Simple Wiki Embedded Editing Tool

SWEET

Requirements Analysis

by

Michael Kouyessein
Brian Sullivan
Yuan-Hsun Tang
Fangyan Xu

Computer Science Department
University of Massachusetts Boston

Released on
DEC.17.2007

MENU

Vision Statement

SWEET is a project to develop a WYSIWYG wiki editor that will suggest relevant web links to its users. The main goal is to have these links be suggested as seamlessly as a spellchecker suggests alternate spellings.

The number of available Semantic Web tools has rapidly increased over the past few years. A semantic web editor integrates ontologies and semantic annotation systems. But, current Semantic Web tools are not automated to help wiki users mark up their documents.

Semantic Mediawiki is an extension to Mediawiki, one of the main wiki engines of the world. This is the wiki engine Wikipedia runs on. Semantic Mediawiki allows the specification of typed data inside articles and the typing of links between them in an easy-to-use manner. These semantic enhancements bring to wikis the benefits of today's semantic technologies: more specific ways of searching and browsing.

Customer and Potential Users

- **Customer:** *Jeff Fried*, vice president advanced solutions, FAST.

- **Users:**
  - Any party that uses MediaWiki website to discuss and organize their knowledge base. For example:
    1. Our SWEET Team, a team formed for the Software Engineering course of UMASS BOSTON CS Dept., is using MediaWiki to maintain and schedule our project.
    2. A high school who feels like using MediaWiki to organize their events or recourses (e.g. field trips, graduation ceremony, homework, study plan for SAT).

  - Those who will also feel like contributing their knowledge and discuss articles on that wiki site. Especially, those who are lacking mark-up language background (e.g. language like html, xml). For example, continuing the examples given above:
    1. Those students who also take the same Software Engineering courses, or other students also studying at UMASS BOSTON.
    2. The students, teachers or students' parents of that high school.

Short Stories

1. Installation

It's May 2008 and Sam Shaw who is an Analyst at Environmental Systems Research Institute is excited - the first release of SWEET is now online and he can't wait to try it out. He has heard the name SWEET a year ago when the developing team at UMass Boston has started the project. He has convinced the manager to use SWEET as an extention to their MediaWiki for the whole company’s internal communication. He has also confirmed with Andy who is the administrator at the IT department to install SWEET today.

Andy downloads the installation package and the documentation from sourceforge.net onto the MediaWiki server. Following the installation instructions, he easily and successfully installs the application as an add-on to the MediaWiki.

He opens up the wiki and sees SWEET is there. He calls Sam to start using SWEET and report any problems if he has.

2. Editing new page

Sam is reading an article - "ESRI Survey and Engineering Resources" on his institute’s wiki. He would like to add a new article "GIS for Civil Engineering" which is related to the one that he is reading. He decides to use SWEET because he can put links automatically as he types along.

As Sam begins to type he sees the second word - "Workflow" he just typed in gets underlined. SWEET has suggested several web pages as possible links for "Workflow". He looks at the underlined word "Workflow" and then at the first suggestion which is a link to another article in the same wiki. He decides that is a good link and tells SWEET to include that link and continues on typing. He looks at the next words - "GIS database" that get underlined and the possible links that SWEET provides. He decides and tells SWEET to include the second suggested link which is a page on msn.com for "GIS database".

He continues typing again on "GIS for Civil Engineering" and a phrase of four words - "Streamline field data collection" becomes underlined. He looks at the suggested links and feels that he likes the third suggestion but he is not quite sure the exact contents on that page. So he selects the third suggestion and previews the page. He decides it was not what he is looking for, but while on that page he browses to another page where he would like that phrase to be linked to. He tells SWEET to use that link instead and then keeps going on. Sam does not believe the next word that gets underlined needs a link so he tell SWEET to ignore it.

As Sam types the phrase "mobile and server GIS technologies", he thinks it is necessary to mark it up. So he selects and highlights the phrase and asks SWEET to make suggestions for it. Sam decides the first link is a great one and tells SWEET to include it. He repeats this process until he gets his work completed. He saves his work and feels very proud of his accomplishments by using SWEET.

