

Learning Radar: a second brain that explains what to learn next

Trace learning notes to a goal, then build a source-backed next-step plan.

Build

Check

Try independently

The situation and your goal

Situation

You want to build a Telegram community bot. Your learning notes cover JSON basics, n8n webhooks and Telegram commands. A superseded note incorrectly claims a standard bot can initiate a private conversation. A second goal, building a website, needs different notes.

Keep this artifact

An Obsidian-compatible learning vault, a cited next-step plan and a missing-evidence test sheet.

Understand the mechanism

01

Links express relationships

In Obsidian, notes and internal links form a graph. The graph makes relationships visible; it does not itself reason or call an AI. Our animated crawler is a teaching simulation of evidence selection.

02

Retrieval comes before an answer

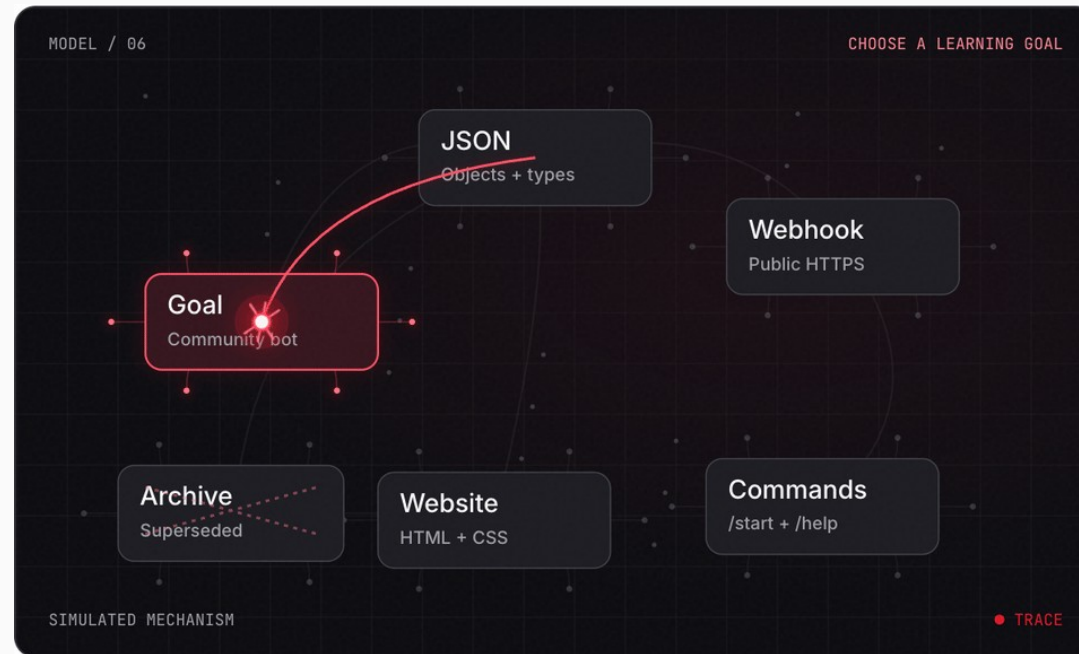
First find the notes relevant to a question, then use those notes as the answer context. Providing every note can bury useful evidence or include stale facts.

03

A source is more than a filename

Record what the note says, when it was updated and whether it conflicts with another note. Prefer verified current evidence and explain the conflict instead of blending incompatible facts.

Explore before you build



Open the Learning Radar vault

1

Open evidence-garden-vault from the practice kit. Read Learning goal, JSON basics, n8n webhook, Telegram commands, Bot archive, and Website basics.

2

Check: Explain which notes are learning fixtures and which claim is superseded.

3

Hint: Inspect the linked source. Our crawler is a supplied-note teaching simulation, separate from Obsidian Graph View.

Link concepts to goals

1

Link Learning goal to `[[JSON basics]]`, `[[n8n webhook]]` and `[[Telegram commands]]`. Link n8n webhook to JSON basics. Link Website basics to the website goal.

