

Reference · Writing these docs

Pages as Markdown, the page menu and print

🕒 7 min read · Updated 11 October 2026

Every wiki page is also published as portable Markdown, has a menu to copy, open or print it, and prints as a clean document: what each does, and what an author must keep in mind.

<https://docs.gurbanisoul.com/contributing/pages-as-markdown/>

Every published page is also served as plain Markdown, so a reader can copy it, a tool can fetch it, and an assistant can be pointed at exactly what the page says:

Address	What it is
/md/<id>.md	one page, for example /md/api/quickstart.md; the home page is /md/index.md
/md-manifest.json	every exported page: title, address, Markdown address, date, reading time
/llms.txt	the site described for language-model tools, one line per page, grouped like the sidebar

Each page also carries `<link rel="alternate" type="text/markdown">` in its head. The 404 page and the generated API reference have no Markdown source and are not exported; the reference's source is `contract/openapi.json`.

What the export changes

The Markdown in `docs/` is written for two renderers, GitHub and this site, so it carries things only this site understands. The export resolves them and changes nothing else:

In the source	In the export
a glossary term in double square brackets	the plain word
a <code>sggs:code</code> comment	the real lines of the file, in a code fence, with the commit they were read at
any other widget comment	removed; the “On the rendered wiki...” paragraph stays, or one line says where the part is live
a relative link to a page	the page's full address on this site
a relative link to any other file	its GitHub address
a poster image	the address of the published SVG
the <code># Title</code> line	removed; a short header gives the title, description, address, dates and the “Verified against” stamp

Mermaid and quiz blocks are left as written.

What it never changes

The export **does not re-write Markdown**. The page is parsed only to find positions; a list of small edits is made against the original text, and every other character passes through untouched. There is no Unicode normalisation and no change to line endings or spacing.

Scripture is held mechanically (`docs-site/plugins/portable-md.mjs`):

- Every run of Gurmukhi in a page is protected, wherever it is: a quotation, a table, a heading, a list. An edit that would touch one fails the build.
- A quotation that contains Gurmukhi is frozen whole. A link inside it stays as written.
- After each page is exported, the build proves that every run is in the export, in order, character for character, and that undoing the edits gives the source back.

docs-site/plugins/portable-md.mjs · verifyExport · lines 60–90

```

export function verifyExport(body, edits, out, where = 'page') {
  let shift = 0, i = 0;
  const inserted = []; // ranges of `out` that are new
  text
  const sorted = edits;
  for (const e of sorted) {
    if (e.text) inserted.push({ start: e.start + shift, end: e.start + shift + e.text.length
});
    shift += e.text.length - (e.end - e.start);
  }
  shift = 0;
  const runs = protectedRuns(body);
  for (const r of runs) {
    while (i < sorted.length && sorted[i].end <= r.start) { shift += sorted[i].text.length -
(sorted[i].end - sorted[i].start); i++; }
    if (i < sorted.length && sorted[i].start < r.end) throw new Error(`${where}: an edit
touches scripture at offset ${r.start} (${sorted[i].kind})`);
    const want = body.slice(r.start, r.end);
    if (out.slice(r.start + shift, r.end + shift) !== want) throw new Error(`${where}: a
Gurmukhi run moved or changed at offset ${r.start}`);
  }
  const outside = (r) => !inserted.some((x) => r.start >= x.start && r.end <= x.end);
  const kept = protectedRuns(out).filter(outside).map((r) => out.slice(r.start, r.end));
  const want = runs.map((r) => body.slice(r.start, r.end));
  if (kept.length !== want.length || kept.some((s, k) => s !== want[k])) {
    throw new Error(`${where}: the export has ${kept.length} Gurmukhi run(s) outside
inserted text, the source has ${want.length}, or they differ`);
  }
  let undone = '', at = 0, delta = 0;
  for (const e of sorted) {
    undone += out.slice(at, e.start + delta) + body.slice(e.start, e.end);
    at = e.start + delta + e.text.length;
    delta += e.text.length - (e.end - e.start);
  }
  undone += out.slice(at);
  if (undone !== body) throw new Error(`${where}: undoing the edits does not give the source
back`);
}

```

Read from the source at build time — [view these lines on GitHub at c3d9360](#) .

Source: docs-site/plugins/portable-md.mjs, read at build time.

What an author must keep in mind

The export fails the build, with the page and the reason, on anything it cannot edit safely:

- **Write links inline**, [text](target.md). A reference-style link to a relative target is refused.
- **Images are posters**. Any other relative image is refused until a rule exists for it.
- **A sggs: code comment starts its own line**, at the left margin.

- **Give an interactive widget its fallback paragraph**, starting “On the rendered wiki”. It is what a reader of the Markdown, of GitHub, and of a printed page gets in its place.
- **Do not put a glossary term whose name is Gurmukhi behind a different shown word.** The export would have to delete the Gurmukhi, and it never does.

