Parag Walimbe

Doodle – Remote Tagging, Storage, Retrieval and Display of User Annotations within a Browser Environment

Master thesis based on the examination and study regulations for the Master of Engineering degree programme
Information Engineering
at the Department of Information and Electrical Engineering
of the Faculty of Engineering and Computer Science
of the University of Applied Sciences Hamburg

Supervising examiner : Prof. Dr. Thomas Schmidt
Second examiner : Prof. Dr.rer.nat Hans-Jürgen Hotop

Day of delivery October 30th , 2006

Parag Walimbe

Title of the Master Thesis

Doodle – Remote Tagging, Storage, Retrieval and Display of User Annotations within a Browser Environment

Keywords

Annotation Engine, Browser Environment, Remote tagging , Retrieval, Display, Sharing, Making Private, Caring for Orphan annotations, Adopting shared annotations, Annotation storage structure, Content addressing, Firefox toolbar, XUL, AJAX

Abstract

Multiple sources of data and information for a topic are available on the World Wide Web. A reader reading content with a motive to understand it writes down notes, observations, comments or other types of self explanatory text for better understanding of the content. These notes, observations, comments, self explanatory notes are collectively called annotations. This thesis has explored an approach to develop an annotation engine that enables a user to insert, store, retrieve, delete, share, and adopt shared annotations on web content in a browser based environment. The thesis has also formulated and defined data structures, algorithms and methodologies to develop such an annotation engine.

Parag Walimbe

Thema der Masterarbeit

Doodle – Remote Tagging, Storage, Retrieval and Display of User Annotations within a Browser Environment

Stichworte

Annotation Engine, Browser umgebung, Remote Tagging, Wiederherstellung, Anzeige, Freigeben, machen privat, verwalten verwaister anmerkungen, übernehmen freigegebener anmerkungen, speicherstruktur der anmerkungen, inhalte addresieren, Firefox toolbar, XUL, AJAX

Kurzzusammenfassung

Das World Wide Web bietet vielfältige Quellen von Daten und Informationen. Leser, die solche Dokumente verarbeiten und verstehen wollen, schreiben gewöhnlich Bemerkungen, Kommentare oder Beobachtungen zur Unterstützung des Verarbeitungsprozesses nieder. Solche Bemerkungen, Kommentare oder Beobachtungen nennt man gemeinhin Annotationen. Diese Arbeit untersucht einen Ansatz zur Entwicklung eines Annotationssystems,welches den Benutzern eines Web-Browsers erlaubt, den betrachteten Web-Seiten Annotationen hinzuzufügen, diese zu speichern oder zu löschen, zu teilen oder geteilte anzupassen. Die Arbeit konzipiert und definiert weiterhin Datenstrukturen, Algorithmen und Methoden für die Realisierung eines solchen Annotationssystems.

INDEX

List of Figures

Chapter 1 - Introduction

Four common activities surrounding documents are reading, annotating, collaborating, and authoring. [06] With the rapid growth of the World Wide Web, access to multiple sources of data and information for a specified topic has become easy. Search engines have further enhanced the availability of such data and information. Thus the acquiring and reading of content has become easier. However the comprehension and understanding of these sources and content of data and information is left to the viewer and reader of the document. A user reading, reviewing or viewing such content needs to write down notes, observations or comments that helps him better understand or review the same document faster on his next read, review or revisit. These notes observations, comments can be collectively called as annotations. Tools like the adobe acrobat are available to write these annotations on atomic documents like the pdf format. But fewer methods are available to insert, store, reload or recall these annotations in a browser based environment.

Every user who retrieves or views specific web content has to individually review the whole document again to comprehend or understand it. As against to this if an expert in a domain has reviewed the content/document and has inserted his annotations or expert comments on the content then later readers may find his annotations to the document useful. Each viewer may not agree to the annotations of the expert and may want to add his own annotations.

1.1 User annotations on web content

Marshall [08] has classified paper-based annotations into 4 groups based on whether the annotation content is *explicit* to another reader (e.g., a scribbled note) or *implicit* (e.g., yellow highlighter ink implying importance) and whether the annotation's anchor is a *margin* anchor (e.g., asterisks, a note scribbled to the side of a paragraph with no other marking) or a *range* anchor (e.g., highlighted text, circled word). Figure 1 shows these different types of annotations.

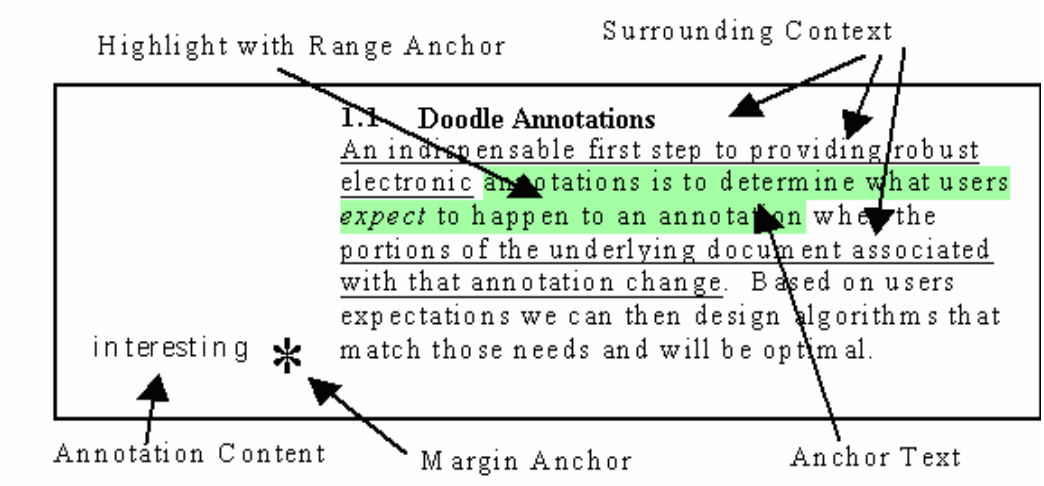

Figure 1: Annotation example showing anchor types and surrounding context.

1.2 Browser environment

The title of the thesis refers to a browser environment. The meaning of a browser environment is an environment which is experienced in a standard browser like mozilla firefox, internet explorer. The following is a list of some of the most notable features found in a browser environment (but not a comprehensive list).

Browsers have support for standards like

- HTTP and HTTPS
- HTML, XML and XHTML
- Graphics file formats including GIF, PNG, JPEG, and SVG
- Cascading Style Sheets (CSS)
- JavaScript (Dynamic HTML) and XMLHttpRequest [15]
- Cookie
- Digital certificates

Browsers have fundamental features like

- Bookmark manager
- Caching of web contents
- Support of media types via plugins such as Macromedia Flash and QuickTime

Browsers have usability and accessibility features

- Autocompletion of URLs and form data
- Tabbed browsing
- Spatial navigation
- Caret navigation
- Screen reader or full speech support

The features that concern this thesis are

- Access to content and ability to address each element of the content loaded into the browser. (via some methods like the Document object model [14])
- Ability of remote tagging via the insertion of imported content which is not native to the original content
- Ability to access the native functions of a browser. (Like the alert popup)
- Support for some scripting language for content accessing, addressing and modifying.

1.3 Remote tagging, storage, retrieval of user annotations

User annotations are inserted on or in reference to the original content. The annotation text and the original content are different entities and thus are stored separately. When the original content is loaded, at that time the annotation text is to be retrieved and inserted in the original content with proper reference to the original content. This decoration of original content with the annotation text is called remote tagging.

Content that is loaded, understood and displayed by a browser is normally coded into some markup language like html for the sake of proper presentation (Raw text and images or other multimedia files are loaded without markup language but are still addressable via Document object model). This markup language consists of tags that have been defined for proper presentation and display of the content. Any addition of content to this tagged content needs also to be in the form of tags (Unless it's an addition to the raw content of raw text). Thus the annotation text which is to be loaded in context to the original text must be wrapped in tags and then inserted in the original content in the browser as shown is figure 2. This wrapping of the annotation text will enable the annotation text to be displayed by the browser in a proper and consistent way with the original text.

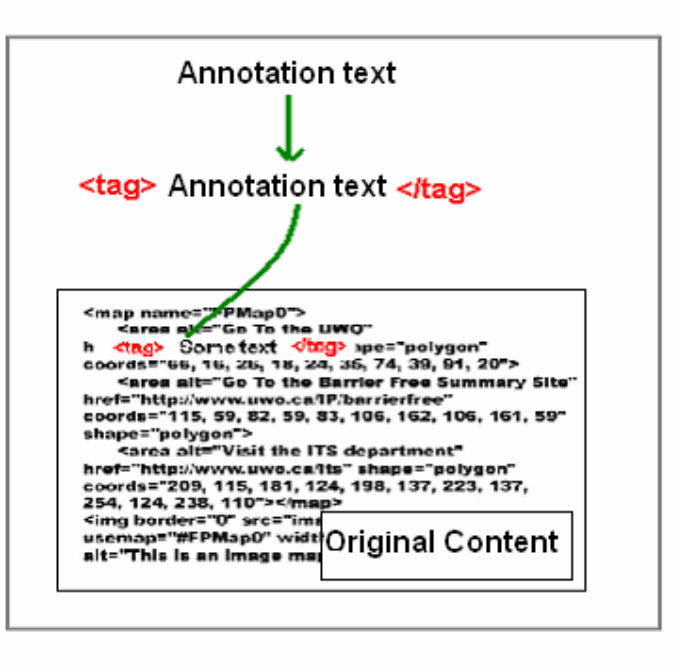

Figure 2 : Wrapping of annotation text into tags

The annotations need to be stored independent of the original content because they are both independent entities. On the event of the load of the original content in the browser environment, the annotation text must be retrieved from the location it is stored and be displayed with respect to its relation and location to the original content.

