

CS 297 Report

Context-Sensitive Wiki Help System for Yioop

Advisor: Dr. Chris Pollett

By

Eswara Rajesh Pinapala,
Department of Computer Science,
San Jose State University.

Introduction

A Context-sensitive Online Help system provides targeted information to users based on their context (i.e., where they are, what their current state is, or what job are they performing) with respect to the application. Context-sensitive help can take several forms, usually, as a widget, overlay page, or as a hyperlink to a completely different page or window. The user need not move out of context to get help for his current task. Each help topic intends to be applied for a given context exclusively.

A Wiki-based Help System allows users to collaborate on help content. Most wiki-based support systems are setup as portals, centralized sites where users can get help and contribute to help content. One problem with a wiki-based help system is that the user has to move out of context and search for help from a large pool of help articles.

A Context-Sensitive Wiki Help System combines the aspects of context-sensitive help with the collaborative nature of a wiki. Users get the targeted help based on their context, but they also have ability to edit and contribute to the help content.

The beginning of this writing project will be a study of help content authoring tools with support for context-sensitive help systems [1] [2] and wiki systems. Using the information acquired from the study, I will build a feature set that will be used in the development of the context-sensitive wiki help system for web applications. Also, as part of one of the deliverables, I will also be building a basic wiki editor. I will be using Yioop; an open-source search engine application developed by Dr. Chris Pollett, as a platform to integrate with our context-sensitive wiki based help system.

2. Deliverables

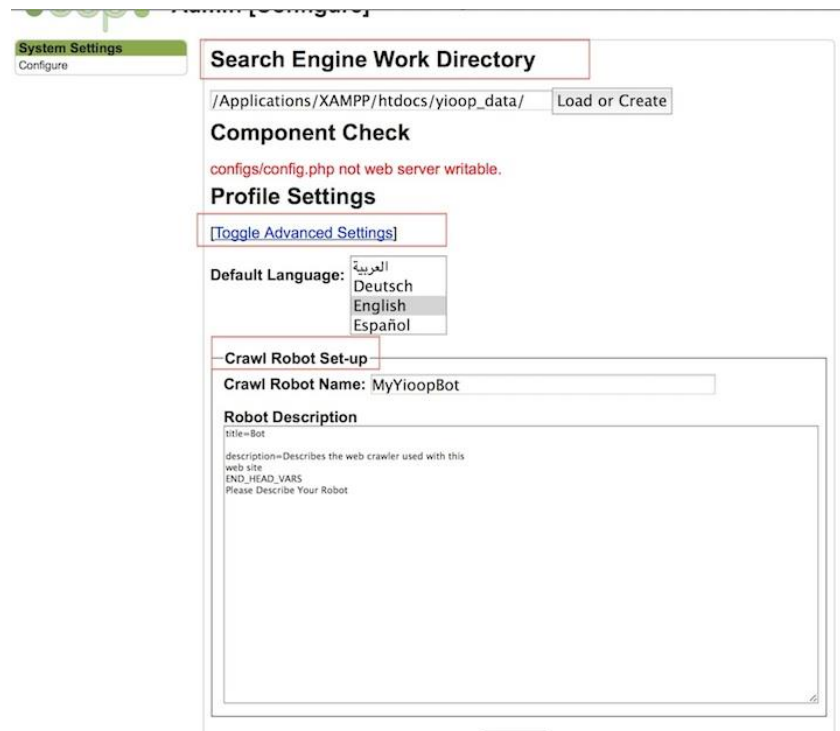
2.1 Deliverable 1: Research on Context-sensitive help

2.1.1 Research on Context-sensitive help authoring tools.

The purpose of this deliverable is to get a clear understanding of how context-sensitive help systems are usually configured and used. Two context-sensitive help content authoring tools were researched to understand the context-sensitive help setup and configuration [1] [2]. The two tools configured were Flare by Madcap Software and Robohelp by Adobe. This research was helpful to understand how to enable help content authors to publish context-sensitive help. In addition to content authoring and publication, it was possible to identify the procedure to guide web developers to easily integrate context-sensitive help into existing web pages.

Studying existing research work on context-sensitive help was another task accomplished as part of this deliverable. Research paper studies were on “Context-Sensitive help for Multimodal Dialogue [1]” published by AT&T researchers and the presentation were on “User-centered Design of Context-sensitive Help [2]”.

Flare by Madcap and Adobe Robohelp are comparable in features and installation. Flare and Robohelp allow the help content authors to easily generate context-sensitive help and integrate the same into web applications. This deliverable throws some light on the findings from the setup and configuration of Madcap Flare and Adobe Robohelp.



