10.0.0.5:8080    ] |
| Type           [Ollama           v] |
| API key        [                  ] |
| Default model   [                  v] |
| Check certificate [Yes            v] |
|
| Where it is, for example http://192.168.1.20:8080.
| https:// needs curl.
|
|           [ Save ] [ Cancel ]
| Tab move  Enter next/open  Ctrl-S save  Esc cancel
+-----+
DEMO DATA: nothing is saved (run without -D for your own chat~
Enter edit  a add  c clone  d del  u use          ? help  q quit

```

The names -k understands:

Token	Key
Enter, Tab, Esc, Space, Up, Down, Left, Right, Home, End, PgUp, PgDn, Delete, Backspace, BackTab	The key of that name.
F1 to F12	A function key.
C-x (or Ctrl-x, or ^x)	Control and a letter from a to z.
A-x (or Alt-x)	Alt and one character.
text:abc	Type the characters abc. Inside it s is a space and \ is a backslash.
click:X,Y	A left click at column X, row Y, counted from 1 as a terminal reports them. Needs the Mouse setting On (the default).
wheelup:X,Y, wheel down:X,Y	A turn of the wheel at that place.
wait	Wait until nothing is arriving any more (a reply, a test) before pressing on.
any single character	That character.

With -D the built-in echo server answers, so -k "Enter text:hello Enter wait" prints a chat with a reply in it, with no model needed.

16.4 Environment

Variable	Effect
ROSEN_FILE	The data file, if -f is not given.
XDG_DATA_HOME, XDG_CONFIG_HOME, HOME	Where the data and the screen settings are kept (section 17.1). A value that is empty or not an absolute path is ignored, as the XDG specification says.

Variable	Effect
NO_COLOR	If set and not empty, start in Mono white. -c overrides it.
COLORTERM	truecolor or 24bit allows 24-bit color; otherwise Rosen uses the 256-color palette.
LC_ALL, LC_CTYPE, LANG	A UTF-8 locale selects the box-drawing characters; anything else selects plain ASCII.
ROSEN_ASCII, ROSEN_UNICODE	Force plain ASCII, or force box-drawing, whatever the locale says.
ROSEN_NO_SPLASH	Skip the start-up screen (same as -n).
COLUMNS, LINES	The default size for -d.
SHELL	The program F2 and /shell open (an absolute path to a file that exists; otherwise sh).
ROSEN_TODAY, ROSEN_TIME	Pretend it is this date (YYYY-MM-DD) and time (HH:MM). The clock is hidden unless both are given. For screenshots and tests.
POSIXLY_CORRECT	The first operand ends the options.

Rosen also sets two variables for the shell it opens: ROSEN holds the version (v0.1.1a) and ROSEN_CODE names the file that holds the newest code block of the chat (Chapter 9).

16.5 Exit status

Status	Meaning
0	Success: you quit, or -h, -V or -d finished.
1	An error while running: the data file could not be read, another copy has the data file open, or there is no terminal.
2	A usage error: an unknown option, a missing argument, a look or page that does not exist, or options that cannot be combined.

Diagnostics go to standard error, each starting with rosen:; a usage error is followed by a line saying how to get help.

```
$ rosen -c mono -d
rosen: unknown look 'mono' (use full, teal, winter, harbor, dusk, shepherd, spring, autumn, ember, samhain, white, green, amber, classic-dark, classic-light)
Try 'rosen --help' for more information.
$ echo $?
2
```

17. Your Data

Everything Rosen keeps is in plain files in your own account. You can read them, copy them, back them up and put them under version control. Nothing is kept anywhere else.

17.1 Where the files are

File	What it holds
<code>~/.local/share/rosen/rosen.json</code>	Servers and settings. Your servers (name, address, type, API key, default model, certificate check), which one is active, the last chat, the default instructions, and the numbers sent to the model (Chapter 13.2). Mode 0600: only you can read it, because it holds your keys.
<code>~/.local/share/rosen/sessions/</code>	One file per chat , named by a random identifier, such as <code>3f9c1a2e.json</code> . Mode 0600, in a folder of mode 0700.
<code>~/.config/rosen/settings</code>	The screen settings: colors, clock, splash, start page, sidebar, mouse and reply alert, as <code>key=value</code> lines (<code>colors=winter</code> , <code>mouse=off</code> , and so on). Edit it by hand if you like; a line Rosen does not understand is skipped and listed on the Checks page.
<code>~/.local/share/rosen/rosen.json.bak</code>	The data file as it was before this run's first change . Same mode as the original.
<code>~/.local/share/rosen/sessions/ID.json.bak</code>	The same for a chat: the file as it was before this run first saved it.
<code>~/.local/share/rosen/rosen.json.lock</code>	Present while Rosen is running: it holds the process number of the running copy.
<code>~/.local/share/rosen/crash.log</code>	Written only if something went wrong inside Rosen (section 17.3).

The paths above are the defaults. If `XDG_DATA_HOME` is set (to an absolute path) the data goes in `$XDG_DATA_HOME/rosen/` instead of `~/.local/share/rosen/`, and `XDG_CONFIG_HOME` does the same for the settings. `-f FILE` (or `ROSEN_FILE`) chooses the data file itself; the chats then live next to it in a folder called `FILE.sessions`, and the lock is `FILE.lock`, so two data files never share chats, and you can run two copies of Rosen on two data files at once.

A file called `rosen.json` is a single JSON document. A chat file is a JSON document too, with its messages in order. The format is the one `llm-tui` used, so existing data can be copied straight over (Appendix B). Keys that Rosen does not know are kept when it saves, so a newer or older version does not lose them.

Note. Exports, imports and attached files are not part of this list: they are where you said they should be, and Rosen touches them only when you ask (Chapter 10). Code you save with `w` and the temporary files used to run code are described in Chapter 9.

17.2 How your data is protected

Rosen is careful with files, because the chats are yours and the data file holds your keys.

- **Private by default.** Every file Rosen creates can be read only by you (mode 0600), and every folder it creates only entered by you (0700). A file that already exists keeps the mode it has; the Checks page tells you if that lets others read it, and how to fix it.
- **Saved as it changes.** A chat is written the moment a message is sent and again when a reply ends, so a crash or a closed terminal loses nothing that was already on screen.
- **Never half written.** Each file is written to a temporary file in the same folder, flushed to disk, and then renamed over the old one; the folder is flushed too. A power cut leaves either the old file or the new one, never a mixture.
- **A backup of what was there.** Before the first change of a run, the old data file is copied to `.bak`; before the first save of a chat in a run, the old chat file is copied to its own `.bak`. If something goes wrong, the file as it was at the start of that run is still there. Deleting a chat deletes its backup too, because it holds the same private text.
- **Checked before writing.** Rosen reads back what it is about to write with the same code that will read it later, and refuses to write a file it could not read again.
- **A file it cannot read is never replaced.** If the data file is damaged, Rosen starts with a message that names it and what is wrong, and stops; it does not make a new one over the top (it may hold your only copy of the keys). If a chat file is damaged, that one chat is listed, the Checks page says why, and the file is left exactly as it is. Everything else works.
- **One copy at a time.** The lock stops two copies from writing the same files. A second copy says already open in another copy (process N) and stops. If the first copy was killed and left its lock, the next start sees that the process is gone and takes the lock over. A copy that is still running under another user's name is never taken over.