1.4 Target Audience

The target audience of this thesis is study groups. Groups of people, who study the same lecture or study notes of the same teacher will largely benefit from the use of the thesis. The concept of social networking that has been introduced later in the chapters helps study groups to share their comments and or annotations to fellow students. This provides a environment for collaborative learning. The teacher of the study notes also uses the annotation engine to insert annotations to aid the process of teaching.

Normal everyday web users also benefit from the use of the annotation engine. The user use the annotation engine as post it notes and also insert annotations on web content thus being able to share their annotations with other users by the use of the social network.

1.5 Overview of this document

This document discusses the approach of making an annotation engine. It starts by discussing about the problem space in the next chapter. Problems like collaboration on web content, remote tagging, remote evaluation and annotation presentation and personalization are discussed. These are the major problems that are faced and thus need to be addressed.

In the related works chapter the thesis list various kinds of related work done by other individuals or groups in the past. The chapter explains the various features of the related works along with discussing the advantages and disadvantages. The motive of this chapter is to finally discuss about all related works and trying to find a candidate related work to be used as a basis for further work.

The results of the discussion of the related works are used as a basis for the chapter on system design where the process of the designing of the system is listed. The chapter gives system architecture and later the database design. This chapter also discusses the various design decision that are taken and the reasons for them to be taken.

Chapter on Implementation and the chapter on testing give details of the process of implementation of the design and the testing of the implemented work. User comments are listed at the end of the chapter.

Possibilities of further work are stated in the further works section

Chapter 2 - Problem Description

2.1 Collaboration on web content

Collaboration can be defined as mutual engagement towards a goal, rather than simply a division of labor among members of a group who complete their tasks individually [01]. Thus collaboration is essentially working together to achieve a common goal. Collaboration on web content can be of various types but 2 major types of collaboration can be considered in detail in the context of web content in a browser environment, those two types are "collaboration between human and machine" and "collaboration between human and human".

2.1.1 Collaboration between human and machine.

A. Placing decorative anchors

Content insertion in the web content that is placing decorative anchors in the web content in a browser based environment has to deal with the issue of node or content addressing. The exact node address where the new content needs to be inserted is required to insert content at the location. This exact node addresses needs to be calculated and remembered because it is needed for reloading the inserted content on the content reload.

The exact node address can be accessed via the Document Object model [14]. The exact node address is made up of a various parameters which make up the hierarchy of the DOM. I have used in the following Figure 3, the DOM inspector tool of the firefox browser [16] to show the various parameters that make up the exact address to a text node on the home page of Google.

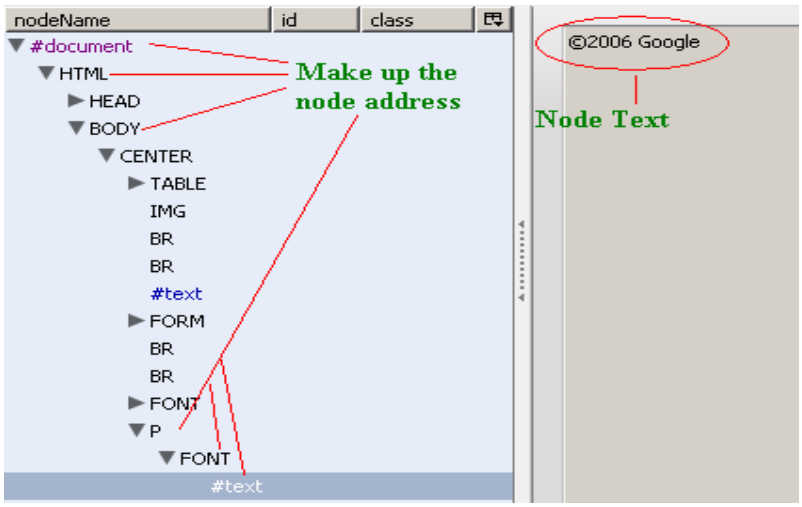

Figure 3 Parameters that make up the node address.

There is obviously a problem with the non DOM content like the raw text files or image files. These non DOM contents are not accessible via the DOM.

Dynamic content also causes problems in the insertion of annotations. When one node address is stored for the sake of some content and on a reload the content is changed then the node address is no more exact and thus the reloading with the exact address will not work. Thus error resilience is required. Dynamic content needs to be cared for the changed exact addresses.

Another issue of content insertion is user management and user rights management. For this purpose roles need to be created which define the access rights and methods of the users. As stated in the paper Structured Cooperative Authoring on the World Wide Web [02] user roles like writer, reader and or manager need to be defined. The roles define the access rights to the content. They also define the access priorities in case of conflicts.

Managing user roles and access right and finding exact node or content address and saving this node or content address for future purposes is an issue to be handled by human and machine collaboration.

B. Sharing inserted content with other users.

There are different types of content sharing approaches like peer to peer content sharing or client server based content sharing.

Peer-to-peer (P2P) can be best described as the direct exchange of data between two computers or peers in a common network. In a P2P network, all client computers in the network, known as nodes, are considered to be equal in their capacity for sharing resources with other network members. Unlike the traditional Client-Server system [17]

Existing P2P systems employ two common architectures, namely centralized and decentralized [18].

Decentralized P2P systems

Pure P2P

Advantages - No central server, Higher fault tolerant, Simpler architecture
Disadvantages - More network, Resource consumption, Low scalability

Server-mediated P2P

Advantages- Less network resource consumption, High scalability
Disadvantages- Central server dependent, less fault tolerant

Centralized P2P Systems

For the centralized P2P system, a central server is present to provide the directory service, yet the server does not involve in file transfer.

In pure peer to peer approach user identity management is an issue that needs to be addressed. In the case of server mediated P2P approach the server can take care of the rights and identity management.

Client Server Systems

A client server system is a distribution concept. The client sends requests to the server and the server serves the request made by the client. Various types of protocols are used for the communication between the client and the server.

In general if inserted content by a user has to be shared, then for that the rights and identity of the user who is sharing the content needs to be registered and so needs the rights and identity of the user adopting the shared content be registered. If this is not done then there won't be any accountability. Thus access and change rights have to be taken care of.

Here is needed a collaboration between the human and machine where the machine is responsible for the identity and the rights management and the user must provide the machine with the details of the user whom he is sharing his content to or from whom he is receiving the content from .

2.1.2 Collaboration between 2 or more humans

A. Using shared inserted content for better searchabiltiy

One issue commonly faced by users of the World Wide Web is "Lack of useful content when needed." [01]. Useful content can be made available by better search results. Better search results can be attributed to better cataloging. Metadata serves the same functions in resource discovery as good cataloging does by [10]

- allowing resources to be found by relevant criteria;
- identifying resources;
- bringing similar resources together;
- distinguishing dissimilar resources;
- Giving location information.

Metadata is structured information that describes, explains, locates, or otherwise makes it easier to retrieve, use, or manage an information resource [11]

The html Meta tags have a specific set of Meta tags defined that consist of Meta tags and keywords. This is a problem because not much Meta data can be formulated into the constraints that html Meta data has. Thus the Dublin core needs to be considered because it has a structured set of meta data and this makes the meta data more readable by machines.

The Dublin Core metadata element set is a standard (NISO Standard Z39.85-2001) for cross-domain information resource description. In other words, it provides a simple and standardized set of conventions for describing things online in ways that make them easier to find. [12]

Simple Dublin Core comprises fifteen elements that are used as Meta information.

Qualified Dublin Core includes three additional elements (Audience, Provenance and Rights Holder)

These 15 Meta elements if added to a document by the author makes searching of the document more comprehensive. A meta search engine will have more Meta data to decide about the characteristics and properties of the document or content. The search engine would also use the meta data to decide the relative ness or relevance of the search words and the documents being searched and retrieved as results.

Usage of these 15 (+3) elements is optional and the author of the document or content has a choice of not using them or using part of them. The author of the document has the right to associate none, some or all of these elements with his document but if the author does not associate or fulfill these elements then the meta searching has insufficient input about the content. There is no way by which any person other than the author can fill in the data for these Meta elements.

2.2 Remote tagging

Web content is normally written in a markup language like HTML. HTML uses tags such as <h1> and </h1> for the sake of rendering text into headings, paragraphs, lists, hypertext links etc.[05] Web content that loads into a browser environment is already coded into a markup language and the browser simply interprets the markup tags to display the content.