2

Check: Graph nodes open real notes with sources.

3

Hint: Inspect the linked source. Our crawler is a supplied-note teaching simulation, separate from Obsidian Graph View.

Ask two goal questions

- 1 For a Telegram bot, select JSON, webhook and commands. For a website, select HTML, CSS and mobile preview from Website basics.
- 2 Check: The selected evidence changes with the goal.
- 3 Hint: Inspect the linked source. Our crawler is a supplied-note teaching simulation, separate from Obsidian Graph View.

Reject the stale bot claim

1

Compare Bot archive with Telegram commands. Exclude the claim that a standard bot can initiate a private conversation; the user must start it or add the bot to a group.

2

Check: The false archived claim is flagged, not blended into the plan.

3

Hint: Inspect the linked source. Our crawler is a supplied-note teaching simulation, separate from Obsidian Graph View.

Write a cited learning plan

1

Ask AI to use only your selected excerpts. Require filenames and supporting facts. For the bot goal, propose JSON basics, then public HTTPS webhook setup, then bot commands.

2

Check: Every recommended step has a reason grounded in the notes.

3

Hint: Inspect the linked source. Our crawler is a supplied-note teaching simulation, separate from Obsidian Graph View.

Test missing evidence

- 1 Ask whether your notes prove an offline model will run on your laptop. Then change your goal to a website.
- 2 Check: Offline model requirements stay unknown; the website answer uses website notes.
- 3 Hint: Inspect the linked source. Our crawler is a supplied-note teaching simulation, separate from Obsidian Graph View.

When it goes sideways

01

Beautiful graph, weak recommendation

Connections were mistaken for relevant evidence. Try: List supporting facts for the selected learning goal.

02

It repeats the archived bot claim

Keyword matches do not establish current source quality. Try: Mark the archive superseded and cite the current Telegram documentation.

03

It promises offline model support

No supplied note establishes hardware or model requirements. Try: Return insufficient evidence and identify what needs researching.

Review your evidence

- 1 Vault links concepts to a learning goal.
- 2 Bot and website goals select different note subsets.
- 3 The false bot-initiation claim is flagged as superseded.

Review your evidence

- 1 Learning sequence cites source facts and explains prerequisites.
- 2 Offline model claims produce insufficient evidence.
- 3 Changing the goal changes the recommendation.

What does Obsidian Graph View do?

- 1 Automatically chooses what you must learn.
- 2 Visualizes notes and their links.
- 3 Guarantees every connected fact is current.

An archived note says a standard bot can initiate a private chat. Current documentation contradicts it. What should you do?

- 1 Blend both claims.
- 2 Exclude the archived claim and record the current source.
- 3 Keep whichever claim makes the project easier.

Build with a clear contract

Use only these supplied learning excerpts. Goal: build a Telegram community bot. JSON basics.md: understand objects, strings and booleans before mapping webhook payloads. n8n webhook.md: Telegram delivery needs a publicly reachable HTTPS webhook; distinguish test and production endpoints. Telegram commands.md: /start and /help support clear bot interaction; a standard bot cannot initiate a private conversation, so the user must message it first or add it to a group. Bot archive.md: superseded false claim that a bot can initiate a private chat. Website basics.md: HTML gives structure, CSS gives presentation, test mobile layout in browser preview. Return relevant sources, a recommended learning order, reasons supported by filenames, conflicts excluded and missing evidence. Suggested bot path: JSON, webhook, commands. Do not claim an offline model will run on my laptop; no hardware/model evidence was supplied.

[Open copyable prompt](#)

Build your own learning radar

1

Relationships are explained, not just drawn.

2

Two questions retrieve different evidence.

3

An unsupported question is handled honestly.

Build independently, then teach it back

Your next challenge

Collect five approved public notes about an AI project you want to build. Link prerequisites to the goal. Ask two goal questions and test one unsupported claim.

Explain your decisions

Show your artifact, one test that passed, one failure you repaired and one boundary the system respects.