3. Editing existing page

Fang Harris, Sam’s assistant at Environmental Systems Research Institute also goes to the wiki. There is some additional information that she needs to put on the wiki about the article - "GIS for Civil Engineering" that Sam created earlier. She opens up the article with SWEET. There are various words that get underlined showing they could be marked-up. So she tells SWEET to include the desired links for the words that she would like to mark up and to ingore the links for the words which do not need further information as described in Editing new page. Smoothly, she gets everything done just in five minutes and saves her work. Perfect! It’s exactly what Fang was expecting.

4. Upload Word document

Sam receives an email from his colleague in Boston and it has a Word file attached with the title “BU Bio Lab Construction Effects to Greater Boston Area”.

He skims the file and wants to upload it to the wiki for further use. He decides to use SWEET to load the Word document. SWEET uploads Word documents very easily because it follows the regular file upload convention. He finds the place where he saved the file, selects the file name and tells SWEET to upload. After SWEET has uploaded the file, the contents appear in the editing area and Sam sees some words are underlined with suggested links. Since Sam has to go to a meeting he saves his work and plans to add in the links later.

After the meeting, Sam reopens the uploaded file in SWEET and he can still see the various underlined words. Sam goes through the Editing existing page process and puts appropriate links to the uploaded file. After he has finished selecting the links, he saves his work. Sam is so pleased with what SWEET can do for him; it is really a sweet gift.

Functional Requirements

- SWEET is an extension to MediaWiki
- Suggest links to relevant web pages or internal documents for keywords
    - Internal Link (to the local Wiki site)
        - via Wiki Engine
    - External Link (to any web pages on the Internet)
        - via Search Engine ex. Google
    - Documents within a corporation
        - via FAST
- Support uploading Microsoft Word documents to the wiki.
- A WYSIWYG User Interface
    - SWEET will allow users to edit an article similar to Microsoft word and not in WikiText.
    - SWEET will allow the user to bold, underline and italicize a selection and see their changes in the editor as it will appear on the wiki article. SWEET will also allow for adding hyperlinks, headings and horizontal lines.

Non-Functional Requirements

- **Technical Aspects**
  1. Operating Systems: OS independent
  2. Application Server: MediaWiki 1.11.0 + Semantic MediaWiki 1.0 Extension
  3. Web Servers: Apache or IIS
  4. User Interface: Web-based (Internet Explorer)
  5. IDE: ECLIPSE + PDT (Eclipse PHP Development Tool)
  6. Programming Languages: PHP, JavaScript
  7. Markup Languages: HTML, XML, CSS
  8. Protocols: HTTP, SOAP
  9. Search Engines: MediaWiki Search Engine, FAST, GOOGLE
  10. Databases: MySQL, Entreprise Databases, the WEB
  11. Version Control System: svn://dssg.cs.umb.edu/cs682-3/.
  12. Project Wiki Page: http://sf07.cs.umb.edu/mediawiki/index.php/SWEET

■ Functional Aspects

1. Security: The product shouldn’t create a security bridge in the user’s system
2. Reliability: The product must perform as expected and free of errors
3. Maintainability: The product must be easy to maintain
4. Portability: The product is portable because implemented in interpreted languages
5. Extensibility: The product is easily extendable, because it is object-oriented-based design
6. Reusability: Components of the software can be reused
7. Flexibility: The product must be easy to modify

Development Methodology

■ Team Organization

The SWEET team has four members and we intend assigning project responsibilities according to the bellow tasks decomposition. But above all, every member is a coder, a commenter and a tester of the module he is implementing.

The different roles can be switched, modified or cumulated momentarily depending on particular situations that can arise. However, all new changes should be done in the only goal of sustaining an effective project development flow.

1. Team Manager

Profile:

- schedules meetings and defines their agenda
- assigns tasks to team member
- divides the project into phases
- coordinates code development
- tracks the project progress and updates the project schedule
- maintains communication with the client and the project advisor

2. Design and Integration Manager

Profile:

- manages the software architecture design process (component and class diagrams)
- researches technologies and tools for the development process
- integrates the different tested software codes
- updates code and build versions in the SVN repository

3. Testing and Quality Assurance Lead

Profile:

- verifies that all modules are bug-free before integration
- manages the bug tracking system
- ensures that the product meets the client's requirements
- runs the overall system testing and the acceptance test

4. Technical Writer

Profile:

- notes taker and meetings reporter
- updates the project website (webmaster)
- writes online help scripts
- writes the user's manual (installation, removal and usage instructions)
- writes the development legacy documentation (analysis and design decisions, the project's final state, possible ameliorations and extensions)

■ Implementation Approach

We've decided to follow the Extreme Programming methodology to a certain point since we are working on a mid-sized project. Indeed, Extreme Programming is adequate for our project because it emphasizes customer involvement and promotes team work. It is also suitable for projects with dynamic requirements and leads to a development process that is more responsive to the customer needs. Our project will then experience greater success and developer productivity.