The screenshot displays the Yioop configuration interface. On the left, a sidebar contains a 'System Settings' menu with a 'Configure' link. The main content area is titled 'Search Engine Work Directory' and includes a text field with the path '/Applications/XAMPP/htdocs/yioop_data/' and a 'Load or Create' button. Below this is a 'Component Check' section with a red error message: 'configs/config.php not web server writable.' The 'Profile Settings' section features a 'Toggle Advanced Settings' link and a 'Default Language' dropdown menu currently set to 'العربية', with other options being 'Deutsch', 'English', and 'Español'. The 'Crawl Robot Set-up' section contains a 'Crawl Robot Name' field with 'MyYioopBot' entered. The 'Robot Description' section has a text area with pre-filled text: 'title=Bot', 'description=Describes the web crawler used with this web site', 'END_HEAD_VARS', and 'Please Describe Your Robot'.

Figure 1 shows parts of the Yioop configuration page to insert context-sensitive help content

The first step involved in generating help content is to finalize places on the web page where help needs to be inserted. We intend that the Yioop configuration page will have the context-sensitive help inserted at a few spots identified as shown in figure 1.

Madcap Flare and Robohelp provide a way to organize the help content into individual help content topics. A group of help topics can be identified and placed into a help chapter.

The identified help topics are as below:

- Install & configure
- Profile settings
- Advanced settings
- Crawl-robot setup

We have used the help content from the official Yioop help docs to gather appropriate help content for the help topics in the above list.

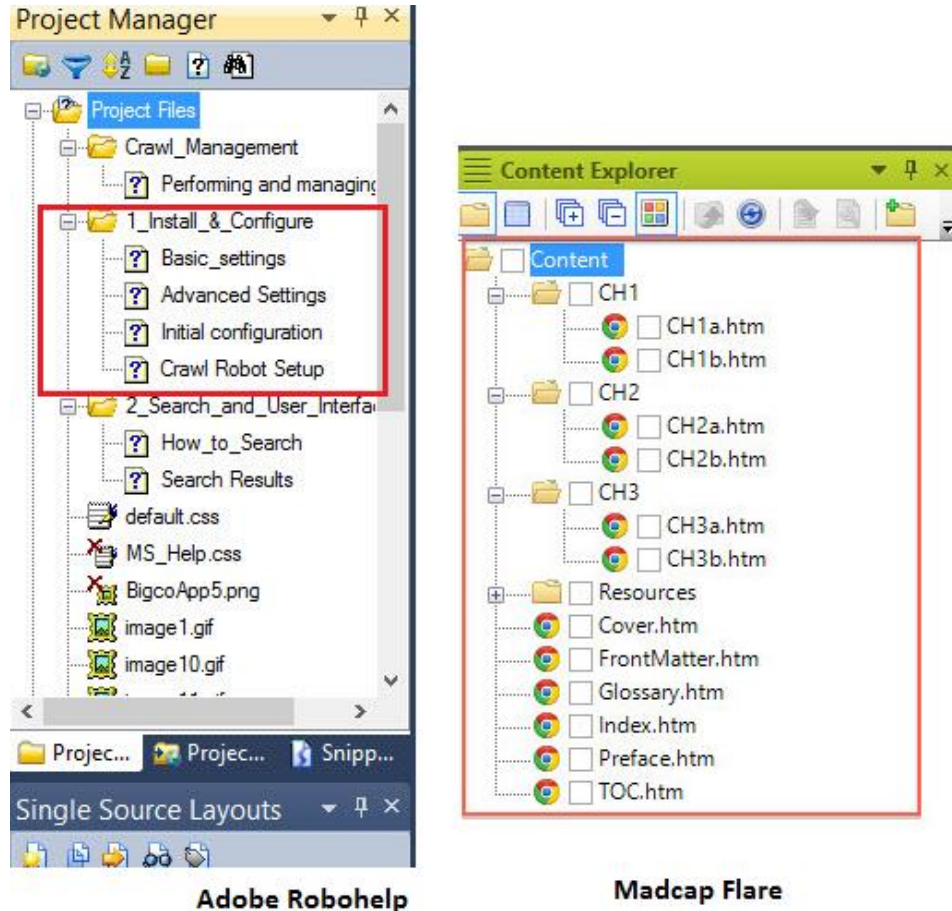


Figure 2 shows the chapter view in Madcap Flare and Adobe Robohelp.

Header files and Alias files

“Header” files and “Alias” files are plain text files, used to define which context sensitive help content will be mapped to which parts of the webpage. Header files help the content authors to easily manage the relationship between the web pages and help content. Below is an example entry in a header file for Madcap Flare and Adobe Robohelp.

```
# define {TopicName}{TopicNumericId}
```

Example: # define CHA1 1000

Alias files, on the other hand, include a mapping between the support chapters and their respective hyperlink. Alias files in combination with Header files serve as an index for the entire help content. The names, which identify the chapters, are assigned a help content page in HTML format. Alias files are standard XML files containing the mapping information in each element.