- **The terminal is put back.** Rosen starts a small helper (itself, with a different argument) that watches over it. If Rosen is killed, crashes badly or loses its terminal, the helper puts the terminal back into its normal mode, leaves the alternate screen, shows the cursor, and removes the lock. You get your prompt back in working order.
- **Disk trouble is not silent.** If a file cannot be written (a full disk, a folder you may not write to), the message line turns red with `NOT SAVED`, the Checks page says what and where, and the alert badge counts it. It clears itself when a later save works.

17.3 If something goes wrong inside Rosen

A mistake in Rosen itself should never take your chats with it. Each key press is handled separately: if one of them hits an internal error, Rosen records it, closes any dialog, and carries on with a message that names the log. If it happens three times, Rosen stops. Either way the chats are on disk.

The record is `crash.log` next to the data file. It holds the version, the time, what Rosen was doing, the **last 40 keys** you pressed, and the technical detail of the error. Letters typed into a message, a form (an API key, say) or a search are shown as `char`, never as what you typed, so the log can be shared. The file is private (mode 0600) and grows by appending; delete it when you have read it. The Checks page reminds you while it exists.

17.4 Backing up and moving

To back up or move your Rosen, copy the data file and the folder of chats, with their modes:

```
cp -Rp ~/.local/share/rosen ~/rosen-backup
```

Do it with Rosen closed, or at least be aware that a chat being written while you copy may be in two states; the files are written atomically, so each file you get is whole. On another computer, put the folder in the same place (or point `-f` at it), and make sure `rosen.json` is mode 0600. If the folder you copy holds your API keys, treat the copy like a password file.

To forget everything, delete the folder. There is no other trace: no registry, no cache, no hidden state. (The screen settings are in `~/.config/rosen/` and can go too.)

17.5 What Rosen sends, and where

- **Only to the servers you listed.** Your messages, the chat so far, the instructions and attached files go to the server of the chat you are in, and nowhere else. There is no analytics, no update check, and no account.
- **API keys** go only to the server they belong to, as a bearer token. On `https://` they are handed to curl on its standard input, never on a command line where `ps` could show them. Over plain `http://` a key crosses the network readable by anyone on the path; the Checks page warns about that unless the server is on this computer.
- **The clipboard** is written only when you press `y` (or copy a code block), through the terminal's own OSC 52 feature. Rosen never reads it.
- **Programs Rosen starts.** `stty` (to put the terminal in raw mode and back), `date` (for your local time), `sh` (to check that a process is alive, to open your shell, to run a code block when you ask), `kill` (to stop a code block that ran out of time), `ps` (to tell whether the program named in a lock is still running), and `curl` (only for `https://` servers). It starts a runner such as `python3` only for a code block you chose to run.

17.6 Coming from llm-tui

Rosen reads llm-tui's files as they are. Appendix B gives the two commands that bring your servers, keys and chats across, and lists what moved and what changed.

18. Trying Rosen Without a Server

You do not need a model server to find your way around Rosen. Start it with `-D` (demo):

```
rosen -D
```

Demo mode loads **sample data** and a **built-in echo server**, and it touches none of your files: it does not read your data file, does not take the lock, and writes nothing, not even an export. A reminder is shown on the message line the whole time (DEMO DATA: nothing is saved), and the Checks page lists "Sample data". It cannot be combined with `-f`.

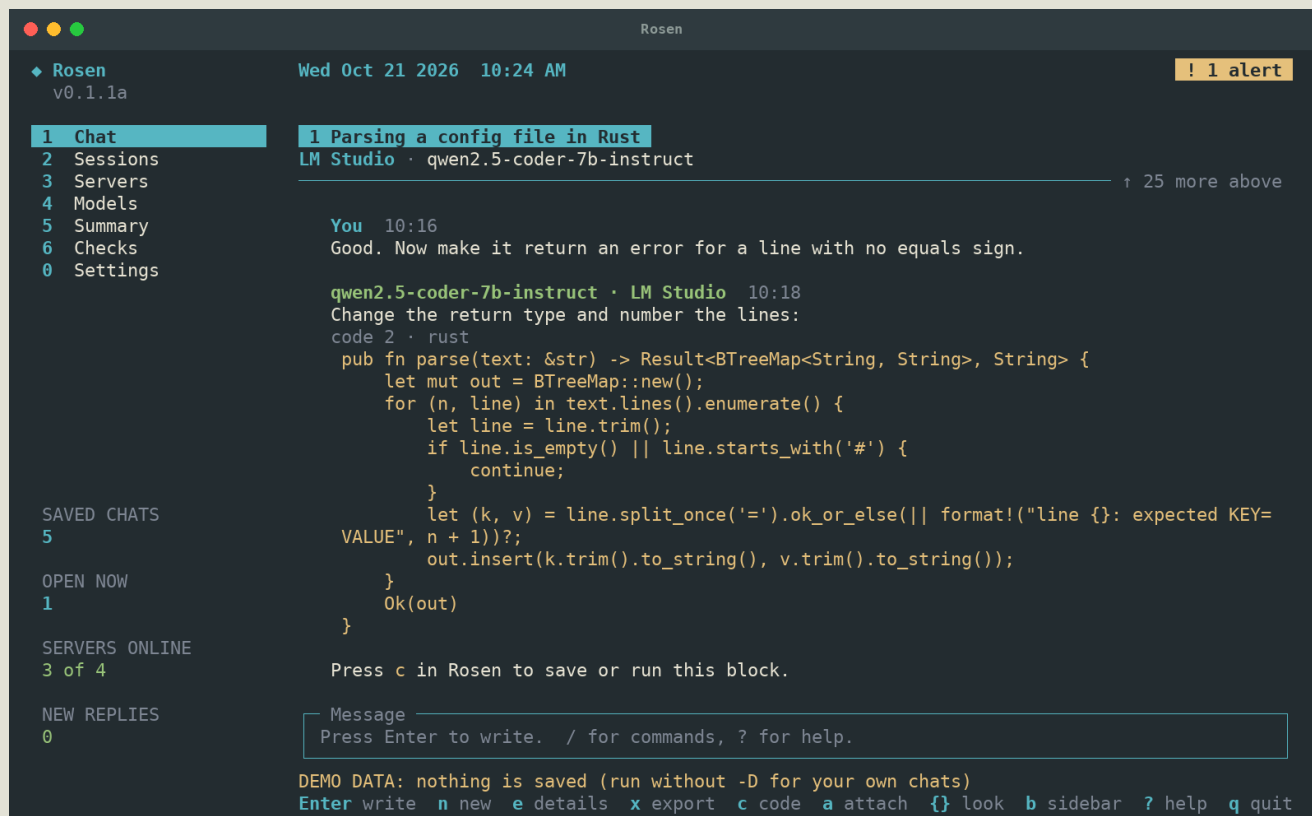


Figure 69. Demo mode. Every screenshot in this manual was taken from it.

18.1 What the sample data holds

- **Four servers.** Ollama (this computer), LM Studio and a llama.cpp Workstation, which answer; and Old laptop (switched off), which does not. Any server whose address contains the word down is treated as switched off, so you can see what a failing server looks like, and what the Checks page says about it. Add one called anything with down in its address to try it yourself.
- **Five chats** with their own servers and models, one of them with a Rust code block in it.
- **Model lists** for each server, so the Models page has something to show.

18.2 The echo server

The echo server answers every message by quoting it back: This is the built-in demo server (qwen2.5-coder-7b-instruct). It cannot think, but it can echo: you said "hello there". It cannot reason, but it behaves like a real server in every way that Rosen can see, so you can try everything that depends on a reply arriving: scrolling, stopping, asking again, tabs, the new-reply marker.