Thus if user annotations (comments or other user content) needs to be inserted into the original content (that is already in marked up format) then it must also be inserted as tags of the markup language (as shown in the figure content mixing) which the browser can interpret and understand and thus display them. These tags (with user content) need to be loaded remotely or locally into the web content. Thus the term remote tagging can be explained as remotely (or locally) adding tags (that contain user annotations) to the content which is loaded (or is currently loading) in the browser environment.

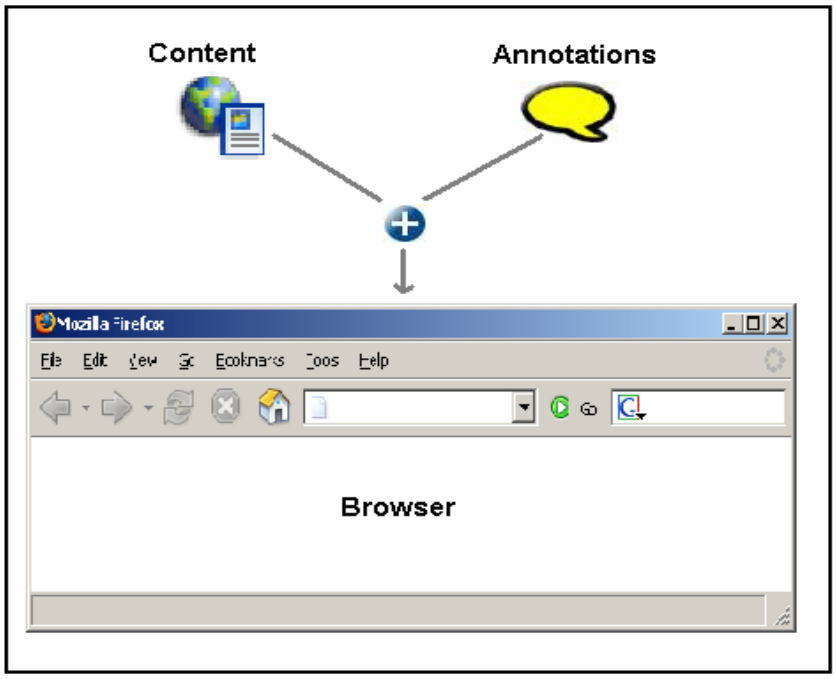

Figure 4 : Content Mixing

2.2.1 Insertion of remote tags (addressing issues, security)

The browser environment understands and displays only the markup language which consists of tags. Tags that are not understood by the browser are skipped and untagged content is dumped as text which is still addressable using the Document Object Model [14]. User annotations must be wrapped into markup tags that are understood by the browser. Problems arising here is to find and use only those tags that are supported and interpreted in the same way by all browsers. Different browsers support and interpret tags in a slightly different way. For example the color codes and their rendering is not uniform within all browsers. Some color codes are rendered differently in different browsers.

Another issue in the insertion of remote tags is addressability of the reference node. The user annotation will be loaded with an reference to the base / anchor / reference tag. Thus before the user annotation is loaded the address of the reference node needs to be found out. Exact addressing difficult but not impossible, leaving the least of ambiguity is the problem to be solved.

2.2.2 Retrieval and redisplay of remote tags

The user annotations if have to be displayed or redisplayed at a later point of time need to be saved in some kind of storage independent of the original content. The scheme of storage needs to be addressed. The methods or retrieval also needs to be worked out. Problems arising here are network issues, local browser security that may prevent local file system access or remote content access.

The redisplay of remote tags need the exact address of the base or reference tag in the web content in the browser environment. Which parameters of the annotation and the base or

reference text or node [anchor node] need to be saved or remembered remains a topic of study and research. The exact addressing of the tags is essential. How this is achieved is an issue that needs to be addressed.

2.3 Remote evaluation

Each user annotation will have the annotation text and will also need to have technical data of the annotation (like node address, annotation id) for the sake of addressing, context, storage, retrieval, insertion and display. The annotation text and the technical information of the annotation can be used to evaluate various parameters remotely. These parameters could be rank of the annotation (that is overall popularity of the content or overall weightage based on various factors), number of times the annotation has been adopted by other users etc. The issue is what effect will these evaluated parameters have on the original content and how these effects need to be handled? That means if the evaluated rank of an annotation is high then it must be rendered in a different way as against a low ranked annotation. Now the issue here is how can these parameters be deduced from the annotation content, context and the technical data of the annotations. The later issues is the usage of these evaluated parameters to implement in the presentation.

2.3.1 Using content of remote tags for remote evaluation.

User annotations have its own text that is entered by the user. This is the content of the annotation. The text entered by the user has certain meaning and relevance to the base / reference/anchor text because it is the basic reason the user has entered this text as an annotation to the base text. The annotation text and base / reference text pair could in a combined way be used to evaluate meaning, context or even display parameters. How this is achieved is an issue which needs to be studied and addressed.

2.3.2 Using technical data of remote tags for remote evaluation

Each annotation has its own technical data for the purpose of insertion, retrieval, storage and display. This technical content could be visible content (annotation id, user who has inserted the annotation) or invisible content (base / reference text address, annotation display parameters).

This technical content can be used for remote evaluation both in the browser context (that is on a reload of the content in the browser) and also in the storage context (on the storage of the annotation). The evaluation yields some parameters like an orphan annotation in case of dynamic content or huge sized annotations yield being displayed as linked annotations. The issue is what effect will these evaluated parameters have on the insertion, storage, retrieval and display of the annotations and how must these effects be handled? Each evaluated parameters have related effects associated to it. Determining and evaluating these effects is an issue to be studied and handled.

2.4 Annotation Presentation and Personalization.

Robert Wilensky in his lecture notes described the following methods for presentation of annotations. [07]

- Separate screen region.

This means that the annotations will be displayed in a separate screen region like a frame.

- Links represented as icons, underlining, etc.

Annotations will be represented by icons which will link to the actual annotations on a remote or local place.

- Previews

When mouse is rolled over the base text a preview of the annotation is shown to the user.

- Superimposition

Option source document modification E.g., double-space original, use copy editor style marks for commentary

The advantages and disadvantages of each method need to be studied in the design section before one method is selected for the presentation of annotations.

2.4.1 Use of remote tags for content personalization

This mixing of remote tags with the original content causes the original content to be altered in a specific way. Like some rendering modifications in case of linked annotations or separate screen regions. The effects caused by the mixing of annotations must be studied. The specific changes implementation is an issue which needs to be addressed.

The presentation of the original content may be altered both by the annotation text and the technical data of the annotations.

Chapter 3 - Related Work

The intention of this section is to reflect on related work by others. This reflection leads to finding the abilities and the disabilities of various related works. Once found the abilities are adopted or retained in the new work and the found disabilities are avoided. Reflection on the related works also leads to some related works that are potential candidates to act as a basis or starting point to start new work or extend the current work done.

While looking for related works four kinds or categories of related works were found. The categories are as follows

1. Independent system approach.

This is the approach where the annotation engine or system is totally independent system. Content that is to be annotated must be loaded in this system and then the user can insert, view, delete, edit or manage his annotations.

2. Server based approach.

A server based approach is one in which the system resides on some server. The user is needed to access this server in order to see, insert, delete, edit or manage his annotations. The annotations are stored on the server here in some kind of storage or database. In this approach the user needs the server that is related to the system and without the help of the server he cannot handle his annotations.

3. Markup Language approach.

A markup language is used for proper presentation purposes of the content. These markup languages can be used to annotate content. In this case the annotations need to be coded in the markup language to be properly displayed in the content. Thus the markup language approach uses markup language to insert and view annotations.

4. Browser based approach.

This approach is one in which the functionality of the browser is extended by creating pieces of code that behave as plugins or toolbars in the browser. The toolbars or plugins are used as front ends by the user to insert annotation or to manage them. The storage of the annotations may be integrated with the original content or be independent from the original content.

3.1 Independent system approach.

3.1.1 CoNote

CoNote[27] is an annotation system developed at Cornell University [25] which is based on HyperNews [26] and written in Perl programming language.

When a reader views a page with annotations, a link to the text of the annotation is displayed in the document along with the author's name, time and date. If there is more than one annotation at that spot, they are listed by date. Those referring to other annotations are indented directly beneath them (similar to a thread). Once an annotation has been selected the user may move through the annotation in the new window via navigation buttons at the bottom.

Annotations are limited to locations selected by the author when creating the document and cannot be placed throughout the page. To add annotations, the reader can use a "Reply" button at the designated place for annotations. If there is a list of annotations already, the "Reply" button is at the end.

Access to annotations is divided into four separate roles defined by a group of people who share these collected documents. The first role allows a viewer to see an annotated document, but not the annotations themselves. The second role, reader, can read the document and the annotations. A user can see, read and add annotations. And finally, an author can read, add and delete annotations where necessary.

Anonymous users are able to access a document if the group defines a default role. This allows outside users to access the document without having to add them to the CoNote group.

CoNote also allows a user to search for specific annotation information by date, author or document. A tree structure is then displayed of the information found by that query.