The main values of Extreme Programming are:

1. **<u>Communication:</u>**
   Building software systems requires communicating system requirements to the developers of the system .The goal is to give all developers a shared view of the system which matches the view held by the users of the system. To this end, Extreme Programming favors simple designs, common metaphors, collaboration of users and programmers, frequent verbal communication, and feedback.

2. **<u>Simplicity:</u>**
   Extreme Programming encourages starting with the simplest solution. Extra functionality can then be added later. Coding and designing for uncertain future requirements implies the risk of spending resources on something that might not be needed.

3. **<u>Feedback:</u>**
   Feedbacks are of major importance in Extreme Programming. Feedback from the system: by writing unit tests, or running periodic integration tests. Feedback from the customer: the functional tests (acceptance tests) are written by the customer and the testers. Feedback from the team: when customers come up with new requirements in the planning game the team directly gives an estimation of the time that it will take to implement.

4. **<u>Courage:</u>**
   Several practices embody courage. Courage enables developers to feel comfortable with refactoring their code when necessary. Another example of courage knows when to throw code away. Also, courage means persistence.

5. **Respect:**

The respect value manifests in several ways. In Extreme Programming, team members respect each other because programmers should never commit changes that break compilation, that make existing unit-tests fail. Members respect their work by always striving for high quality and seeking for the best design for the solution at hand. Nobody on the team should feel unappreciated or ignored. This ensures high level of motivation and encourages loyalty toward the team, and the goal of the project.

Development Process

Extreme Programming also describes four basic activities that are performed within the software development process:

1. **<u>Coding</u>:**
   The only truly important product of the system development process is code

2. **<u>Testing:</u>**
   One cannot be certain of anything unless one has tested it

3. **<u>Listening:</u>**
   Programmers do not necessarily know anything about the business side of the system under development

4. **<u>Designing:</u>**
   Creating a design structure that organizes the logic in the system. Good design will avoid lots of dependencies within a system

To apply the Extreme Programming recommendations effectively, we intend the following:

- Assignation of the different project roles relatively to each member's willingness and skills.

- At least two team meetings will be held each week

- A comprehensive and realistic project schedule will be established that will take into account task planning, risk management and development reviews

- Constant communication will be maintained between the team members

- The client and the advisor will be weekly notified of the project advancement

- Coding, testing and integration rules will be defined to show:

  - Object model of coding (hierarchy and levels of abstraction)