Example mapping element:

Madcap Flare

```
<Map Name="CH1A" Link="/path/to/CH1a.htm" />
```

Adobe Robohelp

```
<alias name="Crawl_Robot_Setup" link="/path/to/Crawl_Robot_Setup.htm ">
</alias>
```

Both Alias files and header files together, provide a way to define a mapping to uniquely identify the chapters using their names and identifiers as parameters. The HTML or JavaScript code embedded into the web pages can use the topic identifiers information from the alias and header files. The HTML buttons enable the retrieval of help content page, by using the numeric identifiers as parameters for the JavaScript API function. They also provide the flexibility for the content authors to change the mapping or crosslink the help articles without making changes to the web pages. If the help content author wants to change the content of a help topic or wants to use a new help article altogether, concerned changes could be performed just in the alias file.

Once the help content, along with header files and alias files is created, Madcap Flare and Adobe Robohelp compile and generate all the help content into a single directory. The compiled help content also includes a tiny JavaScript framework that can be embedded right into any web application. The purpose of this JavaScript framework is to provide an easy API to web developers integrate the help content into their web application. Without the JavaScript framework, the developers have to use the header file to build code to integrate context-sensitive help pop-ups into their web application.

The help content folder can be moved to a subdirectory on the Yioop web server root directory. The generated help content pages are now visible at <http://yioop.com/csh/>. The web page displays context-sensitive help in the form of pop-ups. HTML buttons are used to trigger the help pop-ups.

As described earlier, the unique numeric id (CSHID) is used to trigger the concerned context-sensitive help pop-up, with the support of the JavaScript API provided by Flare. A sample HTML button code looks like below.

```
<input type="button" value="?" onclick="FMCOpenHelp ('CSHID ', null, null, null);" />
```

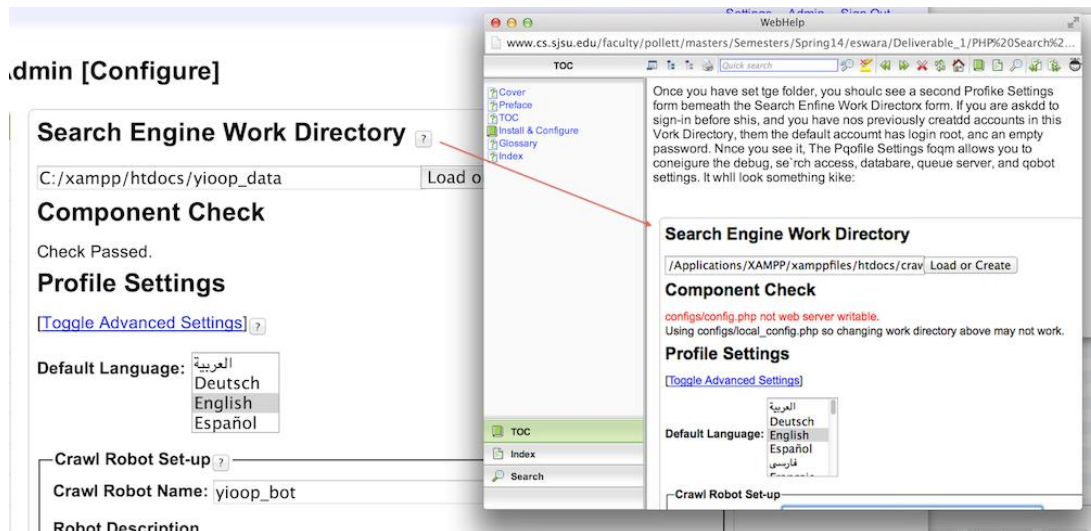


Figure 2 shows Yioop configuration page integrated with context-sensitive help.

2.1.2 Research on Context-sensitive help.

The second part of Deliverable 1 will explain my research on existing context sensitive help systems. The main intention behind studying the existing systems is also to understand the problems and advantages of Context-sensitive help from the past. The first part of the deliverable research was on how context sensitive help systems improved over time [2]. During my research, I discovered that there are mainly two kinds of context sensitive help systems in use. One is reference bases and the other is procedural based. The third type is the contextual help.

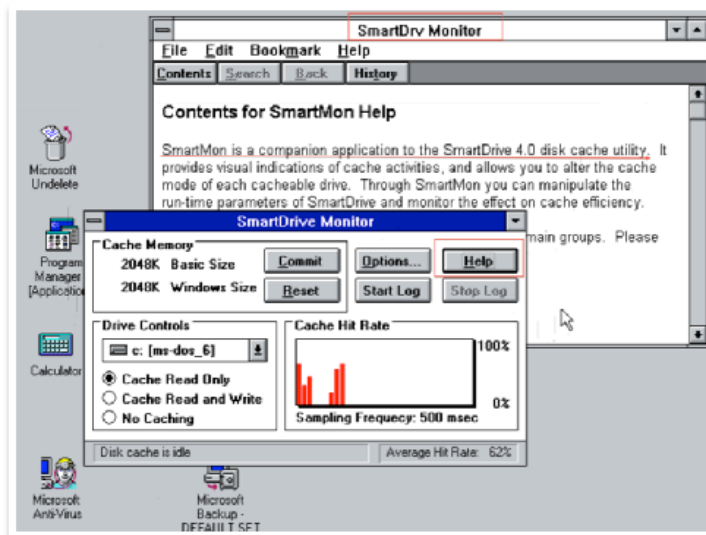


Figure 3 shows reference based context-sensitive help in Windows 3.1

Reference based context-sensitive help is a method of dumping all the help content as reference chapters at a single place. Though the reference documentation is provided specific to a page or a window, a problem arises when the user is searching for something specific. The user is