Advantages

- Annotations are searchable via information linked to them like date or author.
- Supports linked annotations
- Separate roles for users are defined

Disadvantages

- Is a separate annotation system
- Only one kind of annotations is supported.
- Annotations can be placed only at author designated areas and not throughout the content

3.1.2 Melita

Melita[30] is an annotation interface that uses Adaptive Information Extraction (IE) [28] from texts for reducing the burden of text annotation. In Melita, adaptation starts with the definition of a scenario, including a tag set for annotation (possibly organized as ontology) and a corpus of texts to be annotated. Annotations are inserted by first selecting a tag from the ontology and then identifying the text area to annotate with the mouse. Melita actively supports corpus annotation using Amilcare [29], an adaptive Information Extraction tool based on the (LP)2 algorithm.

While users annotate texts, Amilcare runs in the background learning how to reproduce the inserted annotation. Induced rules are silently applied to new texts and their results are compared with the user annotation. When its rules reach a (user-defined) level of accuracy, Melita presents new texts with a preliminary annotation derived by the rule application. In this case users have just to correct mistakes and add missing annotations.

The following diagram shows a screen shot of the Amilcare system. Here the highlighted words are automatic suggestions for annotations offered by the system.

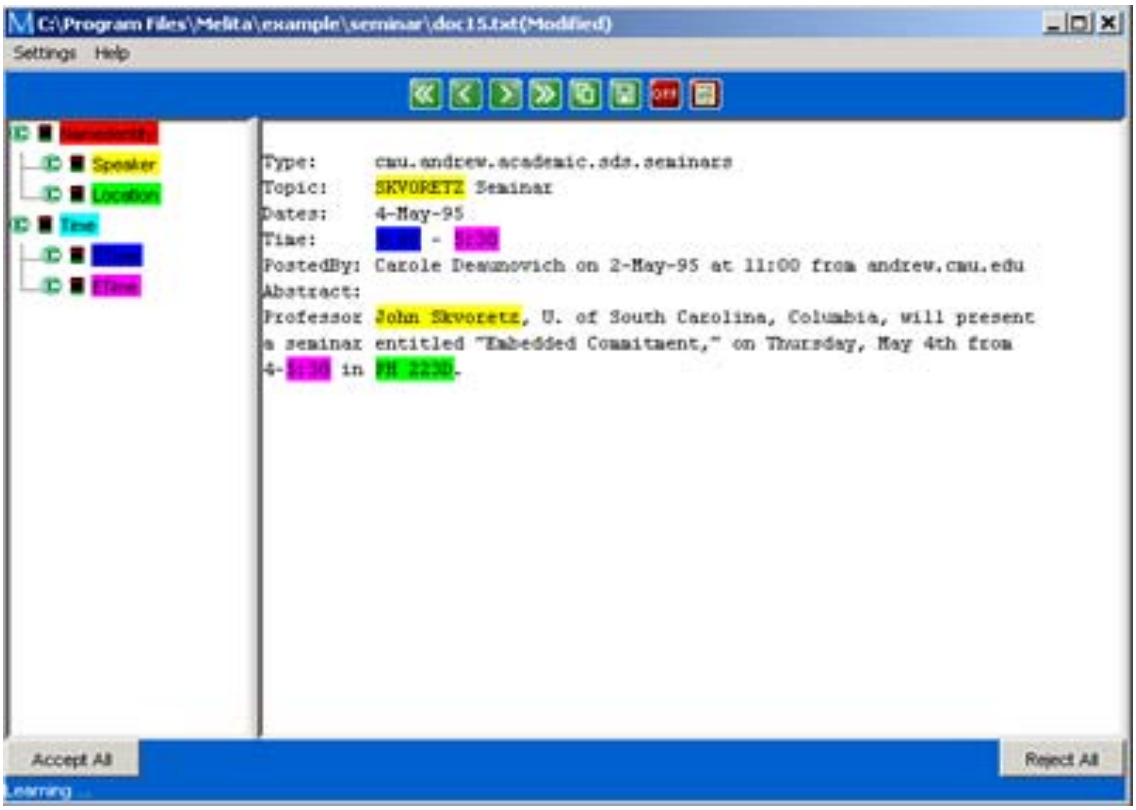

Figure 5 : Melita annotation system

Features

- Makes use of IE (Information extraction) system.
- Speeds up annotation by making suggestions.
- Rules can be customized.
- Abstracts from low level rules.

Advantages

- Allows users to insert annotations
- Is a self learning engine.
- After a number of annotations insertions the engine starts to suggest the insertion of annotations on its own based on the users previous actions.

Disadvantages

- Does not support annotation sharing and does not work in browser environment.

Link for further information

3.2 Server based approach

3.2.1 Wendy Seltzer's annotation engine

The Annotation Engine [31] is a set of Perl scripts and a database that allows readers anywhere to add comments to web pages anywhere else. The comments, URL, and location pointers are stored in the database, enabling the Annotation Engine to add them to a page before it is displayed. The engine does not copy the remote web page itself, only adds its extra information to the HTTP stream when it is called to proxy a page.

The annotation engine uses a webpage that has frames as shown below in figure 9. The right side frame is loaded with the actual content that is to be annotated. This is done by entering the resource link in the given bar below named as "Annotate URL". The left frame has the actual annotation related to the content and tools to insert annotations. User has to select a unique text from the content and then click on "Add A Note" link from the left side frame. The user is given the option to enter some annotation text. The presence of an annotation in the original text is indicated by a small icon and a click on the same leads the user to see his annotation in the left frame.

Features

- Supports annotation insertion
- Supports annotation viewing
- Displays the presence of annotations in the original content via the presence of a icon.

Screenshots

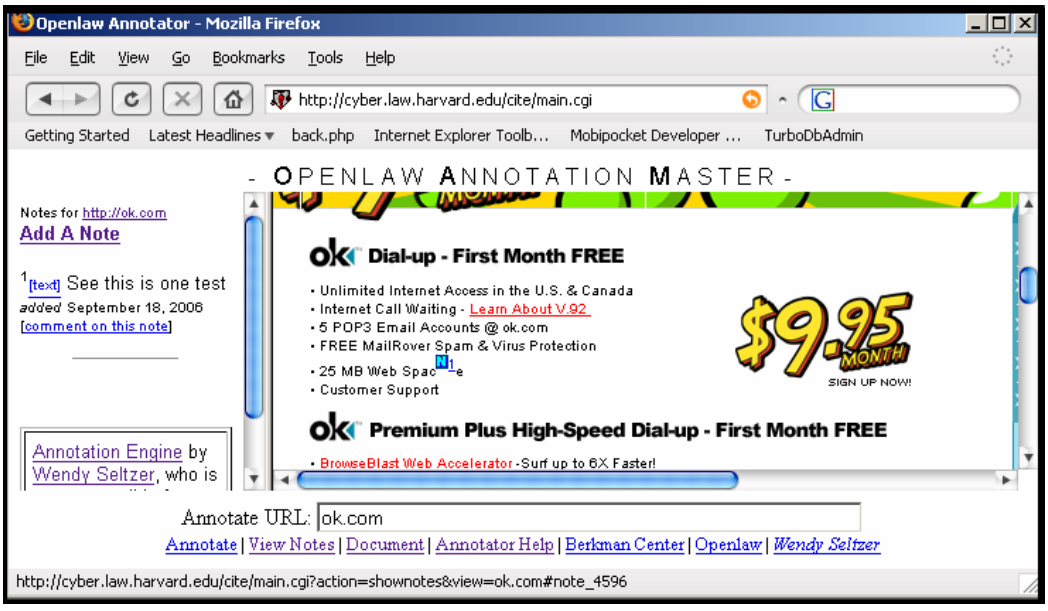

Figure 6 : Wendy Seltzer's annotation engine

Advantages

- Annotations can be inserted and viewed on web content
- Has support for orphan annotations. They are listed at the lower end of the page
- Supports comments to annotations

Disadvantages

- Uses frames and thus annotation insertion and viewing can be done in these frames only.
- Different types of annotations are not supported. Only one kind of annotation that is the icon based linked annotation display is supported.
- Annotations cannot be deleted
- Annotations cannot be shared with other users using the same tool.
- No user accounting. Thus all annotations are open annotations which can be accessed by any user accessing the content.

3.2.2 Connotea

Connotea [32] is a service that allows users to save and organize their bookmarks by the use of associated tags to bookmarks. These bookmarks can then be shared with other users and also can be accessed from any computer. The bookmarks library can be accessed online on the website or can be exported to various formats and then be reused elsewhere.

Thus this proves to be a wonderful tool for book marking website addresses. The tags that are associated with each bookmark can be considered as annotations to the web addresses.