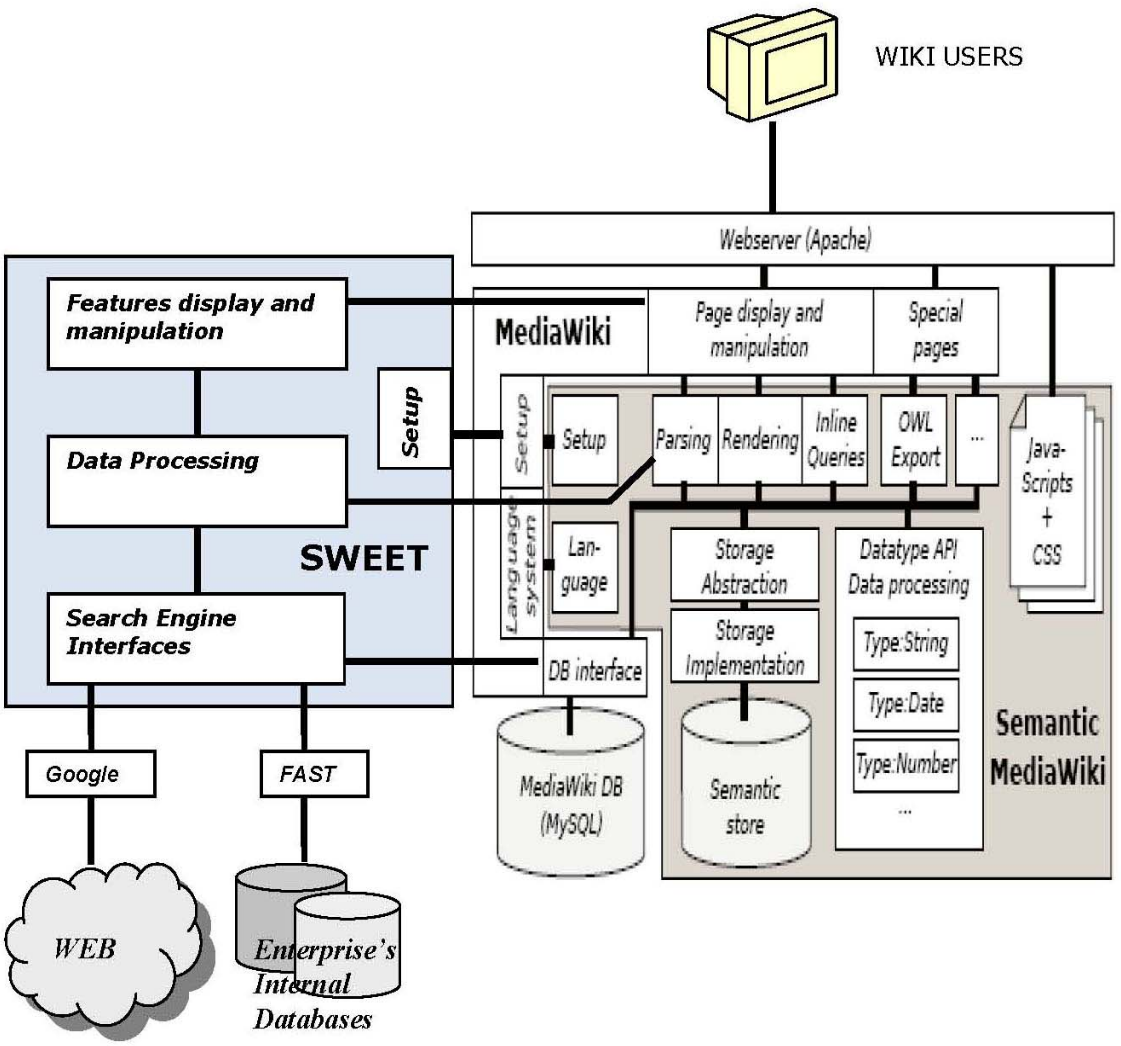
  - Incremental development process

  - Evolvement of architectural and software designs

  - Commenting and documentation of codes and classes

  - Continuous integration and testing

Architecture



ARCHITECTURE DESCRIPTION

SYSTEM LAYERS	SYSTEM COMPONENTS	COMPONENT FUNCTIONALITIES	COMPONENT CODING LANGUAGE
Presentation Layer	Features display and manipulation	- creates an additional tab - creates editing and searching buttons - creates editing area - indicates a word has automatically suggested links available - contextual menu for links selection - menu for uploading a word document - rendering search or query results - page preview - annotations insertion - online help	AJAX (PHP, JAVASCRIPT, CSS, XML DOM,HTML)
Business Layer	Setup	- adds SWEET as an extension to MediaWiki	PHP
	Data processing	- text scanning - parsing html contents to wikitext - search logic in databases and ontology - result sets sorting and ranking - upload word documents	PHP
Database Access	Search engine interfaces	- database APIs knowledge - search initiation	PHP, SOAP

Deliverables

- **Documentation:**
  - User’s manual
  - SWEET API

- **Software:**
  - Source code package
  - Installer script

Risk Analysis

1. The possible customers' data for testing might be classified.
Solution: Use fake data or sign declaration of secrecy.

2. The customer changes his mind about the requirement or adds extra functionalities.
Solution: Politely decline him with understandable reason, or try to accommodate the extra functionality.

3. Some functionality is too complex to be fulfilled.
Solution: Ask customer to accept a possible alternative.

4. Slow performance of certain team members or frequent absences.
Solution: Understand the cause and prevent it from reoccurring.

5. Instable server.
Solution: Make sure to back up the latest source.

6. Frequent deadlines miss.
Solution: Figure out the reason(s) and reschedule.

7. Team is not able to reach an agreement during a discussion.
Solution: Consult with the customer and Prof. Bolker or have a team vote.