required to search in a pool of help contents. Thus, reference based context-sensitive help requires extra time and effort from the user, so this is not always a good idea.

Procedural help [3], as its name suggests, is a procedure oriented context-sensitive help. procedural help article usually takes user's current location and tries to provide help content on how to perform a task. In addition to this, procedural help is also useful to let the user know what kinds of tasks are possible at a specific area.

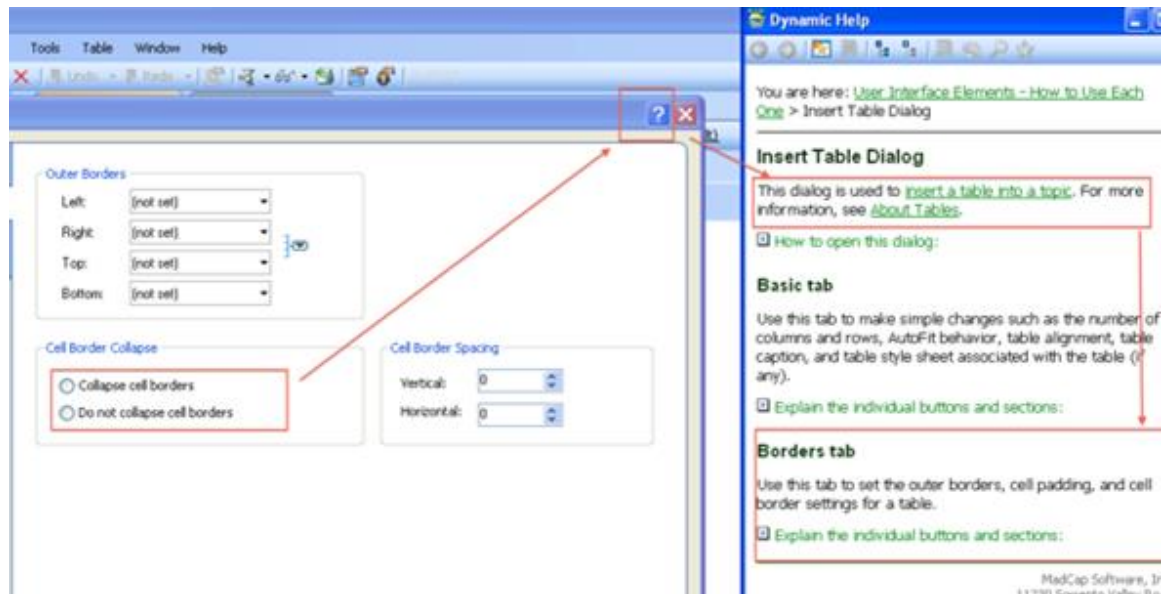


Figure 4 shows procedural context sensitive help in Flare [3]

Contextual help [5], on the other hand, is the additional information on the UI that supplements the UI and provides a brief, up to the point support. Contextual help is usually displayed in a pop-up window or beside some relevant fields. Contextual-help embedded in the application interface to makes sure user does not leave the interface.

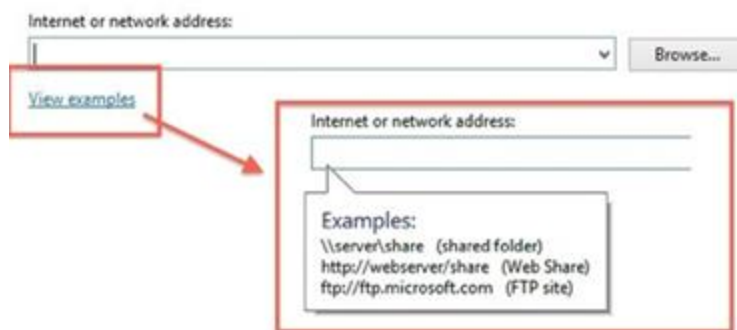


Figure 5 shows contextual help in Windows 8

Currently, all the help content in Yioop is a reference based context-sensitive help system. The idea is to keep the current reference based help content intact and add context sensitive help system which is more focused on contextual help and procedural help.