Where it is tested

- `docs-site/plugins/test/portable-md.test.mjs` exports **every** published page and checks each Gurmukhi run byte for byte; its fixtures build Gurmukhi from code points, so no line of scripture is typed into a test. It fails, not skips, when a pinned repository’s docs are not installed.
- `docs-site/e2e/markdown.spec.ts` fetches every page’s Markdown from the built site and compares a quotation in the export with the same quotation on the page.
- After each deploy, `scripts/ci/docs_smoke.py` fetches `/md/index.md` and checks its type and character set, on builds that announce the md1 feature.
- `docs-site/plugins/test/pdf.test.mjs` breaks each promise the PDF gate makes and expects it reported; `docs-site/e2e/pdf.spec.ts` fetches a PDF from the built site.
- `docs-site/e2e/page-actions.spec.ts` drives the menu on desktop and phone: copying, the keyboard, the refusals, no JavaScript, and axe with the panel open in both schemes. `docs-site/e2e/print.spec.ts` checks the printed page as a dark-theme reader would get it.

The files are served as `text/plain; charset=utf-8` so they open in a browser tab, and with `X-Robots-Tag: noindex` so search engines index the page, not its copy. Both come from a rule in `docs-site/vercel.json`.

The page menu

Above every title is **Copy page**, which copies the page’s Markdown, and a panel beside it:

Item	What it does
Copy page as Markdown, Copy link	to the clipboard; a line says “Copied”, or what to do when the browser refuses
View as Markdown	opens <code>/md/<id>.md</code> in a new tab
Ask ChatGPT, Ask Claude	plain links to those services with an instruction and the page’s public Markdown address (ADR-0013)
Download PDF	the page’s pre-built PDF, <code>/pdf/<id>.pdf</code>
Print	the browser’s print dialog; shown where there is a mouse or trackpad

Every link in it works without JavaScript; only copying and printing need script. The menu is not on the 404 page, and on the generated API reference it copies the link, since there is no Markdown to copy. It records nothing. The line under the title also says how

long the page takes to read (prose, tables and code at 220 words a minute), except on a section's overview page and on pages under 150 words.

On paper

Printing a page, from the menu or the browser, gives the article and nothing else (`docs-site/src/styles/print.css`):

- always the light palette, in Source Serif, with Gurmukhi in Sant Lipi;
- the page's address under the title and a colophon at the end (commit, dates, “this copy may be out of date”);
- a running head and “Page x of y” where the browser supports them;
- every external address listed once under **Links**, with a small number in the text (`docs-site/src/widgets/print.ts`);
- a quotation and its Ang kept on one sheet; a poster scaled to fit one sheet, with its steps printed as a list; table headers repeated; code wrapped;
- a live widget replaced by one line saying what it is.

For authors: a poster's steps come from its `.steps.json`, so write captions that read well as a list. Nothing else is needed: a page that passes the gates prints.

The pre-built PDFs

Every exported page also has a PDF at `/pdf/<id>.pdf`, made from the built page with the same print stylesheet, so it reads exactly like a careful print. `docs-site/scripts/build-pdfs.mjs` makes them **inside the build** (the deploy jobs build the site themselves, and only what a build writes is deployed), when `DOCS_PDF=1`; CI sets it for every build it tests or deploys, and a local build is quicker without (`npm run pdf` makes them afterwards). Each page is rendered at its published address with every request answered from the built files, so links inside a PDF point at the real site and the build never calls the network. A page that cannot be rendered fails the build.

`npm run check:pdf` (part of `check:all`, and run again by each deploy job on what it built) holds them to this:

- one PDF for every exported page, titled, tagged for assistive technology, with bookmarks;
- **Gurmukhi is set in Sant Lipi:** a missing or substituted font fails the build. The one allowance is a vowel sign with no letter before it (shown alone, or put first to show the PDF's visual order): a browser cannot use a range-limited web font for it and may draw the sign, or the word it sits in, in a fallback. That is allowed only on a page whose own text has such a sign, and for no more runs than it has; seven pages do, and the rest must be Sant Lipi throughout;
- an Ang citation is on the same sheet as its quotation;
- no link leads to the machine that built it; each file is at most 3 MB and all of them 75 MB.

A PDF cannot be proven character for character the way the Markdown is: shaping reorders glyphs inside the file. The verbatim export is the Markdown; the PDF is checked for font, presence and place.

Links

1. <https://github.com/Algorythmos-AI/siggs-platform/blob/c3d936031f0c283283c5d1102705affe1f8cf404/docs-site/plugins/portable-md.mjs#L60-L90>

Sri Guru Granth Sahib Ji — Knowledge Base wiki. Generated from commit c3d9360. Updated 11 October 2026. This copy may be out of date; the page is at <https://docs.gurbanisoul.com/contributing/pages-as-markdown/>