LET IT SEARCH, GIVE IT KNOW-HOW

Youth AI Lab · Workshop 3 of 6

Build your own chatbot



Co-funded by the European Union

❓ The question of the series

A chatbot only answers from **what it reads** before it answers. Every workshop of the series is **one answer** to the same question.

> **What enters the model's context, when, and who decides?**

📥 **What?**
rules, documents, search results, live data

⏱️ **When?**
always, at startup, or for each question

🤦 **Who decides?**
you, a search engine, or the model itself

🧭 Today: what if the model searched by itself?

1. Instructions	2. Knowledge, RAG	3. Search, skills today	4. Tools	5. Wiki	6. Fine-tuning
🎭	📚	🔍	📡	🗺️	🧪

Mode B: nobody picks, everything enters.

RAG: a search engine picks, before the model.

Mode C: **the model** picks, step by step, and you see each step.

🗓️ The plan: 2 hours

👋 Welcome	🔎 Search	📥 Let a document in	🗂️ Your skill	🏁 Closing
20 min	20 min	35 min	30 min	10 min

📄 Follow on your screen: the `README.md` of your Codespace, or the one-page sheet.

💻 Your workspace: one Codespace per pair

A **Codespace** is VS Code in your browser: **nothing to install**. Everything you need is already inside.

> ⏱️ **Before you open it**, once per GitHub account: **github.com/settings/codespaces → Default idle timeout → 120 → Save**. It only applies to the Codespaces you create after.

1. Open the **Codespace link** of the workshop, at the top of its README
2. In the terminal, type `workshop-login` and paste the key of your pair
3. Wait for **Key OK**, then type `workshop bare`. First time: choose a theme, then ↓ **to Yes, I trust this folder** (Enter alone exits). To leave: **Ctrl+C twice**
4. Then type `workshop-start` and the number of the workshop. With a save from last time, add `--from my-work.zip`

> ⚠️ **The key is not shown** while you paste it: that is normal. If the Codespace stops after a long pause, reopen it: **nothing is lost**.

💻 No Codespace? The same repository opens on your own computer: dev containers guide, Part 3.

🐳 Bring your work, start Open WebUI

1. Drag `my-work.zip` (the newest) into the file list, then `workshop-start 3 --from my-work.zip`. Check the `saved on` date.
2. Second terminal: `openwebui start`. Do not wait: you need it at the closing.
3. Have the **address of the page of your question 12** ready: it enters your chatbot today.

> 🚫 **Never make port 3000 public.** This Open WebUI has no accounts: anyone with the link could use your key.

🌡️ The gauge: how full is the context?

The **context** is everything the model reads before it answers. Its maximum size is the **context window**. A line above the input box shows **how full** it is.

🪟

200,000 tokens

the size of the window (a token is a piece of a word)

💶

Every message pays

for everything in the window

🤝

Same model for all

so that the pairs can compare

👁️ **Watch the gauge** at every change: it shows what entered the context.

📏 How to score an answer

Each answer gets **0, 1 or 2**. The score rewards **honesty** as much as knowledge.

2

Right, and it says **where it comes from**. Or "I don't know" when nobody knows.

1

Honest but not enough: no source, only partly right, or "I don't know" although the answer exists.

0

Wrong or **invented**.

💡 An **invented answer** is worse than **"I don't know"**.

🔎 Let it search

Who picks, and how can you check?

20 min · mode `c`

🔎 The model searches like a person

📂 **List**
`ls` a folder

🔍 **Search**
`grep` a word

📖 **Read**
a summary, then a full document

> Nothing is loaded in advance. When a search finds nothing, or nothing that answers, the model **sees it**: the failure is visible.

🎬 Your turn: questions 3 and 7

1. `workshop C`, then question 3 (Bouches-du-Rhône): its answer is only in a full order.
2. Press **Ctrl+O** to unfold the steps (again to come back): which files did it open? Open the file it read in `corpus/text/`, **Ctrl+F**: does it say the same?
3. Look at the gauge, then question 7. Which documents does it find? What does it say?

📏 Measured (2026-10-30): question 3 at about **23,000 tokens**; mode B: about 100,000, the whole kit, whatever the question.