2.2 DELIVERABLE 2: RESEARCH ON EXISTING WIKI SYSTEMS

2.2.1 MediaWiki

This section of the deliverable will describe the configuration and features of the MediaWiki system. MediaWiki is an open source Wiki engine written in PHP. Wikipedia is one of the popular real life applications of MediaWiki [4]. Creating Wiki Pages in MediaWiki is the same way as accessing the Wiki Pages. Every wiki page in MediaWiki has a title that serves as the page's unique identifier. The users can search for the page or access the wiki page URL directly. If the page exists, MediaWiki renders an editor page for the users to edit the wiki page. If the page does not exist, MediaWiki prompts the user to create a new page. As the page with the title - "New Page" does not exist; MediaWiki will prompt the user to create this new page. Clicking on the Create the page "New page" will take the user to the Wiki editor to add contents to a new page.

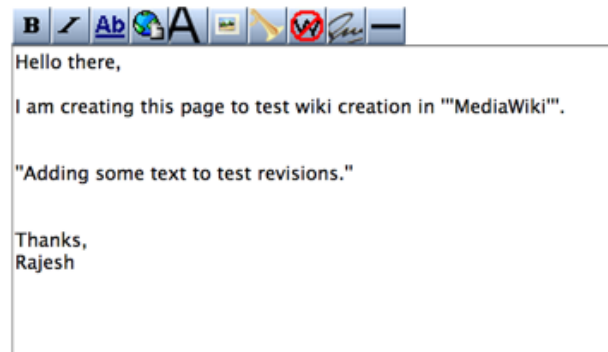


Figure 6 shows the wiki editor to create or edit wiki pages.

MediaWiki organizes content using namespaces. MediaWiki ties all like content or like pages into a namespace. Thus, namespaces tie pages or content by their topic. A good example of a name space is “Category” A page can be grouped into an existing category by providing the namespace as part of the wiki source.

Example: [[Category: Games]]

“Category” is the namespace and “Games” are the name of the Category.

The MediaWiki editor supports some basic content styling features like bold text, italic text, underlined text, image insertions, hyperlinks etc., with its basic WYSIWYG editor. In addition to the edited text, users will be able to add a summary to the page or the editing session. The wiki pages can be edited and revised as many times as possible. The edit button can be used to pull up the editor to edit the concerned page.

MediaWiki Revisions

This part of research will be helpful for building the revision controls system in the context-sensitive wiki help system for Yioop. The “page_table” in MediaWiki stores the higher level details of the wiki page, like the title, page, page_counter, etc.

| page_id | page_namespace | page_title | page_restrictions | page_counter |
|---------|----------------|-------------|-------------------|--------------|
| 1 | 0 | Main_Page | 0B | 20 |
| 2 | 6 | Example.jpg | 0B | 1 |

Figure 7 shows a sample page record.

A table called revision stores all the revisions of all the wiki pages. Below is a figure that shows the schema and different fields in the revision table. Every wiki pages edit inserts a record into the revision table, which consists of information such as the user who created the edit, timestamp of the edit and a reference to the "old_text" column in the text table. This record in the revision table holds information about two things mainly, edit operation and a new wiki text originated from the result of the edit operation. A new record in the revision table always points to the current version of the wiki text. MediaWiki saves the entire wiki content, after the edit, into a data blob field in the database. Again this is similar to other wiki engines where the entire contents of the edited page are stored instead of diff metadata.

| old_id | old_text | old_flags |
|--------|---|-----------|
| 1 | '''MediaWiki has been successfully installed.'''Consult t... 524B | utf-8 5B |
| 2 | '''MediaWiki has been successfully installed.'''Consult t... 529B | utf-8 5B |
| 3 | '''MediaWiki has been successfully installed.'''Consult t... 538B | utf-8 5B |

Figure 8 shows wiki pages and revision in the database.

2.2.2 TikiWiki

This deliverable will describe the configuration and features of the TikiWiki system. TikiWiki is not just a Wiki system; its real name is "TikiWiki CMS/Groupware." Tiki is a CMS, Wiki system, Blog engine and a webmail system. TikiWiki is written in PHP and requires a web-server that can process PHP server side scripts and a Relational Database like MySQL.

It has a huge code base compared to other CMS/Wiki engines. One of the most popular live sites leveraging TikiWiki is the Mozilla support Site at <http://support.mozilla.org/en-US/home>.

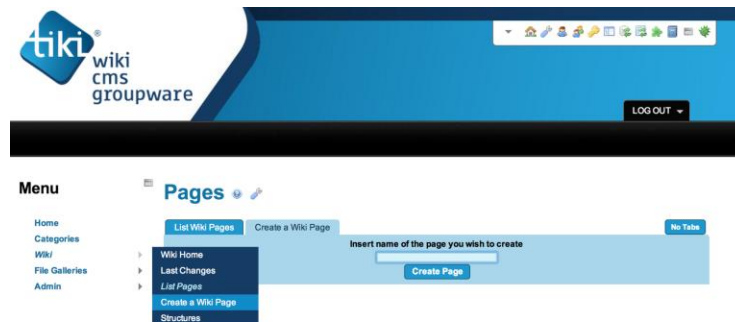


Figure 9 shows TikiWiki home page view.