8. Lacking testers for our product
Solution: We could have our classmates as our testers.

9. Unable to watch testers use our product.
Solution: Unfortunately, we cannot afford proper testing software to help us record their use and we will not be able to watch our users. But we will have the testers submit a questionnaire regarding their interaction.

Project Schedule

■ OCT 2007

Task	**Assigned To**	**Due Date**	**Status(Y/N)**	**Remarks**
Project Name Brainstorm	Team	OCT.11.2007	Y	
Project Name Draft	Brian	OCT.11.2007	Y	
Installing Media Wiki	Team	OCT.11.2007	Y	
Back Up Project Files in Repository	Michael	OCT.13.2007	Y	
Vision Statement	Michael	OCT.16.2007	Y	
Project Logo Draft	FangYan	OCT.16.2007	N	Under Construction
Risks Analysis	Yuan-Hsun	OCT.16.2007	Y	
Preliminary thoughts on process	Brian	OCT.16.2007	Y	
Media Wiki Installation on computers	Team	OCT.16.2007	Y	
Scenario Drafts	Each	OCT.18.2007	Y	
Website Layout Draft	Brian	OCT.21.2007	Y	
Project Logo Draft 1&2	FangYan	OCT.23.2007	Compass & Crossed Hands	
Project Name Finalized	Team	OCT.23.2007	N	Waiting for reply from client
Scenario Final	Team	OCT.25.2007	N	Waiting for reply from client
Logo Final	Yuan-Hsun	OCT.30.2007	N	Waiting for reply from client
Beginner PHP tutorials	Team	OCT.30.2007	Y	
Presentation Draft Version	Brian/FangYan	OCT.30.2007	Y	Commented by Prof.

1&2				Bolker
Presentation Draft Version 3	Michael	OCT.30.2007	Y	
Merge Presentation Draft 1,2 and 3	Yuan-Hsun	OCT.30.2007	Y	

■ NOV 2007

Task	Assigned To	Due Date	Status(Y/N)	Remarks
Presentation Final Version	Team	NOV.01.2007	N	Stay open to revise till NOV.05.2007
Presentation Rehearsal	Team	NOV.05.2007	Y	
Project name approval	Team	NOV.06.2007	Y	
Project name approval	Team	NOV.06.2007	Y	
Scenario approval	Team	NOV.06.2007	Y	
Intermediate PHP tutorials	Team	NOV.06.2007	Y	Michael found a link for Creating Tab on MediaWiki
Prensentation to veture capitals	Team	NOV.06.2007	Y	
Use Case drafts	FangYan	NOV.13.2007	Y	
Advanced PHP tutorials	Team	NOV.13.2007	N	Rearrage to how to parse Target String to suggest possible links
Requirements draft	Team	NOV.13.2007	Y	Discussed in class; Still need to discuss with Jeff
Testing PHP with add-on to WikiMedia	Yuan-Hsun	NOV.13.2007	N	This will be realized by "Creating an empty Tab" Due NOV.20.2007
Use Cases	Team	NOV.20.2007	N	Waiting for Jeff's

				reply
List of WikiMedia mark up that will be supported by SWEET	Team	NOV.20.2007	N	Waiting for Jeff's reply
List of semantic MediaWiki mark up that will be supported by SWEET	Team	NOV.20.2007	N	Waiting for Jeff's reply
Creating Tab	Yuan-Hsun	NOV.20.2007	Y	Found FCK Editor instead
Draft - use cases and short stories	FangYan	NOV.28.2007	Y	
Draft - Schedule, Functional requirements and Vision Statement	Brian	NOV.28.2007	N	
Draft - users, critique and risk analysis	Yuan-Hsun	NOV.28.2007	N	
Draft - Development methodology, non-functional requirements and deliverables	Michael	NOV.28.2007	Y	

■ DEC 2007

Task	**Assigned To**	**Due Date**	**Status(Y/N)**	**Remarks**
1st Revision - use cases and short stories	FangYan	DEC.5.2007	Y	
1st Revision - Schedule, Functional requirements and Vision Statement	Brian	DEC.5.2007	Y	
1st Revision - users, critique and risk analysis	Yuan-Hsun	DEC.5.2007	N	critique part is not ready
1st Revision - Development	Michael	DEC.5.2007	Y	

methodology, non-functional requirements and deliverables				
2nd Revision - use cases and short stories	FangYan	DEC.12.2007	Y	
2nd Revision - Schedule, Functional requirements and Vision Statement	Brian	DEC.12.2007	Y	
2nd Revision - users, critique and risk analysis	Yuan-Hsun	DEC.12.2007	Y	
2nd Revision - Development methodology, non-functional requirements and deliverables	Michael	DEC.12.2007	Y	
Present Requirements Analysis to Jeff	Brian	DEC.13.2007	N	
Present Requirements Analysis to Jeff	Brian	DEC.17.2007	Y	
Submit signed Requirements Analysis to Prof Bolker	Brian	DEC.20.2007		

