SumizAI .com Your Second Brain, Powered by Intelligence.
https://sumizai.com/blog/how-to-work-with-sumizai.html

GETTING STARTED

How do you actually work with SumizAI day to day?

SumizAI is a small application with a specific shape, and it rewards a specific rhythm. **Nothing here takes more than a few minutes to set up** — the useful part is knowing which decisions are worth making deliberately and which ones the app will make better than you will.

Updated: 2026-08-14 **9** questions 5 min read

Q. What are the first five minutes?

Create an account, connect a provider, create a vault, ask a question, look at the table of contents. That is the entire onboarding, and it is worth doing in that order.

Connecting a provider means pasting an API key in settings — your own key, from whichever of the seven supported providers you already use. There is a *test connection* button that makes a real call, so you find out immediately whether the key works rather than at the moment you needed it to.

Then ask something you actually want to know. The answer streams back, and shortly after it finishes you will see that a note was saved. Open the table of contents and it is already on a shelf.

Q. How should I decide what becomes a vault?

Ask one question: **when I am asking about this, what else should the model be allowed to see?** Everything that answers "yes" belongs in the same vault; everything that answers "no" belongs in a different one.

A vault is a hard boundary, not a folder colour. When you ask a question inside a vault, the model gets that vault's table of contents and that vault's notes. Nothing from any other vault reaches it — which is exactly what you want when your client work and your side project use the same words to mean different things.

In practice most people end up with a small number of broad vaults rather than many narrow ones. A vault with forty notes gives the model something to work with; a vault with three is just a folder.

Q. Should I change the duplicate settings, and when?

Start with the default, which asks you. For the first few weeks the questions are useful: they teach you what the app considers the same topic, and they let you correct it while your library is small enough to correct.

Each vault has two settings: the **mode** — ask, always merge, or always create — and the **similarity threshold**. The threshold matters in every mode, because it decides what counts as "the same topic" in the first place.

Set a vault to *always merge* when it is a long-running study of one thing and you want depth: one growing note per concept. Set it to *always create* when the vault is a log — meeting-by-meeting, day-by-day — where two entries about the same subject are genuinely two entries.

Q. How do I ask questions so the notes come out well?

Ask one thing at a time. A note is generated per exchange, so a question that quietly contains four questions produces one note trying to be about four things — and that note will be hard to file, hard to merge and hard to find later.

Give the model the frame you want back. "Explain X" and "Compare X and Y on cost, latency and operational risk" produce very different expansions, and the second one produces a note you can reread in a year.

Say when you are deciding something. Answers that record a trade-off — what you chose, what you rejected, why — are the ones that pay for the whole system later, because they are exactly what a chat log loses.

Q. What should I do with the table of contents?

Mostly leave it alone, and occasionally rebuild it. The filing decision at note-creation time is made with one note in view, which means it is a good decision that cannot see the future.

There is a **rebuild** action at the vault level that reads all the notes at once and lays the chapters out again from scratch, in batches. That is the one to reach for after a burst of work, when the shape of what you know has changed and the shelves have not caught up.

You can also edit the tree by hand — rename, reorder, add, nest or delete chapters. Deleting a chapter never orphans its notes; they move up to the parent.

Q. When is it worth renaming a note?

Whenever the title has stopped describing the note — which happens naturally after a couple of merges.

The rename is not cosmetic. The filename is the slug of the title, so renaming **moves the file** and then rewrites every link to it across the rest of the vault. The order is deliberate: write the new file, fix the links, delete the old one, and roll the whole thing back if any step fails.

The upside is that your folder stays readable from the outside: the file called `retrieval-cost-model.md` really is the note about the retrieval cost model.

Q. How do I get things back out?

Three ways, for three different reasons.

Search, when you know roughly what you said. It covers the title, the summary, the body and the original question and answer, ignores diacritics, and hands back the matching fragment with the phrase highlighted.

Ask the vault, when you do not know where the answer is. Your question travels with the table of contents and the most relevant notes, and the reply lists which notes it used.

Export, when the audience is not you. A vault downloads as a ZIP of Markdown, as a PDF at summary or full detail, or as a `.pptx` deck in table-of-contents order with a choice of fifteen themes.

Q. What does a realistic week look like?

Monday to Thursday: you work the way you already work, asking a model things and answering questions with it. Notes accumulate without you thinking about them. You say yes or no to a merge prompt a couple of times a day.

Friday, ten minutes: open the vault, skim the table of contents, rebuild it if the shape has drifted, rename two or three notes whose titles no longer fit.

When something is due: search for the thread, export the vault to PDF or a deck, and edit down. The reason that step is short is that the structure was decided forty conversations ago, one note at a time.

Q. What is the most common mistake?

Treating it as an archive rather than a workspace. People connect a key, generate two hundred notes in a fortnight and never ask the vault anything — and then conclude that it is a fancy way to store chat logs.

The value shows up on the way back in. The first time you ask a question and get an answer built out of five things you wrote yourself, with the sources listed, the whole arrangement stops feeling like bookkeeping.

The second most common mistake is one enormous vault holding everything. It is not a disaster, but the context sent with each question gets less pointed, and the duplicate check starts comparing things that were never related.