Tiki Wiki home page is a default wiki page which we can edit. However, we can create a new wiki page altogether and edit it. We can click on Wiki on the left side bar, which pulls up a menu to create a new wiki page, along with the wiki editor. After writing the content, we can add a comment pertaining to the current edit by entering it into the field - "Describe the change you made." Clicking on save will save the page in the database.

TikiWiki Revisions

TikiWiki stores the revisions in the database to enable diff generation and diff comparison for each wiki page. Tiki stores all the wiki pages content in the tiki_pages table. However it keeps track of version history in a table called tiki_history. When an edit happens to a Wiki page, the page details are inserted as a new record into the tiki_history table with a version number (version). The wiki content, along with the edits will be inserted as a data blob in the data field. Again this is related to other wiki engines where the entire contents of the edited page are stored instead of diff metadata. TikiWiki saves the wiki pages and their entire edited content as blobs in the database.

There are various parameters in the tiki_pages table for each of the wiki page record. When a new wiki page is created or edited, a new record will be inserted into the tiki_pages table. The tiki_history table stores all the revision history and the edited wiki page blobs.

2.2.3 Fossil Wiki

In this deliverable, we are going to focus on the built-in Wiki system in Fossil SCM. Fossil is a software configuration management (SCM) tool with built in support for Bug tracking, Wiki system, and a CGI enabled web interface. The software uses SQLite for back-end content storage, which makes all the transaction atomic and fail proof from external disasters. The greatest advantage of Fossil apart from being open-source is that, it is a self-contained single binary file, which has everything to serve the SCM needs. With inherent support for revision control and web interface, Fossil can efficiently implement the features of a Wiki system. A Fossil repository keeps all the built-in wiki pages under version control. Users can create, edit, and delete wiki pages in the wiki section.



Figure 10 shows fossil wiki interface

Version control in Fossil

On a higher level, there are two states for any fossil repository, global state and local state. Any fossil repositories global state consists of an unordered set of artifacts. An artifact can be a text

file, a binary file, the source code or a Meta artifact that contains information or relationship between other artifacts. Hence Fossil artifacts are usually individual files on the file system.

A local state for a Fossil repository, on the other hand, consists of user preferences, user access details, ticket metadata, etc. The local state is usually specific to the repository where are global state is common to all repositories in a project. The local state does not contain artifacts. Wiki pages are an artifact in Fossil. A fossil artifact is the form that every wiki page edit will take.

Wiki in fossil

The user can edit all the pages in the Fossil wiki system, including the home page of the Wiki system. A landing page is just another wiki page. Fossil uses wiki markup to allow users create standalone wiki pages, use in check-in comments, bug reports and bug report comments. There is no separate wiki markup for most of the functionalities in Fossil wiki. Most of the wiki markup is a restricted set of HTML tags, with some simplified markup for common tasks like formatting text as bold, italic, etc. Every wiki page or attachment in fossil consists of Metadata that describes the artifact.

The other wiki pages in fossil are called events. The events show up in the fossil timeline. Events are usually used to announce releases and can be tagged with textual information.

2.3 Deliverable 3: Building a wiki editor in JavaScript.

In this deliverable, I have developed a prototype of a basic MediaWiki syntax editor. We decided to use MediaWiki markup as the wiki markup for our context-sensitive wiki help. Popularity of MediaWiki and ease of use have driven us towards this decision. I have taken a subset of MediaWiki Markup features and have used in this demo.

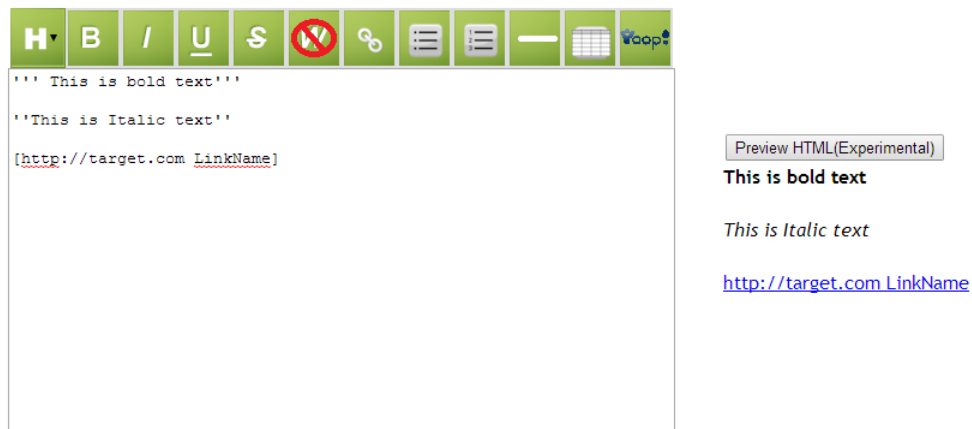


Figure 11 shows the wiki editor demo prototype.