📥 Let a document in, and vouch for it

The model proposes, a person decides

35 min · `corpus-add` · `summarize` · `validate-summary`

Four steps of curation

1. Add	2. Draft	3. Check	4. Sign
`corpus-add`	`summarize`	you, every row, against the original	`validate-summary`
the original, unchanged	summary + table		your name

> First the document of Benevento (question 11, number **80**), then the page of **your** question 12 (number **81**).

📚 Checked documents for the *Knowledge* box: Open WebUI guide, Step 1 · OpenCode guide, Part 4

🎬 Your turn: add, draft, check, sign

1. The `corpus-add` line of `inbox/README.md`, then `corpus-add "<your page>" --name <short-name>`.
2. Open `corpus/text/81-…md` : is your fact in it?
3. `workshop C` : `summarize corpus/81-…`. Look for `Skill(summarize)`. Accept with **1. Yes**.
4. Check every row of `summaries/checks/81-….md` with `Ctrl+F`.
5. `Validate summary 81. I checked every row. My name is <name>.`

⚠️ Never choose an option that stops asking: you would no longer see what it writes.

🖋 A person signs, not the model

`status: draft`

no person has checked it yet

`verified: … human:maria-rossi`
`status: stable`

a person checked every row

❓ Why is it a person's name, and not the model's, in the `verified` line?

🗂️ Write your own skill

An activity sheet for the chatbot

30 min · `.claude/skills/`

🗂️ System prompt or skill?

	System prompt (`AGENTS.md`)	Skill (`SKILL.md`)
When it is read	always, with every message	only when a request needs it
What it says	who the chatbot is, the rules for every answer	how to do one task, step by step
Who decides	you: it is always there	the model, from the description
Cost in the context	the whole text, every time	a few lines, until it is used

💡 The **description** is the trigger.

📚 Skills in a chatbot of your own: your chatbot in Open WebUI (end of this workshop) · with a group, by `chatbot-export` then `openwebui import`: Open WebUI guide, Part 6, Step 2 · another way: OpenCode guide, Part 5 - Skills

🎬 Your turn: a skill that triggers

1. Read `.claude/skills/summarize/SKILL.md`: name, description, steps, rules.
2. `Add this to the summaries: …` → `Skill(source-note)` asks for the **source**. A rumour? It names the summary it contradicts, and asks you.
3. Create `.claude/skills/<name>/SKILL.md`: `rumour-check`, `quiz-maker`, `explain-for-kids`…
4. **Leave and start** `workshop C` again.
5. A request that should trigger it: `Skill(<name>)` (for `rumour-check`, then a search: open the document it cites). And `Hello, who are you?`, which should not.

🔁 Change the description until it triggers **when it should, and only then**.

🏁 Closing: column C, your chatbot, your save

1. Leave the conversation (**Ctrl+C twice**), predict, then `replay C` and `score C 3 8 11 12`. Compare the rows with columns B and RAG.
2. Second terminal: `openwebui chatbot --name "<your chatbot>"`, then your question 12. Your skill comes along, but its whole text goes with **every** message: there, the description triggers nothing.
3. `workshop-save` → **Download...** `my-work.zip`: your documents, your summary, your skill.

> 📝 **Exit card**, alone: (1) In mode C, who decides what the model reads, and how can you check it? (2) Why a person's name in the `verified` line? (3) When does the model open a skill?

📏 Scores: **2** right and sourced · **1** honest but not enough · **0** wrong or invented. Only some questions are scored today: read the rows, not the Total. Saved twice? Bring the newest, maybe `my-work (1).zip`, and the address of the fire danger page of your territory for workshop 4.

🆘 If you are stuck

Every command explains itself. From workshop 3, a few **slash commands** inside the chatbot show what it is using.

❓ In the terminal

`workshop help`

`replay --help`

`score --help`

🔍 In the chatbot

`/context` what fills the window

`/status` the key in use

`/mcp` the tools (from workshop 4)

Workshops 1 and 2: they answer "Unknown command". Watch the gauge above the input box.

🤦 Something breaks?

Ask the facilitator. Your files **stay in the Codespace**.

💡 To leave a conversation: **Ctrl+C twice**. In an open `.md` file, to read it as a page: **Ctrl+Shift+V** (**Cmd+Shift+V** on a Mac). In the terminal, copy / paste: **Ctrl+Shift+C / Ctrl+Shift+V** (**Cmd+C / Cmd+V** on a Mac). To change mode: `workshop <mode>` .