■ JAN 2008

Task	Assigned To	Due Date	Status(Y/N)	Remarks
Proof of concept - create new tab in MediaWiki	Brian	JAN.29.2007		
Proof of concept - editable WYSIWYG in MediaWiki	Brian	JAN.29.2007		
proof of concept - query semantic and display	Michael	JAN.29.2007		

results				
proof of concept - query Google in PHP and display results	Michael	JAN.29.2007		
proof of concept - query FAST in PHP and display results	Michael	JAN.29.2007		
Proof of concept - parsing word docs in PHP	Yuan-Hsun	JAN.29.2007		
proof of concept - create a test/example msi	Yuan-Hsun	JAN.29.2007		
proof of concept - create a pop-up that displays selectable items in PHP	FangYan	JAN.29.2007		
Proof of concept - parsing algorithem	FangYan	JAN.29.2007		Google Search API parsing results Options [1]

■ FEB 2008

Task	Assigned To	Due Date	Status(Y/N)	Remarks
Phase 1 - semantic search of user suggested word in a new "SWEET" tab displaying results in a pop-up box	Michael	FEB.12.2007		
Phase 1 - remove known bugs	Michael	FEB.19.2007		
Phase 1 - msi	Yuan-Hsun	FEB.19.2007		

■ MAR 2008

Task	Assigned To	Due Date	Status(Y/N)	Remarks
Phase 2 - automatically	Fangyan	MARCH.4.2007		

suggest links				
Phase 2 - msi	Brian	March.11.2007		
Phase 2 - remove known bugs	Fangyan	March.11.2007		
Phase 3 - add word document uploads	Yuan-Hsun	March.25.2007		

■ APR 2008

Task	**Assigned To**	**Due Date**	**Status(Y/N)**	**Remarks**
Phase 3 - msi	FangYan	APRIL.1.2007		
Phase 3 - remove known bugs	Yuan-Hsun	APRIL.1.2007		
Phase 4 - editable WYSIWYG	Brian	APRIL.15.2007		
Phase 4 - msi	Michael	APRIL.22.2007		
Phase 4 - remove known bugs	Brian	APRIL.22.2007		
Finish Documentation	Yuan-Hsun	APRIL.23.2007		
Draft Presentation	Brian	APRIL.30.2007		
Finish Testing	TEAM	APRIL.30.2007		

■ MAY 2008

Task	**Assigned To**	**Due Date**	**Status(Y/N)**	**Remarks**
Finalize Presentation	Team	MAY.07.2007		
Present Project	Team	MAY.14.2007		

Use Cases

■ Downloads Installation Package and Documentation

Description: The administrator wants to download SWEET installation package and the documentation.
Actor: SWEET administrator
Entry Condition:

Installed MediaWiki (version 1.11.0)

Configured Web Server such as Apache or IIS

Installed PHP version 5.0 or later

Configured Database Server MySQL (version 4.0 or late)

Browsed at Sourceforge.net.

Exit Condition:

The SWEET administrator has successfully downloaded the SWEET installation package and the documentation.

Used in scenarios:

- Installation

User Action	**System Response**
1. Types the keyword "SWEET" to search for the software	2. Displays a list of software related to SWEET
3. Clicks on the name of the first display which is the exact "SWEET"	4. Displays the big green download sign
5. Clicks on the big green download sign	6. Displays the installation package

7. Clicks the latest release	8. Display the installer zip file
9. Clicks on the file name	10. Displays a list of mirrors
11. Clicks on the nearest mirror	12. Prompts the dialogue box for “Open with” or “Save to Disk”
13. Clicks “Save to Disk” and the “OK” button, selects the destination folder of where MediaWiki Server is located	14. Installation package and the documentation is downloaded onto MediaWiki server.