A few words in your message make it behave differently:

Put this in the message	And the demo server will
[code]	Answer with a short Rust program in a code block, to try the code menu (Chapter 9).
[slow]	Answer with a long reply, slowly, so that you have time to press <code>Esc</code> and see a stopped reply (Chapter 5).
[cut]	Start a reply and then drop the connection, so that you can see how a reply that was cut short is marked and kept.
[fail]	Refuse: the message line says why, and your message is still there to send again with <code>r</code> .

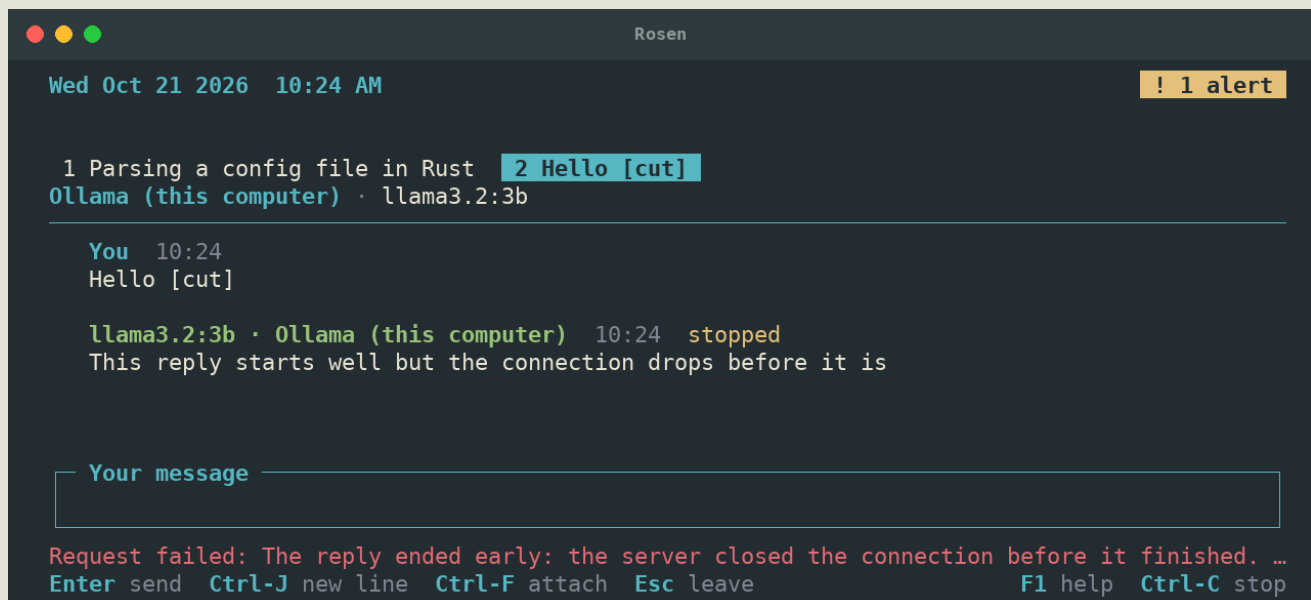


Figure 70. [cut]: what arrived is kept, the reply is marked stopped, and the message line says the server closed the connection.

18.3 What does not work in demo mode

Anything that would write or run something on your computer: **exporting to a file**, **saving a code block**, **running code**, and **opening a shell**. Each says on the message line that it is only done in a live session, instead of doing it, so a demonstration can never change your files or start a program. Everything else behaves as it does for real: sorting and searching, forms and their checks, help, the looks, the settings, even the settings pages (a change lasts until you quit).

To see the real thing with no model at all, use `-d` to print pages as text (Chapter 16).

19. When Something Goes Wrong

Rosen tries to say what went wrong in plain words and what to do about it, on the message line at the bottom of the screen and, for lasting things, on the Checks page (**6**). This chapter lists the messages you are most likely to meet. The text in quotation marks is what Rosen says; the names of the servers and so on will be yours.

19.1 A server does not answer

What Rosen says	What it means, and what to do
"Cannot reach the server (connection refused). Check the address, port and network."	Nothing is listening at that address. Is the server running? Is the port right (Ollama 11434, LM Studio 1234, llama-server 8080 by default)? A server that listens only on 127.0.0.1 cannot be reached from another computer.
"Cannot find 'name'. Check the address."	The name does not resolve. Check the spelling, or use the IP address.
"The server sent nothing for N s (timeout). Raise the timeout in Settings for slow models."	The server accepted the request and then went quiet. A large model can take a long while to start answering: raise Time limit in Settings (section 13.2).
"Server answered HTTP 404: ..."	The server is there but does not know the path. Usually the Type is wrong, or the address ends in a path the server does not have.
"This server speaks the OpenAI-compatible API, not Ollama ... press e and set Type to OpenAI-compatible"	Rosen noticed that the other type works, and tells you where to change it (the Servers page, e).
"Server answered HTTP 401 ... Check the API key." (also 403)	The server wants a key, or does not like the one it has. Edit the server and set API key .
"The server answered, but not with JSON ... Is this the right server type?"	Something answered, but not a model server: another program on that port, or a web page. Check the address and type.
"The reply is not the model list ..."	The server answered, but the answer is not a list of models. The same cause as above, or a server that does not support listing; type the model name by hand (n on the Models page).
"Invalid server address ..." and similar	The address cannot be understood. It should look like http://host:port or host:port.

Test a server again with **t** on the Servers page. The Checks page keeps a line for every server that failed its last try, with the reason.

19.2 HTTPS and certificates

What Rosen says	What to do
"HTTPS addresses need the 'curl' command, which was not found. Install curl, or use an http:// address."	Install curl. It is on practically every Linux and macOS system.
"HTTPS certificate problem. If this server uses a self-signed certificate, turn off "Check certificate" for it (Servers page, then e). curl said: ..."	Either make your system trust the certificate, or, for a test server, turn off Check certificate for that server. The Checks page keeps a reminder while it is off.
"curl failed (exit N): ..."	curl reported something Rosen has no plain words for; the rest of the line is curl's own.

19.3 A reply goes wrong

What you see	What to do
"Request failed: ..." and your message still in the chat	Fix the cause (above) and press r . Nothing was lost.
"The reply ended early: the server closed the connection before it finished. What arrived is kept."	The reply is marked stopped. The server went away (it crashed, ran out of memory or was restarted). r asks again.
"No model chosen for this chat. Press Esc, then m (or 4 for the Models page)."	Choose one: m on the Chat page, or the Models page.
The reply is cut off mid-sentence, with no error	The Longest reply setting, or the server's own limit, was reached. Raise it, or ask the model to carry on.
The model forgets the start of a long chat	Only the last Memory messages (40 by default) are sent. Raise it, or set it to 0 to send everything (the header says when the model is not seeing the whole chat).
The model ignores the instructions	Check that they are what you think: p shows this chat's own, Settings shows the default. A chat's own instructions replace the default.

19.4 The screen

- **"Window too small"**. Rosen needs at least 50 columns by 12 rows. Make the window larger and it redraws by itself.
- **Boxes drawn with letters** (+, -, |). Rosen thinks your terminal does not do UTF-8. Check LANG or LC_ALL; or set ROSEN_UNICODE=1 to force the nicer characters.
