SumizAI .com Your Second Brain, Powered by Intelligence.

https://sumizai.com/blog/why-sumizai-is-different-from-other-note-editors.html

POSITIONING

Why is SumizAI different from every other note editor?

Most note applications compete on the editing experience: a nicer canvas, a faster outliner, a cleaner sidebar. **SumizAI does not compete there at all.** It assumes the writing already happened — in a conversation with an AI model — and takes over at the moment that is normally lost: the moment you close the tab.

Updated: 2026-08-14 **9** questions 6 min read

Q. Where does SumizAI actually start, if not at a blank page?

At an answer. You have a conversation with a model — Claude, GPT, whichever you connected — and somewhere in it there is a reply you would be annoyed to lose. In a normal editor, that is where your work begins: you copy, you paste, you invent a title, you decide where the note belongs, you promise yourself you will clean it up later.

In SumizAI that is where your work **ends**. The answer becomes a note on its own: a title, a short summary, an expansion, and the full record of the question and the answer that produced it. You did not open an editor. You did not name a file. You had a conversation and a note appeared.

This is the whole difference in one sentence: a note editor is a place to put writing, and SumizAI is a place where writing arrives.

Q. So it is a chat app with a save button?

No, and the difference matters. A save button gives you a transcript — the same wall of text you already had, now stored somewhere else. SumizAI produces a **structured note**, and the structure is the point.

Each note has four distinct parts. The **title** is what the note is about. The **summary** is the version you read when you are scanning. The **expansion** is the version you read when you actually need the detail. And the **source material** is the complete original question and the complete original answer, kept underneath as evidence of where the note came from.

That last part is deliberately append-only. A note can be merged, retitled or relinked, but the original question and answer are never rewritten. Six months later, when the summary reads a little too confidently, you can go and check what was actually said.

Q. What happens to the note after it is written?

It gets filed, without you filing it. Every vault has a table of contents — a real tree of chapters, up to six levels deep, numbered `1.` → `1.1` → `a)` → `a.1)` and so on. When a note is created, the model is shown the numbered chapters that already exist and asked to **pick one**, or to ask for a sub-chapter underneath one of them.

It picks by number, not by inventing a name. That detail is the reason the tree stays sane: if the model were asked to name a chapter from scratch each time, you would end up with *Vegetables*, *growing vegetables* and *Vegetable growing* as three different branches. A new branch only appears when nothing existing fits.

The note then becomes a numbered leaf in that tree, and it is never left outside it. Delete a chapter and its notes move up to the parent rather than falling into a void.

Q. What stops the library from turning into a pile of near-duplicates?

A check that runs before the note is saved. Full-text search and trigram similarity produce a shortlist of notes in the same vault that look like they cover the same ground. Then a **second model reads the candidate and the new material and scores the match between 0 and 1**, with a written reason for the score.

If the score clears the threshold you set for that vault, you get a choice: merge into the existing note, or keep both. A vault can also be configured to always merge or always create a new note, in which case the threshold still decides what counts as the same topic.

When a merge happens, it **adds**. The existing note keeps its original question and answer untouched and gains the new ones alongside. Nothing you captured is quietly overwritten by a later, shorter version of the same idea.

Q. How are notes connected to each other?

Through the text itself. Related notes are referenced as ordinary Markdown links inside the body of the note — `[Chunking strategies](chunking-strategies.md)` — woven into a sentence rather than dumped in a *Related* section at the bottom.

There is a guard on this that is worth knowing about. The model is handed a map of real titles and real filenames and may only link to something in that map. Anything else it produces is stripped out on the way in. A model asked for a link will cheerfully invent a plausible-looking filename, and **a dead link is worse than no link**.

Underneath each note you also get the other direction: the list of notes that link *to* this one. Together they turn a folder of files into something you can actually walk through.

Q. Does the AI see my other notes when I ask a question?

Some of them, deliberately. Your question is sent with the vault's table of contents and the **full text of up to five notes** the app judged most relevant, inside a budget of 24 000 characters.

It is not "the AI reads your entire vault" — that would be slow, expensive and worse, because a model given everything answers from the average of everything. Five relevant notes and a map of the rest is a sharper instrument.

And when the answer arrives, the app tells you which notes went into it. You can check the reasoning against the source instead of trusting a confident paragraph.

Q. Where do the files live?

In a folder you chose, as ordinary `.md` files. The layout is plain: a directory per vault, a `base.md` holding the generated table of contents, and a `notes/` directory with one file per note. Each filename is the slug of its title, so the folder is readable before you open anything.

The database is only an index — it knows titles, summaries and where files are. The file is the source of truth, and there is a reindex operation that rebuilds the index from the files if the two ever disagree. **In a conflict, the file wins.**

The practical consequence: point Obsidian, VS Code or a Git repository at that folder and everything works, because the links between notes are relative paths between real files.

Q. What is SumizAI deliberately not?

It is not a writing tool. There is no canvas, no whiteboard, no database views, no kanban. If you want to draft a document, use something built for drafting documents.

It is not a chat client with better styling. The conversation is the raw material, not the product.

It is not a place your notes are locked into. Every vault downloads as a ZIP with `base.md` and a `notes/` folder, and the links inside are relative, so the archive opens in Obsidian, VS Code or GitHub with nothing installed.

And it is not an AI reseller. You connect **your own API key** to one of seven providers and pay that provider directly. SumizAI charges a dollar a month for the application and takes no cut of what the model costs.

Q. Who is it a bad fit for?

Anyone who wants a shared team workspace. There is no multi-user vault and no synchronisation between devices — your laptop and your phone hold two separate sets of notes. The data model was built with a future sync in mind, but it is honest to say it is not there today.

Anyone who wants the app to supply the AI. It will not: without your own provider key there is nothing to talk to, and the model list is fetched from the provider using your key rather than guessed from a built-in catalogue.

Anyone who does not actually think out loud with a model. SumizAI is very good at capturing conversations you were having anyway, and pointless if you were not having them.