Features

- Quickly saves and organizes links to users references
- Easily shares references with users colleagues
- References can be accessed from any computer
- Save references with just one click
- Easy to use. Nothing to download and nothing to learn. User can start creating his library right away.
- References can be exported to, or imported from, desktop reference managers
- References can be public, private, or shared with a selected group of colleagues
- Save links to anything you find on the web

Advantages

- Supports adding bookmarks on a single click
- Tags can be associated to the bookmarks
- The tags can be used to search stored bookmarks
- Online user account support and a library of bookmarks
- Library of bookmarks can be exported

Disadvantages

- No annotation insertion, viewing, edition and deletion support

Screenshots

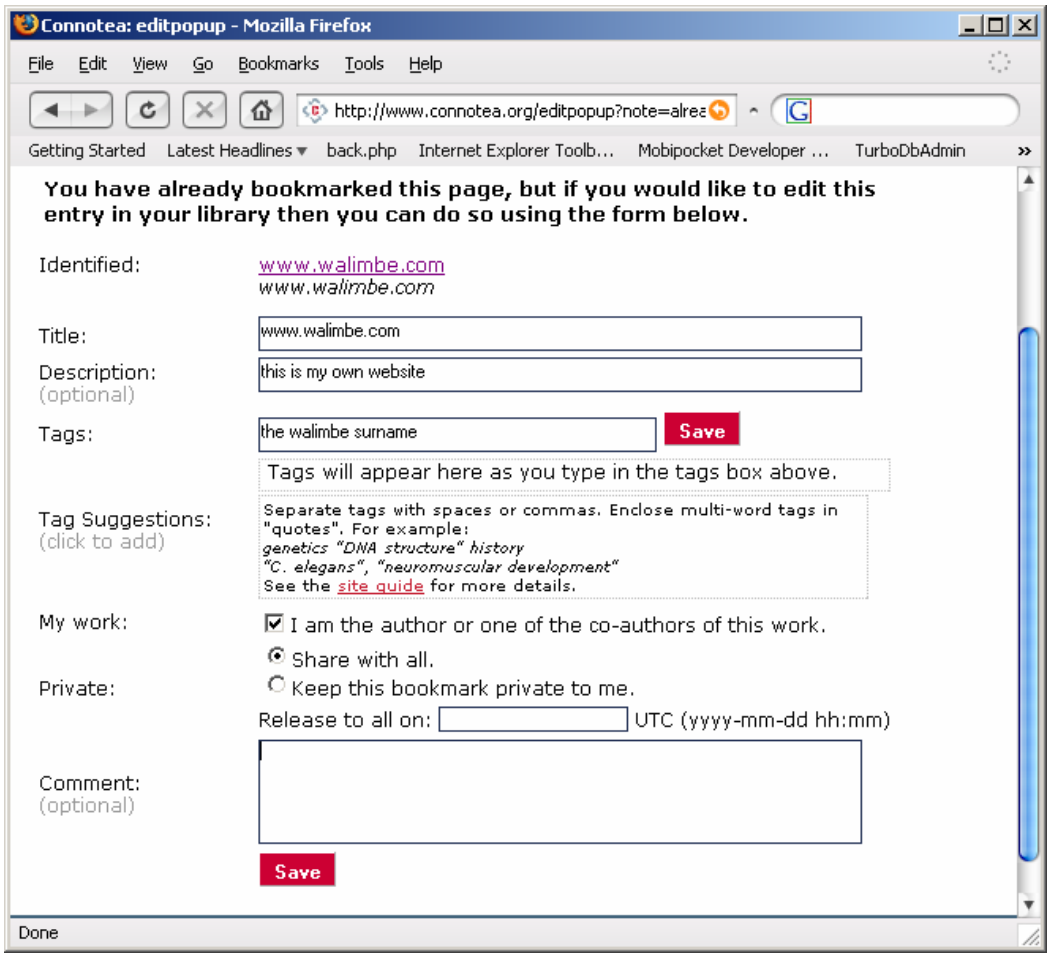

Figure 7 : Connotea bookmark insertion

3.2.3 E-Marked

E-Marked [20] is a application that enables users to insert annotations on web content in a browser based environment. Users need to create a account on the website of E-Marked where they can create new topics (or categories) or use one of the topics created by another users. Each topic is related to a webpage or multiple web pages. Clicking on the topic leads the user to the specific related webpage. This web page is actually loaded through a proxy server located on E-Marked. When this webpage is loaded through the proxy server, this proxy server adds the annotations to the content of the webpage.

Thus finally the user gets to see the original content loaded with the annotations. The base text is marked with yellow background once an annotation is inserted on by the user.

Screen shots

Figure 8,9,10 show the working of the E-Marked annotation engine.

Once the user creates a user account and then creates a category and browses to a webpage then as shown in Figure 8 the “Edit” field pops up at the location once a base text is selected by the user. The user clicks this “Edit” fields to see a pop up appear as shown in figure 9. Here the user enters the annotation text and clicks “save”. This popup disappears and in the background the annotation text is saved in reference to the base text.

Figure 10 shows the coloring of the base text with yellow color and when the mouse is rolled over the text the annotation text appears as a popup on the base text.

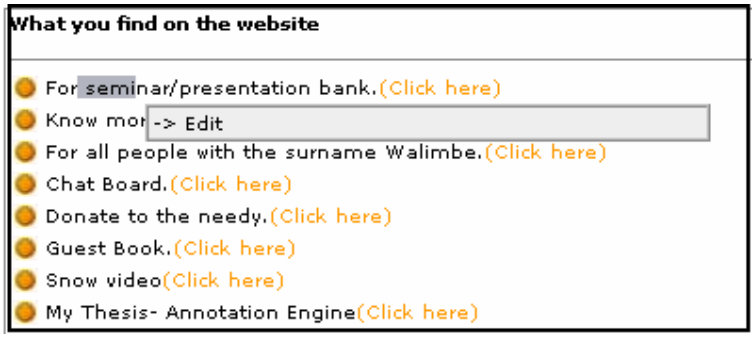

Figure 8 : E-Marked : Select some text and click edit

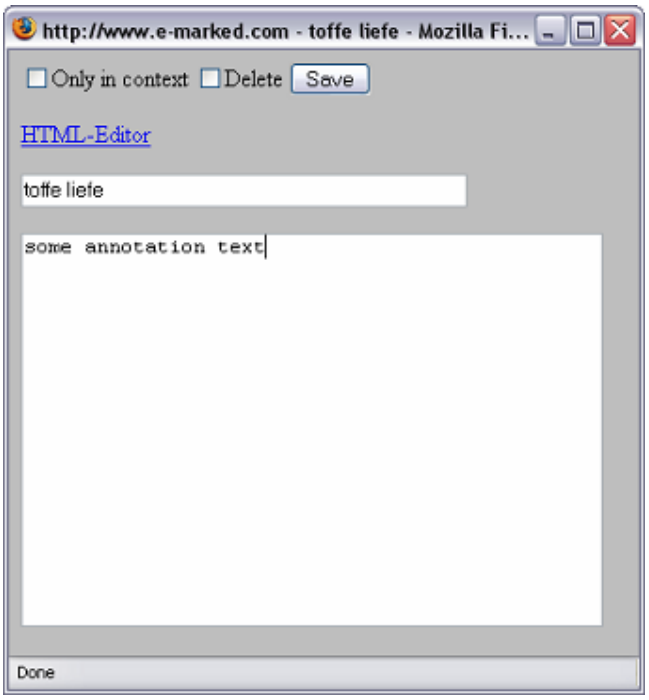

Figure 9 E-Marked : Enter some annotation text

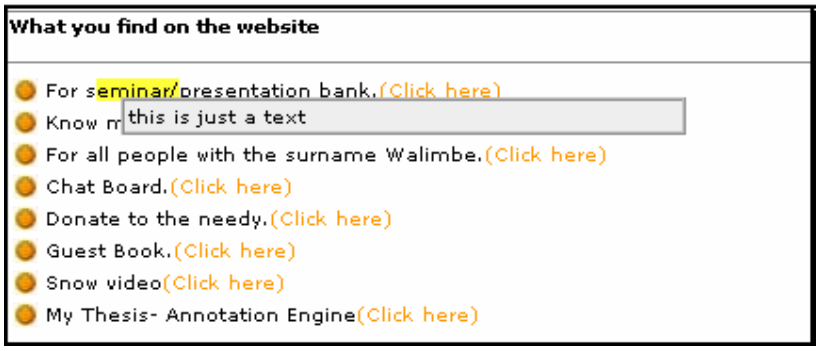

Figure 10 E-Marked : The annotation text appears on mouse roll over.

Advantages

- Works with many different browsers
- Viewing of annotations is easy because annotation appears on a mouse roll over the base text

Disadvantages

This tool does not work with the following cases

- Pages with password- or cookie restricted access
- Pages on your intranet or local directories
- Some pages which use dynamic frames

Also the usability is not very good because to create an annotation a user must login to the account and then create a category then link a url to this category and then browse via the proxy to the web page to insert his annotations.

3.2.4 Marginalia Web Annotation

Marginalia [19] is an open source JavaScript web annotation system that allows users of web applications to highlight text and write margin notes. [19]. Marginalia is a set of JavaScript functions that has functions to insert and view annotations. It has 2 different versions

1. The moodle version
2. Stand alone version

1. The Moodle version

The Moodle [23] version adds annotation to Moodle discussion forums. Moodle is a free open content management system. The JavaScript functions (source codes of Marginalia) need to be installed in the base directory where the Moodle is installed on the server. The JavaScript functions add functionality to the Moodle in terms of allowing the user to insert annotations and then view them.