Alternatives: · If the administrator gives up downloading the package he just returns back to the entery point.

Exceptions: · If the installation package is not there on the Sorceforge.net

Comments:

■ Installs SWEET

Description: The administrator wants to install SWEET onto MediaWiki
Actor: SWEET administrator
Entry Condition:

- Downloaded the SWEET installation package and the documentation

Exit Condition:

- The SWEET administrator has successfully installed SWEET onto MediaWiki

Used in scenarios:

- Installation

User Action	**System Response**
1. Opens up the folder where MediaWiki Server is located and finds the SWEET installation files	2. Displays the installer program
3. Clicks on the SWEET installer program	4. Pops up the installation wizard
5. Follows the wizard to select the appropriate installation destination and confirms it	6. Shows installation progress message

7. Follows the wizard again with confirmation	8. Shows SWEET has installed successfully

Alternatives:

- If the administrator chooses the wrong installation destination before he confirms it at step 5, he can reselect the desired installation destination. However, if he has already confirmed it and goes to step 6, he has to reinstall the program.

Exceptions:
Comments:

■ Provides Links to Words by SWEET

Description: SWEET makes suggestions automatically to the word/words that the user types in.
Actor: SWEET user
Entry Condition:

- Under SWEET Editing mode

Exit Condition:

- Displays the suggested links automatically and successfully to the word/words that the user types in

Used in scenarios:

- Editing
- Editing existing page
- Upload Word document

User Action	**System Response**
1. Types in SWEET.	2. Notices a word/words for which it knows some possible links and informs the user.

Alternatives: · If the user does not want to add the link to the word/words, he/she can just ignore it.
Exceptions:
Comments:

■ ProvidesLinksToWordsByHand

Description: SWEET gives possible suggested links to user selected word/words.
Actor: SWEET user
Entry Condition:

- Under SWEET Editing mode

Exit Condition:

- Possible links is displayed to user selected word/words successfully

Used in scenarios:

- Editing
- Editing existing page
- Upload Word document

User Action	**System Response**
1. Types in SWEET, selects the word/words that need to be marked-up and tells SWEET to make suggestions	2. Informs the user with possible links

Alternatives:
Exceptions:

Comments:

■ Previews Suggested Link

Description: The user wants to preview the contents of the suggested link
Actor: SWEET user
Entry Condition:

- SWEET has informed the user a list of suggested links for the word/words

Exit Condition:

- Previews the contents of the link

Used in scenarios:

- Editing
- Editing existing page
- Upload Word document

User Action	**System Response**
1. Selects the link and decides to preview the link	2. Displays the web contents of that link

Alternatives:
Exceptions:
Comments:

■ Selects Link

Description: The user wants to add the link for the word/words
Actor: SWEET user
Entry Condition:

- User has seen a list of suggested links

Exit Condition:

- The link is selected successfully

Used in scenarios:

- Editing
- Editing existing page
- Upload Word document

User Action	**System Response**
1. Tells SWEET to include the desired link.	2. The word/words is marked-up with that link successfully.

Alternatives:
Exceptions:
Comments: The desired link can be the first link, the second link and etc. The desired link can be the link from internal wiki, from Google search and Fast search. It can also be the link from the previewed page.

■ Includes Link from Previewed Page

Description: The user wants to mark-up the word/words with the link that he/she sees on the previewed page
Actor: SWEET user
Entry Condition:

- User has previewed the suggested link

Exit Condition:

- Link added to the word/words successfully

Used in scenarios:

- Editing
- Editing existing page
- Upload Word document

User Action	**System Response**
1. Sees a good link on the previewed page and asks SWEET to include that link.	2. The word/words is linked to that page successfully.

Alternatives:
Exceptions:
Comments:

■ Ignores Links

Description: The user does not want to put links for the word
Actor: SWEET user
Entry Condition:

- Under SWEET Editing mode

Exit Condition:

- No links added to the word

Used in scenarios:

- Editing
- Editing existing page
- Upload Word document

User Action	**System Response**
1. Types the word in SWEET	2. The word gets underlined and has a few suggested links
3. Indicates to ignore the link	4. The word has no links added

Alternatives:
Exceptions:
Comments:

■ Opens Up Exiting Page

Description: The user wants to open the existing page in SWEET
Actor: SWEET user
Entry Condition:

- MediaWiki is running

Exit Condition:

- Opens up the existing page in SWEET successfully

Used in scenarios:

- Editing existing page
- Upload Word document

User Action	**System Response**
1. Indicates to open up the page in SWEET.	2. The SWEET editing interface is displayed and various words in the page get underlined.