- **Colors look wrong**. Try another look (**o** , then **Colors**); Mono white uses only your terminal's own colors. Set COLORTERM=truecolor if your terminal supports 24-bit color and Rosen has chosen the 256-color palette; unset it if the colors are off in a terminal that does not.
- **The screen is full of stray characters** after something else wrote to the terminal: **Ctrl-L** redraws it.

- **The bell does nothing.** The terminal decides what a bell does. Check its preferences, or set **Reply alert** to *Marker only* and watch the tabs instead (Chapter 15).
- **Clipboard copy does nothing.** `y` uses the terminal's OSC 52 clipboard feature, which some terminals switch off by default (and some multiplexers filter). Turn it on in the terminal, or select the text with the mouse.
- **Keys that do nothing in the message box.** Some terminals send `Alt-Enter` to themselves; use `ctrl-J` for a new line.

19.5 Starting, files and locks

What Rosen says	What to do
"standard input and output must be a terminal"	Rosen was started with its input or output redirected. Run it directly in a terminal, or use <code>-d</code> to print a page as text.
"already open in another copy (process N)"	Another Rosen is running on the same data file. Go to it, or use another file with <code>-f</code> . If you are sure it is not running, the lock goes away by itself the next time you start: Rosen notices that the process is gone. (A copy that belongs to another user counts as running.) A process number reused by an unrelated program can make an old lock look live; the message names the lock file to delete.
"cannot use the lock file ... Permission denied"	The folder that holds the data file cannot be written by you. Use another file with <code>-f</code> , or fix the folder's permissions.
"... is a folder, not a file"	<code>-f</code> was given a folder. Give it the name of the data file, such as <code>~/rosen-work/rosen.json</code> .
"... is one of Rosen's own files (servers, chats, settings); choose another name."	An export or a saved code block was aimed at a file Rosen keeps for itself. Pick another name or folder.
rosen: FILE: line 4, column 3: unexpected character (or "temperature" must be between 0 and 2 (found 9)., or the file is not valid UTF-8 text), and Rosen stops at start	The data file cannot be read, and Rosen will not replace it (it may hold your keys). The message names the file, the place and what is wrong, and the exit status is 1. Fix the file with an editor, or restore <code>rosen.json.bak</code> , or move the file aside to start with a fresh one.
"NOT SAVED (...)" in red	A file could not be written: the disk is full, or a folder is not writable. The message says which. Fix it and the message goes away by itself after the next successful save.
"N things to know about the settings file: see Checks."	The screen settings file has a line Rosen does not understand. It uses the defaults for those lines. See the Checks page for which.
! 1 alert and the Checks page says your data file or chats folder can be read by other people	Run the <code>chmod</code> command the Checks page prints. Rosen creates its files private; files copied from elsewhere may not be.

19.6 The terminal is in a mess

If Rosen is killed, crashes or loses its terminal, a helper restores the terminal. In the rare case that it cannot (a `kill -9` of both programs at once, say), type `reset` and press Enter, or `stty sane`; the screen may not show what you type until you do.

19.7 Reporting a problem

Please include: the version (`rosen -v`), your terminal and operating system, what you pressed and what you expected, and the contents of `crash.log` if there is one (section 17.3). It does not hold what you typed. For a screen that looks wrong, `rosen -d -s WxH` page prints exactly what Rosen thinks the screen shows, which is easier to read than a photograph.

20. Limits and Known Issues

Rosen is an alpha release. These are the limits it has by design, and the things you should know before relying on it.

20.1 Limits

Thing	Limit
Chats open at once	9 (as many as you like are saved)
Title of a chat	120 characters
A message you type	200,000 characters
Files attached to one message	10 files, each up to 200,000 bytes, 600,000 bytes in all
A chat or text file imported	20 MB
Messages read from one Markdown or text export	10,000
Entries shown for one folder in the file browser	20,000
Default instructions, and each chat's own	20,000 characters
Memory (messages sent with each question)	0 to 10,000 (0 is the whole chat)
Longest reply	0 to 10,000,000 tokens (0 is the server's choice)
Time limit	5 to 86,400 seconds
Output kept from a code block that is run	200,000 bytes; the program is stopped after 60 seconds
Text pasted in one piece	up to 8 MB is received whole (anything beyond it is dropped, never typed as keys); the message box keeps the first 200,000 characters
One line of a streamed reply	16 MiB; a longer one without a line break is an error, and what arrived is kept
Chats counted on the Summary page	the newest 2,000; the page says when it stopped
Window size	50 by 12 at least; 1,000 by 1,000 at most

20.2 What Rosen does not do

- **No Windows.** Linux and macOS only; on Windows use WSL.
- **No images, voices or tools.** Text in, text out. Rosen does not send pictures, and it does not let a model call functions.
- **Servers it has not seen.** It speaks Ollama's `/api/chat` and the OpenAI `/v1/chat/completions` protocol, in their streaming and non-streaming forms. A server with its own variations may answer in a way Rosen reports as "not with JSON" or "not the model list".
- **Character widths are approximate.** Combining marks and some emoji or unusual scripts can be mis-measured and misalign a line. Text from the model has control characters and escape sequences removed before it is drawn.

- **File names must be valid UTF-8.** A file or folder whose name is not (a Latin-1 name made on an old Linux system, say) is listed with replacement characters in the file browser and cannot be opened from Rosen. Rename it first.
- **Tokens are estimates.** Sizes in tokens on the Summary page are about four characters each, not what your model's tokenizer would say.
- **One reply per chat at a time.** You can have many chats replying at once, but each chat has one reply in progress.
- **No encryption of its own files.** They are private to your account (mode 0600), not encrypted. Use an encrypted disk if that matters to you.
- **Chats are not synchronised.** Two computers with two copies of the data do not merge; copy deliberately (section 17.4).
- **Times in saved chats are local clock times.** They carry no time zone, so a chat copied to a computer in another zone shows, and sorts by, the clock time it was written at.
- **The clock comes from date.** If your system's date command is unusual, the time at the top of the screen may be missing.

20.3 Terminals

Rosen needs an ANSI terminal and `stty`. It has been exercised against its own test harness (a pseudo-terminal that sends real key bytes and reads the screen back) on Linux; macOS use is expected to work and has been built for, but this release has not been run on a Mac by its author. If you find that a key, a color or a box character misbehaves in your terminal, `rosen -d` shows what Rosen drew, which tells whether the fault is Rosen's or the terminal's.

20.4 Known issues

- A server that streams a reply without ever sending its end marker is treated as a reply that was cut short, because Rosen cannot tell the difference. The reply is kept and marked, and `r` asks again.
- Resizing the window is noticed within about a second, not instantly.
- Programs a code block starts and leaves running in the background are not stopped when you quit Rosen.
- At the smallest window sizes a form scrolls to keep the field you are on in view, and shows a shorter key-hint line.
- The Sessions page reads every chat file when Rosen starts and when you refresh the list (saving a chat does not). With 3,000 chats of about 20 KB each Rosen starts in a quarter of a second; the Summary page takes under a second. Thousands of chats are fine; a small index file is planned for much larger libraries.
- The screen settings file is rewritten whole when you change a setting; comments you added to it by hand are not kept.

Appendix A. Key Reference

Every key and every click, by where you are. The same lists are in the program: **?** or **F1** opens help for the page or dialog you are in. The keys follow the conventions of Inventory Central, so the two programs feel the same under your fingers.

A.1 On every page

Key	What it does
1 to 6 , 0	Chat, Sessions, Servers, Models, Summary, Checks, Settings.