2. Stand alone version

The stand alone version can be integrated with other web applications. The annotation feature works in Internet Explorer and Firefox. The applications also includes smart copy, which automatically includes citation information when a user copies & pastes text in Firefox. All of these features are integrated with a web application; none require special browser or other client software.

Screen Shot and functioning

In figure 5 it is seen that the standalone version is working on a webpage. The left text is the actual text or original content and the right frame has the annotations. The base or anchor text is marked by the yellow background. When a specific annotated base text is selected using the mouse the corresponding annotation in the right frame is also highlighted. The annotation insertion needs a user to select a base text then click the arrow near the right frame thus a new text box appears in the right frame where the user can insert his annotations. Annotations can be close on request by clicking the close mark of the respective annotation.

Figure 6 shows the functioning of Marginalia with the moodle discussion forums. The moodle source codes need to reside in the base directory of the discussion forums to enable a user to insert annotations.

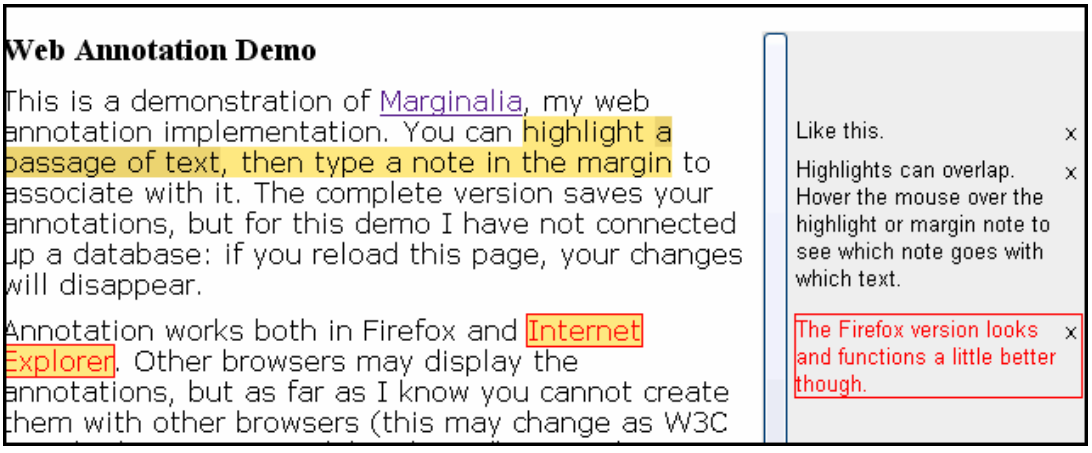

Figure 11 : Functioning of the Marginalia stand alone code with web applications.

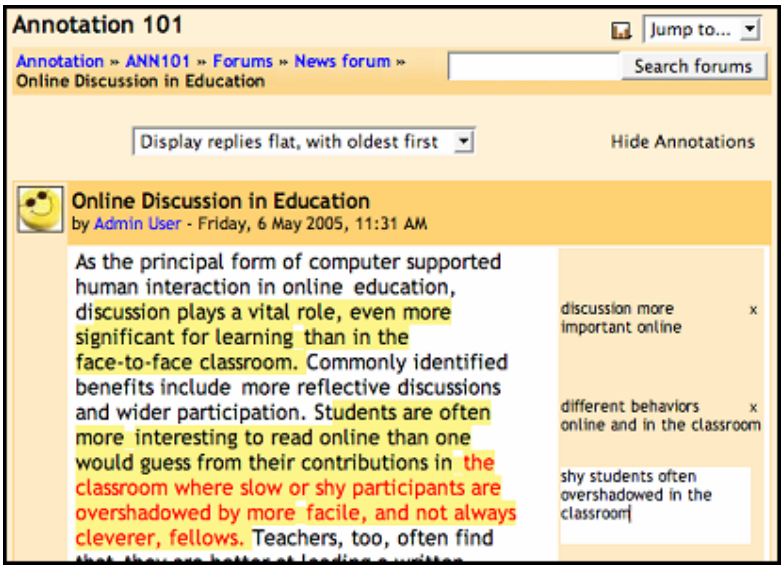

Figure 12 : Functioning of Marginalia with Moodle discussion forums.

Advantages

- Marginalia supports Internet Explorer and Firefox
- Supports annotation insertion, deletion and retrieval.
- Has support for orphan annotations.
- Users can view each others' highlights and annotations directly on forum pages, not just in the searchable summary

Disadvantages

- No orphan annotation support in internet explorer. [24]
- The tool clearly works differently for different browsers.

3.3 Markup Language approach

3.3.1 Ruby

"Ruby" are short runs of text alongside the base text, typically used in East Asian documents to indicate pronunciation or to provide a short annotation [33]. This approach is a markup language which is used in the coding of the html documents and thus does not serve the purpose of decorative annotations.

In ruby there is normally a base text and an annotation text or so called ruby text. The base text is the text that is to be annotated which will have the annotation text linked to it.

The ruby text is the one that is the annotation itself this will appear near the base text

Terms/Tags used in ruby

- rbc – ruby base container
- rtc – ruby text container
- rb – ruby base
- rt – ruby text

Example of the ruby coding

```
<ruby>
  <rbc>
    <rb>10</rb>
    <rb>31</rb>
    <rb>2002</rb>
  </rbc>
  <rtc>
    <rt>Month</rt>
    <rt>Day</rt>
    <rt>Year</rt>
  </rtc>
  <rtc>
    <rt rbspan="3">Expiration Date</rt>
  </rtc>
</ruby>
```

Figure 13 : Sample ruby encoding

The following shows how it is displayed.

Figure 14 : Ruby sample output

Features:

This can be used for Asian languages to give over the text runs or under the text runs.

Advantages:

- This can be used for annotating content. (by using the ruby text to insert the annotation text)
- Very useful for displaying comments

Disadvantages

- Has to be hard coded in the text (could also be inserted by DOM addressing)
- Needs the browser to understand the markup. (Thus is browser specific, currently supported by internet explorer only)

3.3.2 Intellitxt

IntelliTXT is a commercial application. It provides advertisements that are linked to the specific words in the web content. Publishers of the web content must subscribe to this service. IntelliTXT then analyses web pages to find out chosen words that are relevant to the advertisers and places double underlines under the words to indicate the presence of an advertisement there. An advertisement popup appears when a viewer moves his mouse over this word.

IntelliTXT works by analyzing the text on the page, identifying keywords and then matching those keywords against relevant advertisers. The areas of text to be analyzed are marked by surrounding those areas with special tags. When IntelliTXT is applied, the HTML in those areas is replaced with a modified version containing the IntelliTXT keyword. The modified version looks the same except for the addition of the highlighted IntelliTXT keywords.[34]. IntelliTXT works in all browsers.

Features

- Pop up advertisement is possible
- Advertisements are relevant in context to the word.
- Annotations may be inserted in the popup in place of advertisements.

Screenshots

As seen in the figure below the word “poster” has a pop advertisement attached to it on mouse roll over that is relevant to the word “poster”.

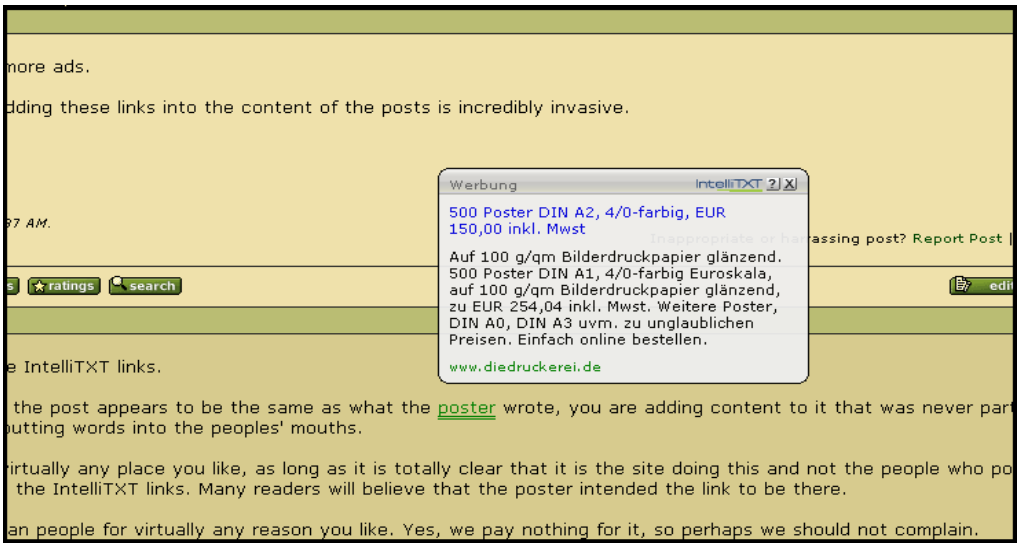

Figure 15 : Intellitxt pop up appearing on mouse over on double underlined word.