I have created an editor that enables the users to markup their text with these features. I have also included experimental preview functionality. The entire wiki editor requires the main JavaScript framework to be included in the target HTML page.

```
<script type="text/JavaScript" src="js/wikify.js" ></script>
```

The wiki editor requires its parent “Div” to be specified in the target page, something like below:

```
<div class="wikiEditor" id="wikiEditor" data-formName="wikiTextRajesh" data-formAction="something.php" data-formMethod="POST"> </div>
```

The “data-*” attributes define the wiki editor HTML form configuration parameters. The Javascript framework takes care of the rendering the wiki editor dynamically, provided the target “Div” is defined with proper attributes.

Below are the main JavaScript functions that are used to convert plain text, entered by the user, into wiki markup. There are two main functions for end to end processing, they are wikify (), and “getSelection ()”.

Once the user selects some text in the wiki editor, this function is used to get the selected text, its suffix and prefix text. Currently, the text extraction is done for MSIE (older IE browsers) and the newer browsers. Completely different approaches are implemented respectively, to get the text selection. This approach also solves the problem of cross browser compatibility. The “Wikify” function is a utility method used to wrap the wiki prefix and suffix to the text selection. Some wiki markup, like bullets and numbered lists need not require the selection to have suffixes inserted. The editor itself is smart enough to make this decision.

2.4 Deliverable 4: Building Feature set for the context-sensitive wiki help.

This deliverable focus will be on the feature set of the context sensitive Wiki help for Yioop. The following are the features that our Context-sensitive wiki help will include, from the front-end user’s perspective.

Invoking Help

- Users can invoke help with a dedicated “Help” button along the top right box. Clicking on the help button takes the user into the help mode.
- There are two ways help will be displayed to the user. Contextual help (using tooltips) & procedural help (displayed windowpane). Procedural help will be task-oriented explaining a possible task to be performed and how to perform it.
- There will be predefined areas on the page, to which Help can be provided to; these areas are called “Help points.” For invoking procedural-help, help-points will be used.
- A question mark beside an element will be used to allow the users invoke contextual help.
- For procedural help, help content will be exposed on the right pane. The help panel opens up when the user clicks on any “Help point” (procedural help) on the page.
- If the user clicks on a help tip that was specified to provide contextual help, help will be pictured in a tooltip.

Editing Help content

- For procedural help, Users will be able to edit the help content right in the help window pane itself. An edit button pops up when help content is being displayed. When the help content is being edited, edit button will be hidden.
- For contextual help, the help window will be leveraged to edit the tooltip help content for all the tooltips at the same place. It might be hard to get the tooltips editable directly, so the tooltip’s list can be displayed, which could easily be edited by the users.

- There will be a preview button for previewing the changes made to the help content. The preview is applicable to both procedural as well as contextual help.

Wiki system and Wiki editor features

- The primary portion of the Wiki system is the Wiki-editor. The wiki editor will support all the basic functionalities leveraging the MediaWiki Markup.
- The editor will permit users to upload Images/files that can be used to attach to the help articles. The insertion of files/images will be according to markup specified by MediaWiki markup where Filenames will be utilized to track or insert files into support articles.
- If the editor is being used by the user and if there is inactivity for n (configurable) seconds, the wiki-editor pages will timeout.
- Wiki system will be able to save help content revisions with other metadata that includes a time stamp at which the author of the content performs save operation.
- The wiki system will feature a diff engine that displays the diff between two revisions line-by-line.

3. Conclusion

For Deliverable 1, I researched on Context-sensitive help content generation and Context-sensitive help system usage. Results from Deliverable 1 helped me understand how to generate and integrate context sensitive help articles and organizing context-sensitive help. With my findings from researching different context-sensitive help systems, I was also able to understand what kind of context-sensitive help is required for Yioop.

For Deliverable 2, I researched on wiki engines like MediaWiki, TikiWiki and Fossil to understand how the wiki organization, revision control and markup work in the same. After thorough research and discussions, Dr. Pollett and I have agreed to use MediaWiki markup for Yioop context-sensitive help wiki. I will also develop the other features of the wiki, like revision control, diff management engine and back end storage following the footsteps of MediaWiki.

For Deliverable 3, with the findings and agreement from Deliverable 2, I have started to design a JavaScript framework that will support the front end part of the Wiki interface. I have started by building a Wiki editor prototype from scratch, also making it compatible across several old and new web browsers. The Wiki editor is now production ready, and I have already submitted the wiki editor to be integrated with Yioop.