Tab , Right ,]	The next page.
Shift-Tab , Left , [	The previous page.
v , v	The next or previous view of the page (on Chat: the next or previous open chat).
b	The sidebar off, or on again.
{ , }	The previous or next look. The choice is kept.
? , F1	Help for where you are.
! , F2	Open a shell.
Ctrl-L	Redraw the screen.
q , Ctrl-C	Quit (asks if a reply is still arriving).

The keys above work wherever you are looking at a page. While you are typing into a message, a form, a search or a menu, a letter is a letter: **q**, **b**, **v** and the digits are typed, not obeyed.

A.2 The Chat page

With the message box closed:

Key	What it does
Enter , i	Write a message.
/	Start a command.
n	A new chat on the same server and model, ready for typing.
N	A new chat on a server and model you choose.
v , V , Alt-n , Alt-p	The next or previous open chat.
Alt-1 to Alt-9	Go straight to that chat.
e	Edit the chat's details: title, server, model and instructions in one form.
t , m , p	The same form, with the cursor on the title, the server and model, or the instructions.
d , Delete	Close this chat (it stays saved on the Sessions page).
x	Export this chat.
X	Clear every message from this chat (asks first).
r	Ask again; the old reply is kept until the new one arrives.
s , Esc	Stop a reply that is arriving.
a	Attach text files to the next message.
c	The code blocks in this chat.
y	Copy the last reply.
w	Save everything now.
Up , Down , j , k , PgUp , PgDn , Space	Scroll the conversation.
g , G , Home , End	The first message; the newest text.

A.3 Writing a message

Key	What it does
Enter	Send.
Ctrl-J , Alt-Enter	A new line inside the message.
Esc	Leave the box; the text stays.
Ctrl-C	Stop a reply that is arriving; if none is, leave the box.
Ctrl-F	Attach text files.
Ctrl-X	Remove the last attachment.
Ctrl-R	Ask again.
Ctrl-N	A new chat.

Key	What it does
F1 , F2	Help; open a shell.
Up , Down , PgUp , PgDn	Move in the text; at the first or last line, scroll the conversation.
Home , End , Ctrl-A , Ctrl-E	The start and end of the line.
Ctrl-Left , Ctrl-Right , Alt-b , Alt-f	A word back or forward.
Ctrl-W , Ctrl-U , Ctrl-K	Delete a word back, everything back to the start, or everything on to the end.

A.4 The Sessions, Servers and Models tables

These keys work on all three tables:

Key	What it does
Up , Down , j , k	Move the highlight.
PgUp , PgDn	Move a page.
g , G , Home , End	The first and last row.
h , l , Ctrl-Left , Ctrl-Right	The previous or next column. The current column is the one o sorts and f filters.
o , s	Sort by the current column; again to turn the order round.
O	Choose the column and the direction in a form.
r	Reverse the sort.
Alt-1 to Alt-9	Sort by that column.
/	Search every column; every word must match somewhere in the row.
f	Filter on the current column only.
Esc	Clear the search or filter.
w	Save everything now.
v , V	The next or previous view.

On the **Sessions** page the views are All and one for each server. **Enter** opens the chat; **a** or **n** starts a new one; **e** or **t** edits its details; **c** clones it; **Space** reads it without opening it; **d** or **Delete** deletes it; **x** exports it; **X** exports every chat; **i** imports.

On the **Servers** page: **Enter** or **e** edits; **a** adds; **c** clones; **u** uses it for this chat; **n** starts a chat on it; **t** tests it; **R** asks it for its models again; **Space** shows what is known about it; **d** or **Delete** deletes.

On the **Models** page the views are the servers. **Enter** , **e** or **u** uses the model for this chat; **D** makes it the server's default; **a** types a name the server did not list; **R** asks again; **Space** shows what is known about it.

Note. A capital **R** asks again because a small **r** reverses the sort, as in Inventory Central.

A.5 Summary, Checks and Settings

Key	What it does
Up , Down , j , k	(Summary) scroll; (Checks, Settings) move.
PgUp , PgDn , Space	(Summary) scroll a page.
g , G	(Summary) top and bottom.
R	(Summary) count again; (Checks) check again.
Enter , e	(Checks) go to the page where it can be dealt with.
Enter , Space , e	(Settings) change the highlighted setting.

A.6 Forms, menus and questions

Key	What it does
Tab , Down , Enter	(Form) the next field; Enter on the last text field saves. Tab from the last field reaches the buttons.
Shift-Tab , Up	(Form) the previous field.
Ctrl-S , F2	(Form) save, from any field.
Enter , Space	(Form, on a button) press [Save] or [Cancel]; Left and Right move between them.
Left , Right	(Form) change a choice without opening it.
Enter on a choice	Open its menu; typing narrows it.
Ctrl-N , Ctrl-P	(Form) move down and up in a menu or a list of suggestions.
Ctrl-B	(Form) browse for a folder, where the field is one.
Ctrl-T	(Server form) test the connection with what is typed.
Esc	Close a menu; a second Esc cancels the form.
Up , Down , j , k , Home , End , Enter , Space , Esc , q	(Menu) move, choose, cancel.
y , Enter	(Question) yes.
n , Esc , q	(Question) no.

A.7 The file browser

Key	What it does
Up , Down , PgUp , PgDn , Home , End	Move the highlight.
Enter	Open a folder, or pick the file (or the marked files).

Key	What it does
Right	Open the highlighted folder.
Left , Backspace	Go up one folder (Backspace first undoes typed letters).
Space	Mark or unmark a file, when several can be picked.
letters	Narrow the list to the names containing them; Ctrl-U or Esc clears.
Tab	Show or hide names that start with a dot.
Ctrl-G	Type a path to go to.
Ctrl-S	When choosing a folder: use the folder you are in.
Esc , Ctrl-C	Cancel.

A.8 Code blocks and the text viewer

Key	What it does
Up , Down , 1 to 9	(Code menu) pick a block.
Enter	(Code menu) read it.
y	(Code menu) copy it.
w	(Code menu) save it to a file.
r	(Code menu) run it (asks first) and show what it prints.
s	(Code menu) open a shell with \$ROSEN_CODE set to the block's file.
Up , Down , j , k , PgUp , PgDn , Space , g , G	(Viewer) scroll.
Esc , Enter , q	(Viewer) close.

A.9 The mouse

Left clicks and the wheel, while Mouse is On in Settings. With it Off, the program never asks the terminal for the mouse, and a click does nothing.

Action	What it does
Click a page	Go to it, in the sidebar or the top bar.
Click a row	Highlight it. Click the highlighted row again to open it, as Enter does.
Click a heading	Sort by that column; click again to turn the order round.
Click a view or a chat tab	Open that view or chat.
Click the message box	Start writing.
Click a hint at the bottom	Press the key it names.
Click a field in a form	Go to that field; a choice opens its menu.
Click [Save] or [Cancel]	Press the button.

Action	What it does
Click an entry of an open list	In a form's menu or list of suggestions: take it. In a settings menu or the file browser: the first click highlights it and the second takes it.
Wheel	Scroll three rows. In a menu or a list it moves the highlight.
Click outside a dialog	Nothing.

The commands you can type in a message are in section [5.9](#).

Appendix B. Coming from llm-tui

Rosen is llm-tui, rebuilt. It keeps what llm-tui did (chat with Ollama and OpenAI-compatible servers, streaming replies, saved sessions, exports) and adds several chats at once, a shell, code blocks you can run, import, a file browser, looks, notifications, and a manual. Your servers, keys and chats come with you.