Advantages

- Publisher side annotation is achieved
- The display of annotation is the easiest to be viewed. The user needs not perform special actions to view the annotations. User needs to just roll over the mouse over the marked based text to view the annotation.

Disadvantages

- Client side annotation is not possible
- What annotation appears for each word is completely the discretion of the IntelliTXT engine.

3.4 Browser based approach

3.4.1 Annozilla (a client of the Annotea project)

Annotea

Annotea [35] is an RDF [37] standard sponsored by the w3c to enhance document-based collaboration via shared document metadata based on tags, bookmarks, and other annotations.

In this case document metadata includes:

- keywords
- comments
- notes
- explanations
- errors
- corrections

In general, Annotea associates text strings to a web document or selected parts of a web document without actually needing to modify the original document.

Users that access any web documents can also load the metadata associated with it from a selected annotation server (or groups of servers) and see a peer groups comments on the document. Similarly shared metadata tags can be attached to web documents to help in future retrieval.

Annotea is an extensible standard and is designed to work with other W3C standards when possible. For instance, Annotea uses an RDF schema for describing annotations as metadata and XPointer for locating the annotations in the annotated document. Similarly a bookmark schema describes the bookmark and topic metadata.

Annotea basically provides an annotation schema [43] which has the following definitions for a given annotation.

annotates
Defines the resource which has been annotated

author
The name of the person or organization most responsible for creating the Annotation.

body
Relates the resource representing the 'content' of an Annotation to the Annotation resource

context
The context within the resource named in 'annotates' to which the Annotation most directly applies.

created
The date and time on which the Annotation was created.

modified
The date and time on which the Annotation was modified.

related
A relationship between an annotation and additional resources that is less specific than 'body'.

Annozilla

The Annozilla project [35] is designed to view and create annotations associated with a web page, as defined by the W3C Annotea project. It is basically an implementation of the Annotea project. The idea is to store annotations as RDF on a server, using XPointer [38] (or at least XPointer-like constructs) to identify the region of the document being annotated. The intention of Annozilla is to use Mozilla's native facilities to manipulate annotation data. Its built-in RDF handling to parse the annotations, and nsIXmlHttpRequest [39] to submit data when creating annotations.

Features

Supports the following with relation to annotations

- **Posting a new annotation:** a client can publishes a new annotation
- **Querying an annotation server:** a client can send a query to a server and get back annotations
- **Downloading an annotation or its body:** after identifying an annotation, typically as the result of a query, a client retrieves all the data describing that annotation
- **Updating an annotation:** a client can modify an annotation and publish these modifications back to the annotation server
- **Deleting an annotation:** a client can delete an annotation from the server

Screen shot

The screenshots below shows how the extensions window looks like when the 3 extension are installed in firefox browser. The three extensions are the xpointerlib, Annozilla and annotations. (These three extension are part of the Annozilla system)

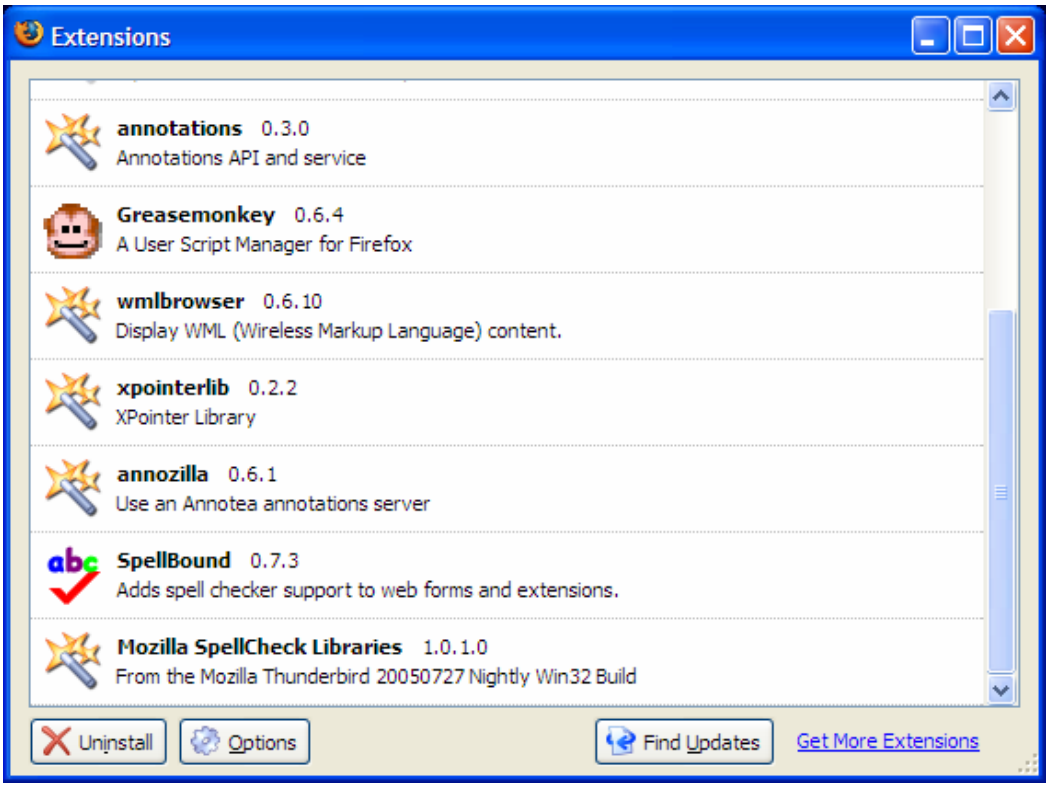

Figure 16 : extension in firefox

Once the 3 extensions are installed then the firefox browser needs to be restarted for the changes to take place. Once the browser is restarted the user is free to insert his annotations. The user just selects some text with the mouse and right clicks to select “Annotate current selection”.

When this is clicked a window popup appears where the user can insert the annotation text and some other properties like server address, type of the annotation and language. This popup window is shown in figure 17

Figure 18 shows further possibilities of Annozilla like spell checking when other extensions like the spellbound are installed.