Alternatives:
Exceptions:
Comments: This use case opens up an existing page and there should be another use case which opens up a blank page.

■ Uploads Word Document

Description: The user wants to upload a Microsoft Word document to the MediaWiki
Actor: SWEET user
Entry Condition:

- MediaWiki is running and under SWEET editing mode

Exit Condition:

- Microsoft Word document successfully uploaded to the MediaWiki

Used in scenarios:

- Upload word document

User Action	**System Response**
1. Indicates SWEET to upload the Word document	2. Displays the dialogue box of where to select the file to be uploaded
3. Finds the right Word document and selects the file name	4. Displays the contents of that document to the MediaWiki with some words been highlighted and with suggested links

Alternatives:
Exceptions:

- The Word document does not exist

Comments:

Critique and Analysis of Work to Date

CS682 Homework

Homework 1

We learned how to make ourselves a personal introductory website, and a professional resume to apply for a job. By reading Brooks' The Mythical Man Month we have a general idea of what kind of problems we might encounter and why they happen and how to deal with them.

Homework 2

We learned how to evaluate a project proposal. It is hard to have a clear understanding of a project with a short presentation.

Homework 3

We learned how to make a decision with limited information. We wrote a professional cover letter. We gained experience trying to prove ourselves in an interview for a job we are interested in.

Homework 4

We learned how to begin a software project. We had to figure out the objective of our project. So, we could choose a proper name and render a logo, which really took us a long time. For this assignment we had to use our creativity for designing a logo and selecting a good name. We also designed a website and met our client. So, we had an opportunity to practice these topics, which we barely had experienced before.

Homework 5

We learned how to write short stories and to imagine how possible users will use our software. We have four stories in total. They are Installation, Editing new page, Editing existing page, and Upload Word document. It was difficult to distinguish the Editing new page and Editing existing page because it's all about "Editing".

About SWEET Project

- **OCT 2007**

We had an important meeting with our client, Jeff on OCT.03.2007. After the meeting several parts of the project were still not decided. Fortunately, the day after Professor Bolker informed us that he had a conversation with Jeff. They decided we would make a MediaWiki extension. The extension would help users easily edit the content of the wiki, automatically suggest relevant links for words in the article and import a Word document into an article. Once we thought we had a good understanding of what our project would do we thought it would be best to choose a good name. All the other groups already had a name and we felt a bit behind. After several brainstorming sessions as a team we were able to put together SWEET. Once we had our name we had to work on our products logo. Designing a proper logo was not a natural task for us, as we all have been directing our lives towards becoming software engineers. It required a skill set that none of us have been developing. After countless doodles we came up with a design that we are very proud of. Next we focused on getting a template for our projects website that was clean, sophisticated and uniform. While we were developing the website we were also working on our short stories. The most difficult aspect of the website was (is) keeping it updated.

- **NOV 2007**

The first thing was to conceive the Short Stories in order to capture the functional requirements of the system from the user’s perspective. The second thing was to give a presentation about the project to date to seek venture capital. Since the goal of the Software Engineering class is not only to learn software development, but also to learn communications with various people in and outside of the development team. This is a great opportunity that we had to present our project to the “potential investors”. This is a very important part of the real business. Without investment and marketing it will be too difficult and meaningless to build the software.

- **DEC 2007**

We had two important missions within this month. One is to complete the Short Stories and the Use Cases which is the detailed description of how the users will interact with the system. We are lucky to have Fangyan in our team because she had some experience in this field. She sketched the basic outline and revised them carefully. The other thing is to write the document for the Requirements

Analysis. We split the work and assigned several parts to each of the team member. Fangyan takes care of the revision and polishing for the Short Stories and the Use Cases; Brian arranges the Schedule and completes the Vision Statement and the other functional requirements; Michael deals with the project architecture, development methodology and the non-functional requirements; YuanHsun takes care of the review and possible Risks Analysis and Critique and Analysis of Work to Date. We also had team meetings to discuss about the document and helped each other to revise possible mistakes. We are very confident that we can get this document done successfully and professionally. We are the SWEET team and we are enjoying doing our project.