

A browser-side-dynamic website framework for annotated Indic texts

Vishvas Vasuki

Dyugaṅgā, Beṅgaḷūru

<https://sanskrit.github.io/groups/dyuganga/>

Abstract

Perusing Indic classics on the web is an increasingly popular activity. However, serving and maintaining feature-rich websites involves significant costs, knowledge and effort. We present an easy to use website framework to solve mitigate these drawbacks. Besides being simple and low-cost, the proposed solution is rich in features particularly useful for presenting Indic language classics - including easy transliteration, navigation aids, annotation support, dynamic content inclusion and more.

1 Motivation

The advent of internet, personal computers and mobile devices have radically changed the way we consume literature. We can now carry an entire library in our pockets. Besides access to simple text, digital readers are able to utilize features such as text-to-speech, embedded multimedia, easy navigation and dictionary lookup.

On the production end - publishing on the web is far simpler and cheaper than publishing paper books. Yet, doing it well (i.e. with the dynamic features digital readers expect) still involves significant costs, knowledge and effort. Furthermore, for highly structured texts - like originals with series of commentaries and translations - content management becomes non-trivial.

We present a framework to publish annotated Indic language texts, providing the following features:

- **Content management**

- “Suggest edits” links, which work when the content is stored in an online Github-like repository.
- Basic ability to include contents from another page using the same theme within another.
- Inline annotation
- It’s easy to copy or export the underlying content.
- Version control is available when backed by a GitHub-type online repository.
- Very low operating costs ¹ - only a simple web page server is needed - no database server or ability to run custom code on server is required.

- **Navigation**

- Collapsible “accordion” sidebar with automated directory listing.
- Collapsible “accordion” table-of-contents for each page.
- “Next and Previous” page navigation buttons.

- **Search**

- JSON based title/ url search
- Search engine optimization markup and indices - which you would use with various search engines.

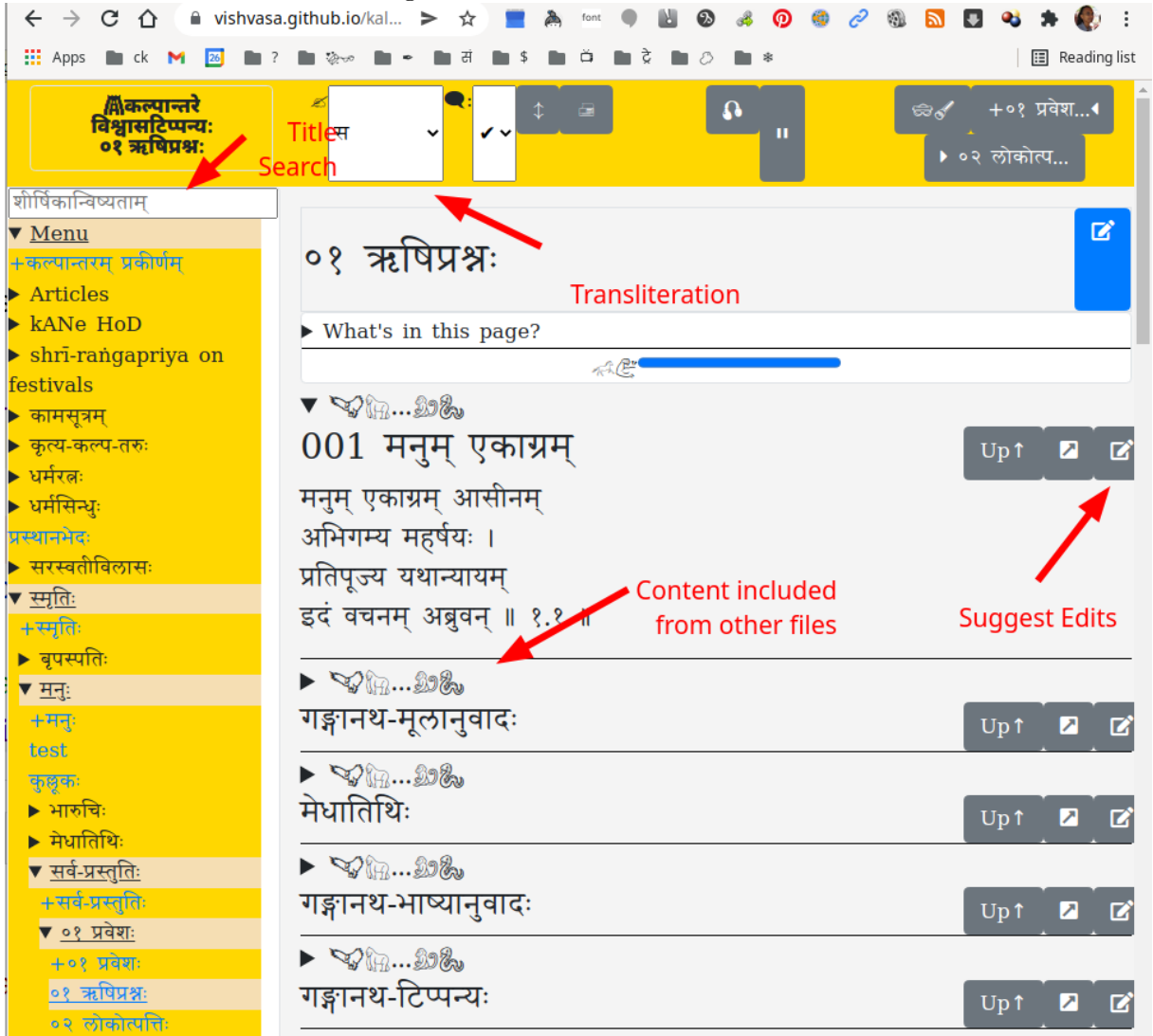
¹Zero cost if services like GitHub Pages are used.