For the final deliverable, I have come up with a design and feature set that my context-sensitive help system will include. I will use the specs defined in the feature set to start building my context-sensitive help system in CS 298. In the next semester, I will implement the context-sensitive help system including, but not limited to the features specified the feature set.

Wiki Markup in MediaWiki, TikiWiki and Fossil Wiki [6] [7] [8]

| Wiki functionality | MediaWiki | TikiWiki | Fossil |
|----------------------|--|--|---|
| Bold Format | "bold" | <u>bold</u> | Text in Bold |
| Italics Format | <i>"italic"</i> | <i>"italic"</i> | <i><i>Italic text</i></i> |
| Underline Format | <u><u>underlined</u></u> | <u>===underline===</u> | <u><u>Underlined text</u></u> |
| Internal Link | [[a link]] [[a link with title]] | ((Name of page)) or ((Name of page link to a page)) or (semantic(link to a page)) or, if activated: CamelCase WikiWords | "[target]" or "[target name]" . |
| External Link | [http://example.org The title] | [http://example.com] or [http://example.com label] | [http://example.com] or [http://example.com label] |
| Headlines | <u>==Section==</u> <u>===Subsection===</u> <u>====Sub-subsection====</u> | <u>-= Titlebar =-</u> <u>! Level 1</u> <u>!! Level 2</u> <u>!!! Level 3</u> <u>!!!! Level 4</u> <u>!!!!! Level 5</u> | <u>HTML based :</u> <u><h1> <h2> <h3> <h4> <h5> <h6></u> <u><h1>Heading 1</h1></u> <u><h2>Heading 2</h2></u> <u><h3>Heading 3</h3></u> <u><h4>Heading 4</h4></u> <u><h5>Heading 5</h5></u> |
| Monospace Format | <code><tt>monospace</tt></code> | <code>--monospace+-</code> | <code><tt>monospace</tt></code> |
| Strikethrough Format | <s>strikethrough</s> | <code>--strike--</code> | <strike>strikethrough</strike> |
| Superscript Format | ^{<sup>superscript</sup>} | <code>{SUP()}text{SUP}</code> | ^{<sup>superscript</sup>} |
| Subscript Format | _{<sub>subscript</sub>} | <code>{SUB()}text{SUB}</code> | _{<sub>subscript</sub>} |
| Images | [[Image:wiki.png]] | {img src=http://foo.bar/foo.jpg} | Image alt text can be placed in [] |
| Aligning Text | <div><center>Centered</center></div> | <div>::centered text::</div> | <div><center>Centered</center></div> |
| Text Indentation | <div>: indented line</div> | <div>leading spaces, if activated.</div> | <div>two leading spaces or tab.</div> |
| Bulleted Lists | <div>* Item 1 ** Item 1.2 * Item 2</div> | <div>* Item 1 ** Item 1.1 * Item 2</div> | <div>* Bullet Bullets are "*" sorrounded by two spaces.</div> |
| Numbered Lists | <div># Item 1 ## Item 1.2 # Item 2</div> | <div># Item 1 ## Item 1.1 # Item 2</div> | <div># Enum Enumerations are "#" sorrounded by two spaces.</div> |
| Definition Lists | <div>; term : definition</div> | <div>;term:definition</div> | <div><dl> <dt>Test</dt> <dd>Tst DD</dd> </dl></div> |
| Horizontal Rule | <div>----</div> | <div>---</div> | <div><hr></div> |
| No wiki formatting | <div><nowiki>Text not wiki</nowiki></div> | <div>~np~This "text" not ===formatted===~np~</div> | <div><nowiki>Text not wiki</nowiki> <verbatim>Text not wiki</verbatim></div> |

References

- [1] M. J. P. E. Helen Wright Hastie, "CONTEXT-SENSITIVE HELP FOR MULTIMODAL DIALOGUE," 2002.
- [2] M. Ellison, "User-centred Design of Context-sensitive Help," 2009.
- [3] "Five Steps to MadCap Flare," in *Fiddlehead Publications*.
- [4] D. J. Barrett, "*MediaWiki*," O'Reilly Media, 2008.
- [5] "Adobe Robohelp context-sensitive help," Adobe, [Online]. Available: http://help.adobe.com/en_US/robohelp/robohtml/WS5b3ccc516d4fbf351e63e3d11aff59c571-8000.html.
- [6] "Fossil Wiki formatting rules," Fossil SCM, [Online]. Available: https://www.fossil-scm.org/index.html/wiki_rules.
- [7] "MediaWiki Help : Formatting," MediaWiki.org, [Online]. Available: <http://www.mediawiki.org/wiki/Help:Formatting>.
- [8] "Tiki Wiki Markup Syntax," TikiWiki CMS, [Online]. Available: <https://doc.tiki.org/Wiki-Syntax+Text>.
- [9] "MediaWiki on Wikipedia," [Online]. Available: <http://en.wikipedia.org/wiki/MediaWiki>.