B.1 Bringing your data across

llm-tui kept everything in `~/.llm-tui/` (or `$LLM_TUI_HOME`): `config.json` for the servers and settings and a `sessions/` folder with one file per chat. Rosen uses the same formats under new names, so moving is a copy:

```
mkdir -p ~/.local/share/rosen
cp -p ~/.llm-tui/config.json ~/.local/share/rosen/rosen.json
cp -Rp ~/.llm-tui/sessions ~/.local/share/rosen/sessions
chmod 600 ~/.local/share/rosen/rosen.json
chmod 700 ~/.local/share/rosen/sessions
```

Start Rosen and your servers, active server, model choices, instructions and numbers are as you left them, and the Sessions page lists every chat. Leave `~/.llm-tui/` where it is until you are happy; Rosen never touches it. If the Checks page says a file can be read by other people, run the `chmod` command it prints.

Two things from the old configuration are not carried over: the **theme** (choose *Classic dark* or *Classic light* under Settings, Colors, or step to them with `{` and `}`, to get the old colors back) and the **sidebar** switch (the sidebar now appears by itself in a window of 100 by 24 or more; `b` switches it off and on, and the **Sidebar menu** setting remembers the choice). Both are kept in `rosen.json` untouched, in case you go back.

B.2 Keys that moved

llm-tui used Ctrl keys for its screens. Rosen puts the pages on the number keys, and the other everyday keys on plain letters, as Inventory Central does, so that Ctrl keys are free for the message box. Rosen also answers the mouse (Chapter 3).

llm-tui	Rosen
<code>Ctrl-P</code> , <code>F2</code> Servers	<code>3</code> . (<code>F2</code> now opens a shell.)
<code>Ctrl-L</code> , <code>F3</code> Models	<code>4</code> . (<code>Ctrl-L</code> now redraws the screen.)
<code>Ctrl-O</code> , <code>F4</code> Sessions	<code>2</code> .
<code>Ctrl-G</code> Settings	<code>0</code> .
<code>Ctrl-T</code> , <code>F5</code> Dark or light	<code>{</code> and <code>}</code> step through the fifteen looks from anywhere; or Settings, Colors; or <code>/</code> look.
<code>Ctrl-B</code> Sidebar	<code>b</code> , from any page; the choice is kept.
<code>Ctrl-N</code> New chat	<code>Ctrl-N</code> in the message box; <code>n</code> on the Chat page; <code>a</code> or <code>n</code> on the Sessions page.

llm-tui	Rosen
Ctrl-R Ask again	Ctrl-R in the message box; r on the page.
Ctrl-S Save as JSON	x (export) and choose <i>Chat file (JSON)</i> ; or <code>/export json</code> .
Ctrl-X Export	x on the Chat page and on the Sessions page. (Ctrl-X in the message box now removes an attachment.)
Esc , Ctrl-C Stop the reply	Esc or s with the box closed; Ctrl-C in the box.
F1 Help	? or F1 : help for where you are.
Ctrl-Q Quit	q , or Ctrl-C with the box closed.

The text-editing keys (**Ctrl-A** , **Ctrl-E** , **Ctrl-W** , **Ctrl-U** , **Ctrl-K** , word movement) are unchanged.

B.3 Commands that changed

Almost every slash command works as before (section 5.9). The differences:

- `/theme [dark|light]` still works, as another name for `/look`; `dark` and `light` choose the two classic looks.
- `/set sidebar` is gone; use **b** or the Sidebar menu setting.
- `/save [path]` used to write a JSON file. In Rosen chats are saved as they change, so `/save` just saves now; to write a file use `/export json [path]`.
- `/exportall [folder]` asks for the folder in a form instead of taking it as an argument.
- `/server` takes part of a name, as before, or a number counting from the first server added. With nothing, it opens a menu.

B.4 Command line and environment

llm-tui had only `--help` and `--version`. Rosen follows the POSIX conventions and adds options (Chapter 16). `LLM_TUI_HOME` becomes `ROSEN_FILE` (or `-f`, or `XDG_DATA_HOME`); `LLM_TUI_COLORS=256` has no equivalent, because Rosen asks for 24-bit color only when `COLORTERM` says the terminal has it, so unsetting `COLORTERM` gives the 256-color palette.

B.5 What is better, and what you might notice

- A damaged `config.json` no longer risks your keys: Rosen refuses to start rather than replace a file it could not read, and tells you which line is wrong.
- A reply that is cut off is now an **error**, not a quiet success, and what arrived is kept.
- Asking again keeps the old reply until the new one has arrived.
- Exporting over an existing file asks first, and a name with no extension gets the right one for the format.
- A server address ending in `/v1` or `/api` is cleaned up once, not twice.
- A line starting with `/` that is not a command is no longer thrown away.
- Pasting a large block into the message box can no longer be cut short by a pause.

- Temporary files are made private from the start.
- A killed Rosen, or a closed terminal, puts the terminal back and removes its lock.

Appendix C. Standards and Conformance

Rosen is written to the same rules as the other Electric Shepherd terminal tools: the Rust standard library only, no unsafe code, no crates; POSIX.1-2024 (SUSv5) conventions for the command line; and Unix standards for files and the terminal. A conformance file (`CONFORMANCE.md`) in the source records the details; this appendix summarises it.

Item	Statement
Mode	Extended. Everything the POSIX Utility Conventions specify for an invocation like Rosen's behaves as specified; the extras are listed below.
Standard	POSIX.1-2024 (IEEE Std 1003.1-2024; Open Group Base Specifications Issue 8), Utility Conventions (XBD chapter 12) and the chapters named here. Rosen has no POSIX counterpart utility, so the conventions are the whole of what applies.
Caveat	Only a whole system conforms to POSIX. The claim is that Rosen matches the specification pages named, and nothing wider.

C.1 Command line

- Options** are single letters preceded by `-`, may be grouped (`-dn`), and take their argument attached or separate (`-ffile`, `-f file`); `--` ends the options; a lone `-` is an operand. At most one operand, which is the data file.
- Exit status** is 0 on success (including `-h`, `-v` and a finished `-d`), 1 for an error while running, and 2 for a usage error. Diagnostics go to standard error, each starting `rosen:`. Nothing is written to standard error during normal use.
- Environment** honoured: `ROSEN_FILE`, `XDGLDATAHOME`, `XDGLCONFIGHOME` (absolute paths only), `HOME`, `COLUMNS`, `LINES`, `LC_ALL`, `LC_CTYPE`, `LANG` (in that order of precedence, only to choose box-drawing or ASCII), `NO_COLOR`, `COLORTERM`, `POSIXLY_CORRECT`, `SHELL`, `PATH`, `TMPDIR`, and `ROSEN_ASCII`, `ROSEN_UNICODE`, `ROSEN_NO_SPLASH`, `ROSEN_TODAY`, `ROSEN_TIME`. `TERM` is ignored (see E4).

C.2 What it asks the system

Rosen declares no libc bindings. Besides the standard library it runs these programs as child processes:

Program	Used for
stty	Raw mode and its undoing, and the window size (<code>stty -g</code> , <code>stty raw -echo ...</code> , <code>stty size</code>).
date	The local date and time.
sh	Asking whether a process still exists (<code>kill -s 0</code>), restoring the terminal from the helper, opening your shell, and running a code block when you ask .
kill	Stopping a code block that has run out of time, with the processes it started.
ps	Asking whether a process still exists when <code>kill -s 0</code> says no (it says no for a program that belongs to another user, just as for one that is gone), so that a lock held by someone else's running copy is not broken.
curl	Reaching <code>https://</code> servers only (extension E7).