- **Reading**

- A layout which automatically adjusts to the user's screen size.
- Embedding audio and video items. Ability to sequentially play audio tracks within a page.
- Transliteration dropdown: scripts ranging from devanāgarī to grantha, brāhmī to ISO-15919.
- Special formatting consideration for fonts which need to be displayed bigger (eg: Devanagari for sanskrit.)
- Text to speech interface (under development).
- Disqus for comments.
- Collapsible detail sections.

This framework has been used in several websites, such as (Vishvas Vasuki, 2022c), (Vishvas Vasuki, 2022e) and (Vishvas Vasuki, 2022a).

Figure 1: Screenshot of a website



2 Content format

Content is stored in the form of plain text files with Markdown content (John MacFarlane et al., 2021) and YAML or TOML metadata headers (Tom Preston-Werner, Pradyun Gedam, et al., 2022).

2.1 Markdown

The advantage of Markdown is that it is very simple and intuitive - while having most basic features (such as sectioning, bold text, italics, lists, footnotes, images). One can edit markdown with just a plain text editor, without the need of special editing software as in the case of TEI².

Markdown format is very popular, and is supported by several parsers in popular programming languages, with HTML being most common target language. This makes the content highly portable accross display software (including static website generators).

An example file is as follows:

```
+++
english_title = "001 nārada briefs vālmīki about rāma"
title = "००१ सङ्क्षेपरामायणम्"
unicode_script = "devanagari"
```

```
+++
## नारदाय प्रश्नः
```

```
तपस्स्वाध्यायनिरतं तपस्वी वाग्विदां वरम् ।
नारदं परिप्रच्छ वाल्मीकिर्मुनिपुङ्गवम् ॥1.1.1॥
```

```
कोन्वस्मिन्साम्प्रतं लोके गुणवान्कश्च वीर्यवान् ।
धर्मज्ञश्च कृतज्ञश्च सत्यवाक्यो दृढव्रतः ॥1.1.2॥
```

```
चारित्र्येण च को युक्तस्सर्वभूतेषु को हितः ।
विद्वान्कः कस्समर्थश्च कश्चैकप्रियदर्शनः ॥1.1.3॥
```

...

2.2 Extensions to Markdown

Basic markdown can be enhanced with HTML code as needed - but we recommend being conservative about it as adding too much HTML almost never necessary, and HTML code can begin to obscure content and reduce plain-text readability. We have found the details tag to be particularly useful to produce collapsible boxes; for example:

```
<details><summary>मूलम्</summary>
```

```
क सूर्य-प्रभवो वंशः
क चाल्पविषया मतिः ।
तितीर्षुर् दुस्तरम् मोहाद्
उडुपेनास्मि सागरम् ॥ २ ॥
</details>
```

Occasionally, we embedding find page-specific scripts useful as well, using the regular <script> tag (for example to open a random verse or stotra upon visiting a certain URL).

2.2.1 Content inclusion

In addition to this, our framework fetches and fills in markdown/ text/ HTML content from other URL-s with tags such as the following:

```
<div class="js_include" newlevelforh1="4" title="Oldenberg" url="/khAdira/1/02.md"></div>
```

We've found such content inclusion to be very useful in managing structured data.

²TEI is a popular XML-based format for storing structured Sanskrit texts. With XML, as in the case of HTML, the tags begin to eclipse the content - making plain text editing cumbersome.

- For example, a particular verse may be associated with 10 commentaries - storing them separately - say in numbered files - replicates the advantages of using a database with the simplicity and convenience of a plain file system. One can edit some commentary without being bothered by the clutter of unrelated content.
- Furthermore, the exact same content (example the rk "tat savitur ..." with annotation) may be included in multiple texts (example - rgvēda, yajurvēda, upākarma ritual procedure ...). We're able to avoid having to copy / paste and maintain identical content in multiple locations by efficiently using include directives.

2.2.2 Embedding data and media

We can embed tables from - say TSV or TOML files - as spreadsheets using the directive shown below:

```
<div class="spreadsheet" src="../../sutras.toml" fullHeightWithRowsPerScreen=8> </div>
```

Similarly, for embedding multimedia:

```
<div class="audioEmbed" src="http://www.xyz.com/raghuvaMsha1.mp3"
```

```
caption="Recitation by Kuṭumbaśāstrī"></div>
```

```
<div class="videoEmbed" src="https://www.youtube.com/watch?v=TDNkBFbp9Pk"
```

```
caption="Abhinaya by Padmā"></div>
```

For figures with caption:

```
{{< figure src="../../images/khanjar.jpg" title="The khanjar dagger" class="thumbnail">}}
```

2.2.3 Inline comments

A favorite reading practice is to insert one's comments in the middle of some text. With our framework, this can be done as shown below:

सेना परिच्छदस् तस्य,

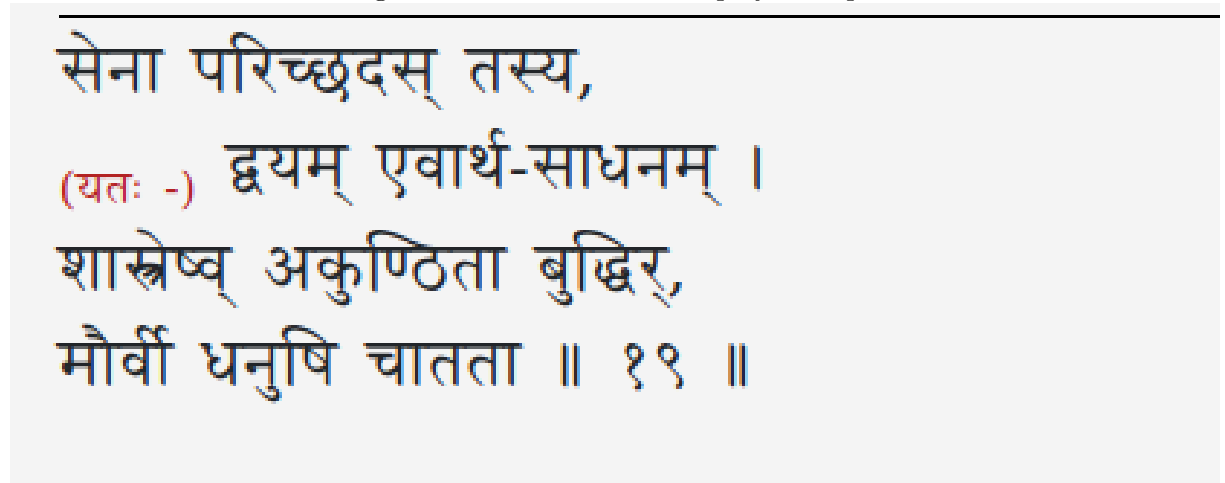
+++ (यतः -) +++ द्वयम् एवार्थ-साधनम् ।

शास्त्रेष्व् अकुण्ठिता बुद्धिर्,

मौर्वी धनुषि चातता ॥ १९ ॥

This can be made to yield:

Figure 2: Inline comment display example



2.3 File metadata

Metadata is stored in TOML or YAML format at the top of the file, as shown in an example earlier. The most important metadata for a given file is its title. That apart, specifying the script used in the content is important for transliteration and display.

2.4 Mechanical content management

Content files are often hand crafted. Others are the generated mechanically from pre-existing content. For example:

- It may be desirable to split a large content file with multiple sections into multiple different files for ease of navigation and speed. Or one might want to join multiple files into a single file.
- One may want to generate markdown files from pages scraped off some preexisting website.
- One may want to pre-include content within the include directives (mentioned earlier) so as to facilitate proper indexing by search engine crawlers.
- One may want to transliterate content, or add a few commentaries for each verse.