Files are made with mode 0600 and folders with 0700, written through a temporary file that is synced and renamed over the old one, and a read-back check refuses to write what could not be read again (section 17.2).

C.3 Extensions

#	Extension
E1	GNU-style long options: <code>--file</code> , <code>--demo</code> , <code>--dump</code> , <code>--size</code> , <code>--keys</code> , <code>--colors</code> (or <code>--look</code>), <code>--page</code> , <code>--no-splash</code> , <code>--help</code> , <code>--version</code> , <code>--licence</code> (or <code>--license</code>), with the <code>--file=PATH</code> forms.
E2	Options may follow operands (<code>rosen chats.json -d</code>); <code>POSIXLY_CORRECT</code> turns this off.
E3	<code>--restore-guard SETTINGS LOCK PID</code> : an internal form with which Rosen starts its own terminal-restore helper. Not for people to type; it must be the first argument.
E4	Terminal control by fixed ECMA-48 and xterm sequences, not a terminal database: alternate screen, cursor visibility, autowrap, bracketed paste (?2004), focus reporting (?1004), mouse reporting in the SGR form (?1000 with ?1006, only while the Mouse setting is On), SGR colors up to 24-bit, and the OSC 52 clipboard write. <code>TERM</code> is not consulted; any xterm-compatible terminal works.
E5	Keys decoded from VT and xterm sequences, with UTF-8 text input; mouse reports (left button and wheel) decoded from the SGR form.
E6	The <code>ROSEN_*</code> environment variables and <code>NO_COLOR</code> .
E7	HTTPS through the system <code>curl</code> , because the standard library has no TLS. The key is handed to <code>curl</code> on standard input, and <code>curl</code> is started with <code>-q</code> . This is the one place where Rosen depends on a program that POSIX does not specify.
E8	Programs you start on purpose: your shell (<code>\$SHELL</code>) and the runner for a code block (<code>sh</code> , <code>bash</code> , <code>zsh</code> , <code>python3</code> , <code>perl</code> , <code>ruby</code> , <code>node</code> , <code>awk</code>), each only when you ask and, for code, after you confirm.

C.4 What it does not do

- No job control while it runs: raw mode makes `ctrl-c` and `ctrl-z` ordinary keys. `ctrl-c` quits (asking first if a reply is arriving) or, in the message box, stops a reply or leaves the box; `ctrl-z` does nothing.
- No SIGWINCH: the window size is polled once a second.
- Character widths are approximate; wide characters and combining marks can misalign a line.
- The lock is a process number checked with `kill -s 0` and `ps -p`: it cannot see another machine that shares the data folder, and it protects against a second Rosen, not against other programs editing the files.
- Linux and macOS only; tested on Linux x86-64.

C.5 How it is checked

`cargo test` runs 293 tests (and `cargo test -- --ignored` one more, a timing test with 3,000 chats): 222 unit tests (including the option parser, the data loaders, atomic private saves, the lock, the key and mouse decoder, the looks, the screen layouts and the HTTP client against local servers) and 71 key-driven flow tests that press keys and click, draw the screen and check the saved files, including random key-and-mouse runs and a click on every cell of every page at every window size. A pseudo-terminal test (`tests/pty_smoke.py`, 83 checks) drives the release binary against a fake model server and checks raw mode in and out, a real conversation, export and import, the bell, the shell hand-over, a second copy being refused (also when the first belongs to another user), the restore helper after SIGKILL and SIGTERM and after a SIGKILL under a shell with job control, the exit statuses, COLUMNS and LINES, the mouse reports being asked for and given back (also after SIGKILL), and `-p`, `-L` and `-c`. `cargo clippy --all-targets` and `cargo fmt --check` are clean.

Appendix D. Release Notes

D.1 v0.1.1a (alpha), 22 October 2026

Rosen now works the way Inventory Central does: the same pages, keys, lists, forms and hints, and a mouse. Crate `rosen-tui 0.1.1-alpha`.

New

- **The mouse.** Click a page, a row (click the highlighted row again to open it), a heading to sort, a view, a chat tab, a field, a button, a menu entry or a key hint; the wheel scrolls three rows. It is a real switch: with **Mouse** Off on the Settings page the terminal is never asked to report the mouse (Chapter 3).
- **Views.** Sessions has a view for All and one for each server; Models has one for each server; on Chat the open chats are the views. `v` and `V` step through them.
- **Tables as in Inventory Central.** A current column (`h` and `l`), sorting by `o`, `O`, `r` and `Alt-1` to `Alt-9`, search (`/`) and filter (`f`), a "Sorted by" line and a count, and `>` on the highlighted row.
- **Fifteen looks**, seven of them new: Winter, Harbor, Dusk, Spring, Autumn, Ember and Samhain, joining Full color, Blue/teal, Shepherd, the three monochrome looks and the two classic ones. `{` and `}` step through them from anywhere (Chapter 14). Eight of the looks are held to fixed contrast measures by a test.
- **Forms with buttons.** `[ Save ]` and `[ Cancel ]`, `Enter` to walk through the fields, menus that narrow as you type, `Ctrl-N` and `Ctrl-P` to move in them, and `Ctrl-S` or `F2` to save from anywhere.
- **The sidebar and the bottom three lines.** `b` switches the sidebar; the last three lines are the information line, the message line and the key hints for the page, which end `{}` look `b sidebar ? help q quit`.
- **Clone a chat** (`c` on the Sessions page) and **read a chat without opening it** (`Space`).
- **Settings:** Splash screen, Start on page, Sidebar menu and Mouse, and the `-p` option to open a page for one run; the terms of use are built in (`rosen -L`).
- Help for every page and dialog lists the new keys, with a section on the mouse.

Keys that moved

In v0.1a	In v0.1.1a
Chat: <code>x</code> closes the tab	<code>d</code> (or <code>Delete</code>) closes it; <code>x</code> exports.
Chat: <code>e</code> exports	<code>x</code> exports; <code>e</code> edits the chat's details in one form.
Chat: <code>t</code> renames, <code>m</code> chooses the model	<code>t</code> , <code>m</code> and <code>p</code> open the details form with the cursor on the title, the server and model, or the instructions.
Chat: <code>[</code> and <code>]</code> change chat	<code>v</code> and <code>V</code> (or <code>Alt-n</code> and <code>Alt-p</code>); <code>[</code> and <code>]</code> now change the page.
Servers: <code>Enter</code> uses the server	<code>u</code> uses it; <code>Enter</code> edits it.

In v0.1a	In v0.1.1a
Servers, Models, Summary, Checks: r asks again	R ; a small r reverses the sort.
Models: d makes the default, n adds a name	D and a .
Tables: [and] change the column	h and l .

Fixed while preparing this release

An independent review of the keys, forms and mouse found 14 defects, none a crash and none losing data. All are fixed.

- **ctrl-S** and **F2** save a form while one of its menus is open; **Up** and **Down** move one line, not two, in a field of several lines.
- Deleting another chat no longer switches the chat you are in. Clearing a chat stops a reply that is arriving.
- Long lists in a dialog (the code menu, a settings menu) scroll and keep the highlight in view.
- **ctrl-L** redraws the whole screen instead of leaving it blank.