For such cases, we have developed an open source python library (Vishvas Vasuki, 2022b), which inturn uses a python transliteration library as needed (Sanskrit Programmers, 2022a).

3 Generating HTML pages

The content is ultimately presented to the user via a web browser. Hence, HTML pages are generated from content markdown files efficiently. This process can be automated with simple "Continuous Integration" scripts on hosts such as GitHub (Vishvas Vasuki, 2022d). For the most part, we use Hugo - a popular static site generation framework (Hugo Authors, 2022) together with UI templates we've designed as part of our Hugo "theme" (Vishvas Vasuki, 2022f).³

For faster builds, enable some markdown files to be served without parsing them into HTML⁴. They are parsed into HTML as needed on the reader's browser.

4 Browser side dynamism

For the convenience of readers, it is essential that the website be interactive. Rather than rely on running custom code on the HTTP server (and incur significant costs and complexity), we rely on the reader's browser. Whereas some of this browser-side dynamism can be accomplished simply via HTML directives (as in the case of details tag or CSS styles), some interactivensess is accomplished via Javascript code (as in the case of navigation, search, content inclusion and transliteration). This code has been incorporated into our Hugo theme (Vishvas Vasuki, 2022f), and is embedded into each webpage generated by the static site generator. An advantage on relying on browser side scripting is that the code is rendered independent of any particular static website generation software (which oft use custom template languages).

One disadvantage of relying on Javascript to include or modify content is that it may thwart proper indexing of the generated content by search engine web crawlers. This is potentially caused by rendering delays⁵. This problem can be mitigated by mechanically "pre-including" a snapshot of the content to be included at the time the HTML pages are built. This content can then be updated dynamically by browser side scripts.

An important feature supported by browser-side scripting is transliteration of Indic content. This is accomplished using an open source Javascript transliteration library (Sanskrit Programmers, 2022b).

5 Future work

We want to enhance the capabilities of the framework so as to support text-to-speech, aligning text with the audio as it is played (Avinash Varna, 2022), quiz questions and so on. Some search engine optimization problems remain as well - such as transliterated content not being crawled

³We initially used (Vishvas Vasuki, 2020) Jekyll (Jekyll Authors, 2022), which is orders of magnitude slower.

⁴This is accomplished by keeping them within the static folder in case of Hugo.

⁵Experience suggests that the Google web crawler, for example, does not wait longer than 5s for the page to be rendered - if it waits at all.

and unaccented content not being submitted for indexing. Exporting the generated web content into ebooks will also be of great interest.

References

- Avinash Varna. 2022. Text-aligned sanskrit recitations on a friendly website. https://avinashvarna.github.io/audio_alignment/.
- Hugo Authors. 2022. Hugo. <https://gohugo.io/>.
- Jekyll Authors. 2022. Jekyll. <https://jekyllrb.com/>.
- John MacFarlane et al. 2021. Commonmark markdown specification. <https://commonmark.org/>.
- Sanskrit Programmers. 2022a. Indic transliteration python pip package. https://github.com/indic-transliteration/indic_transliteration_py.
- Sanskrit Programmers. 2022b. Sanscript package. <https://github.com/indic-transliteration/sanscript.js>.
- Tom Preston-Werner, Pradyun Gedam, et al. 2022. Toml. <https://toml.io/en/v1.0.0>.
- Vishvas Vasuki. 2020. Sanskrit documentation theme for jekyll. <https://github.com/sanskrit-coders/sanskrit-documentation-theme>.
- Vishvas Vasuki. 2022a. Bhāṣāntarēṣu viśvāsa-ṭippanyaḥ. <https://vishvasa.github.io/bhAShAntaram/>.
- Vishvas Vasuki. 2022b. Doc curation python pip package. https://github.com/sanskrit-coders/doc_curation.
- Vishvas Vasuki. 2022c. Kalpāntarē viśvāsa-ṭippanyaḥ. <https://vishvasa.github.io/kalpAntaram/>.
- Vishvas Vasuki. 2022d. Kalpāntarē viśvāsa-ṭippanyaḥ - sources. <https://github.com/vishvAsa/kalpAntaram/>.
- Vishvas Vasuki. 2022e. Kannaḍadalli viśvāsa-ṭippanigaḷu. <https://vishvasa.github.io/kannada/>.
- Vishvas Vasuki. 2022f. Sanskrit documentation theme for hugo. <https://github.com/sanskrit-coders/sanskrit-documentation-theme-hugo>.