- A click on the frame of an open menu no longer presses **[Save]** under it; a click on a file in the file browser picks that file even while a path is being typed; one click opens the highlighted row after the last row was deleted.
- With **-c**, the Colors setting works.
- A chat whose server was deleted is shown as (deleted) and no longer sends its old model name to another server.
- The message about a reply still arriving names the key that works where you are; in a window of 50 by 12 the "Run this code?" question and the alert badge no longer cover other text; the wheel moves three rows on the Settings page too; with Mouse Off, a mouse report does nothing at all.

Known limits

As for v0.1a (Chapter 20): Linux and macOS only, text only, no tool calls, no encryption of its own files. Tested on Linux x86-64; built, not yet run, on macOS by its author. The mouse was tested with a pseudo-terminal and a click on every cell of every page at every window size, but not with a real mouse.

D.2 v0.1a (alpha), 21 October 2026

The first release of Rosen, the successor to llm-tui. Crate `rosen-tui 0.1.0-alpha`; the program is `rosen`.

New

- **Several chats at once**, each in a tab with its own server and model, up to nine open and any number saved.
- **Several servers**, with a table, sorting and search; testing; a form with a connection test; Ollama and every OpenAI-compatible server.

- **Pages and a sidebar** in the Electric Shepherd style: Chat, Sessions, Servers, Models, Summary, Checks and Settings; a top-bar layout for windows from 50 by 12; contextual help on every page and dialog.
- **Eight looks:** Full color, Blue/teal, Shepherd, Mono white, Mono green, Mono amber, and llm-tui's Classic dark and light; previewed as you choose.
- **A start-up screen** with a flock of sheep, and a clock if you want one.
- **New-reply alerts:** a marker on the tab, a count in the sidebar, and the terminal bell, when a reply ends out of sight.
- **Code blocks:** found, numbered and shown on their own; read, copy, save or run (after asking), with a time limit and an output limit.
- **A shell on demand,** with the newest code block waiting in `$ROSEN_CODE`.
- **Export** to Markdown, plain text, HTML or a chat file, one chat or all; **import** of chat files, Markdown and text; **attach** text files to a message; all through a built-in file browser.
- **Checks and Summary pages:** what is worth knowing about your files and servers, and counts from your chats.
- **A command line** that follows the POSIX conventions, with `-d` to print any page as text and `-D` for sample data and a built-in echo server.
- **This manual.**

From llm-tui

- Reads llm-tui's data files unchanged (Appendix B).
- A damaged data file is never replaced; a reply cut short is an error and is kept; asking again keeps the old reply until the new one arrives; exporting over a file asks first; a server address loses one `/v1` or `/api`, not both; a line that starts with `/` but is not a command is put back in the box; large pastes are not cut off by a pause; temporary files are private from the start.
- A killed Rosen, a crash or a closed terminal gives you your terminal back, and frees its lock.

Fixed while preparing this release

- A bad `-c` look name now lists every look; it listed two old names and left out three.
- In windows of about 70 columns the product name no longer overdraws the end of the message line.
- The crash log no longer records letters typed into a message, a form or a search.
- Sample-data mode (`-D`) no longer runs a code block.
- `/server 2` (a number, as llm-tui took it) works again.
- A large paste into the middle of a long message is one move, not one shift per character.
- Help for `/set` named options the command does not have; it now names the real ones.
- Rename (`t`) and export (`x`) on the Sessions page do nothing, and say why, when the highlighted chat cannot be opened (nine chats are open); they used to act on the chat you were last in.
- A server that sends one endless line can no longer fill the computer's memory: a line of a reply is limited to 16 MiB, and a header, chunk-size or trailer line to 64 KiB.

- A paste of more than 8 MB is cut and the rest dropped, never typed as keys; a long paste is no longer searched again from the start for every piece that arrives.
- A code block that leaves a program running in the background no longer loses what it printed.
- If a background thread (a reply, a model list, a connection test or a code run) ever failed through a fault in Rosen itself, the chat or page would have waited for an answer for ever; it now shows "Rosen hit an internal error" and carries on.
- A chat title or instructions containing a line break can no longer start a heading or a message of its own in a Markdown or text export.
- Data files follow the JSON number rules (`01`, `1.` and `- .5` are refused), and a lone high-surrogate escape no longer swallows the escape after it.
- A wide character drawn across two others no longer leaves a row one column short.
- Every part of the program, the build script and the screenshot tool now forbids `unsafe` code, not only the library.
- A copy of Rosen running under another user's name is no longer mistaken for a dead one: the lock is not taken over. (Rosen asks `ps` as well as `kill -s 0.`)
- In a window of 50 by 12 the key-hint line of a form and two help texts were cut short; they now fit, in every form and at every size.
- The crash log is only appended to an ordinary file, never through a link or into a folder, and says why when a report cannot be written.
- Saving a chat no longer reads every other chat file again.
- An export or a saved code block can no longer be written over Rosen's own files (servers, chats, settings, lock, backups, crash log), by name or through a link.
- A folder that cannot hold the lock file is named in the message, a folder given to `-f` is refused with a hint, and "color" is spelled one way throughout.

Known limits

Chapter 20 lists them. In short: Linux and macOS only, text only, no tool calls, no encryption of its own files, window size noticed within a second, and an alpha's worth of testing; tested on Linux x86-64 and built, not yet run, on macOS by its author.

Appendix E. Glossary

Word	Meaning
Active server	The server new chats start on, marked * on the Servers page.
Alert	A check that needs attention; counted by the ! N alerts badge.
API type	Which of the two protocols a server speaks: Ollama's, or the OpenAI-compatible one that llama.cpp, LM Studio, vLLM and others use.
Bell	The character a terminal rings, flashes or notifies for. Rosen sends it when a reply ends out of sight, if you ask for it.
Chat	One conversation, with its own server, model and messages. Called a <i>session</i> in the files and on the Sessions page.
Chat file	The JSON file that holds one chat. The format is the same as llm-tui's.
Code block	A passage fenced with three backticks in a reply, drawn on its own and numbered.
Data file	rosen.json: your servers, keys and the numbers sent to the model.
Echo server	The built-in demo server, which quotes your message back. Only in -D.
Instructions	The standing text sent first with every chat (the <i>system prompt</i>). A chat can have its own.
Current column	The column of a table whose heading is highlighted: the one o sorts, f filters and is shown in full.
Look	A set of colors for the whole program. There are fifteen; { and } step through them.
Lock	A small file that stops a second copy of Rosen from using the same data file.
Memory	How many of the latest messages are sent with each question.
Mouse reporting	The terminal feature that tells Rosen about clicks and the wheel. Rosen asks for it only while the Mouse setting is On.
Model	The language model a server runs, such as llama3.2:3b.
OSC 52	The terminal feature Rosen uses to put text on the clipboard.
Raw mode	The terminal state in which Rosen sees every key as it is pressed, not line by line.
Restore helper	A second copy of Rosen that puts the terminal back if the first one dies.
Server	A program that runs a model and answers over the network.
Splash	The start-up screen.
Streaming	Showing a reply as it is written rather than all at once.
Temperature	How adventurous the model is when it picks words: 0 is steady, 2 is wild.
Token	A piece of a word; what models count in. Roughly four characters.
Top-p	Limits the choice of words to the likeliest ones that add up to this share.
Unread marker	The dot on a tab that says a reply ended there while you were looking elsewhere.
View	One of the named lists a page can show: All or one server on Sessions, one server on Models, one chat on Chat. v and v step through them.