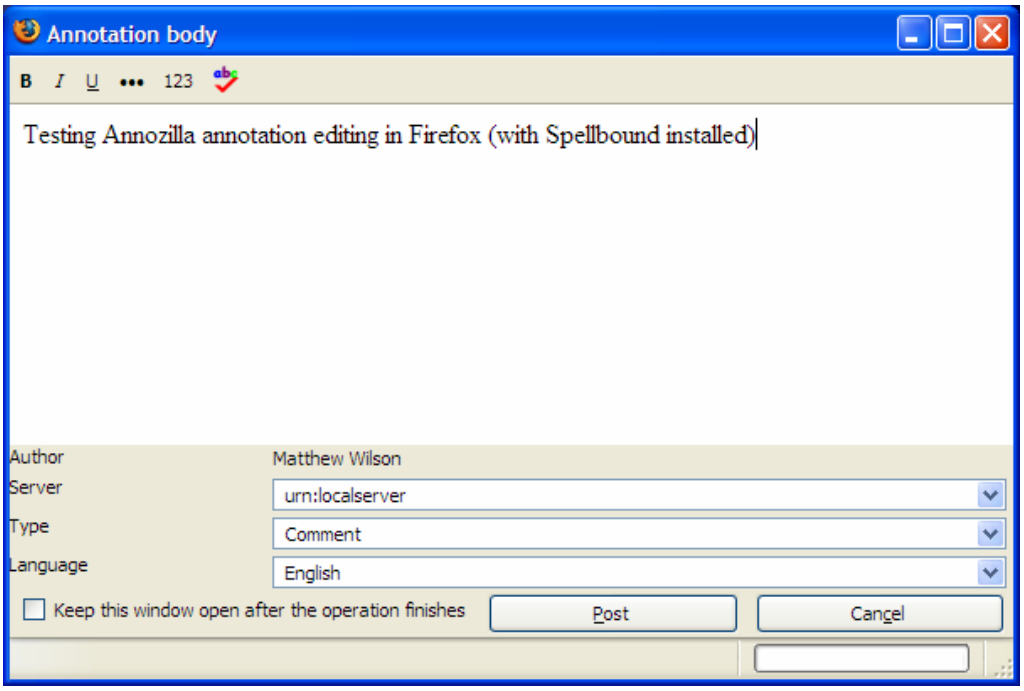

Figure 17 : entering an annotation

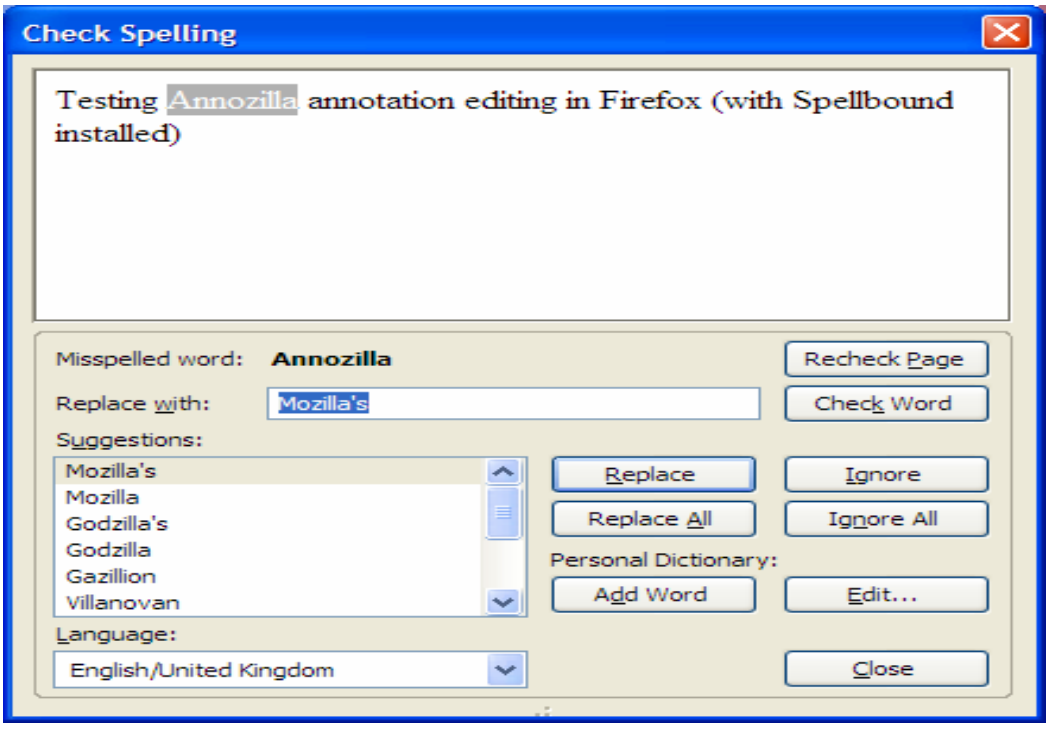

Figure 18 : editing the annotation

Advantages:

- Can insert annotations
- Is RDF based
- Has support for distributed servers
- Has support for different types of annotations
- Uses a Xpath [40] library
- Is browser plug-in
- Has Multilanguage support

Disadvantages:

- The annotation schema has no support for user accounting, annotation display properties, annotation type properties, no user and annotation rights management.
- No annotation sharing is possible
- No web interface to see or manage annotations
- Display of annotations consists of only a single type where a small icon appears near the base text that has been annotated.
- No searchabiltiy of annotations
- No statistics display and use preferences in the plug-in of the browser.
- As servers are distributed so no social networking (where users can connect or communicate to each other on a common platform and thus share their annotations etc.) is possible.
- No group sharing of annotations. (Sharing of annotations to a closed group is not possible)
- No categorization.
- No 3rd party support

3.4.2 The Thirdvoice

Third Voice [44] was a commercial product for annotating World-Wide Web documents, which was freely available from their Web site (previously www.thethirdvoice.com). In order to use this software, a reader had to download the application and register with Third Voice. Each time the reader wanted to browse the Web, he/she was required to log in at Third Voice. A small menu-like icon was then displayed in the top left corner of the browser window throughout the reader's Web session.

When this software was running and the user was logged in, existing annotations were marked with a small arrow in the document text. The reader could click on the arrow or hold the mouse over it to view the annotation.

The reader could also choose to have a list of annotations displayed in the left margin of the page (in a frame). This allowed the reader to click on any of the annotation titles in the margin, which displayed the annotation in the document part of the window. The annotations were not searchable, though.

To add annotations, the reader highlighted the text to be annotated and chooses "Post" from the menu. Then the author chooses one of three types of annotations: private, group, and public (viewed by anyone registered and logged in to Third Voice). The author then entered the text of the annotation and the title in a dialog box.

This software required Internet Explorer 4.0 running on Windows 95 or 98.

Advantage

- Users could insert annotations
- Users could retrieve and view annotations
- Annotation management was possible

Disadvantages

- Annotations could not be searched
- The service was discontinued

3.4.3 YAWAS - Yet Another Web Annotation System

Yawas is a very simple and lightweight program that lets users highlight web pages. All functions are accessible through the context menu in Internet Explorer. [41]

Functionality

- Yawas can post the annotations to a server, allowing the user to access them anytime, anywhere.
- Remote access to Yawas database: in Yawas.ini, user can specify the location of the database.
- Highlight any web page: simply select some text and select "Yawas:Highlight" from the context menu
- User can search annotations
- Highlight the annotations.
- Find all words on a page and quick jump from the list of words to the document as in Yawas:View
- Import/export user annotations

Screenshots

The following figure shows how a user selects some text and right clicks his mouse then a popup appears which gives him an option to annotate via “YAWAS:Annotate”

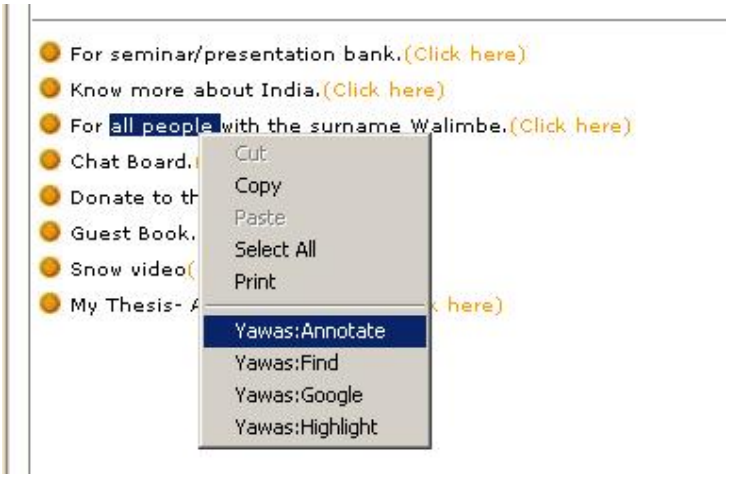

Figure 19 : Yawas - select text and right click

Once the user clicks “YAWAS:Annotate” the following popup appears where user is allowed to insert his annotation text and other annotation properties like gerne or theme of the annotation.

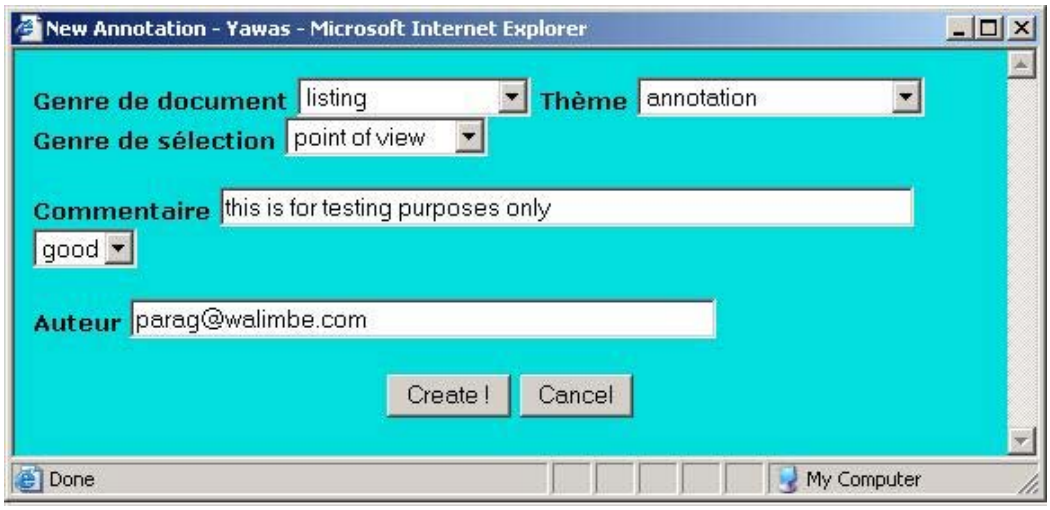

Figure 20 : Yawas – inserting the parameters of the annotations

Advantages:

- Annotations can be inserted and viewed.
- Has highlight and annotate functions.
- Has different classifications for the annotations (like theme of the document, gerne of the document, gerne of the selection, comment type)

Disadvantages:

- Annotations are not shown directly, user has to click the highlighted text right click and select view to see the annotation
- Needs an entry to be made in the registry of windows. Which is not automatically removed on an uninstall of the toolbar.
- Highlighting display properties are not changeable (only yellow color)
- Annotation deletion is not supported.

3.4.4 iMarkup

The iMarkup Plug-in is an ActiveX control that runs in Internet Explorer (IE) browser [21]. It provides features for inserting, retrieving and displaying annotations.

Three major types of annotation can be inserted using this toolbar.

- Sticky Notes can be attached to a specific screen region.
- Paint Brush which can be used to mark screen regions.
- Text Markups

Paint brush has the following options

- Brush color can be changed.
- Brush width can be adjusted.
- Paint brush
- Straight brush

The text markups have the following options

- Bold
- Highlight
- Italics
- Strikethrough
- Underline

The tool bar is shown in the figure below. This toolbar only works when it is properly connected to the server which is a commercial application and has to be purchased.

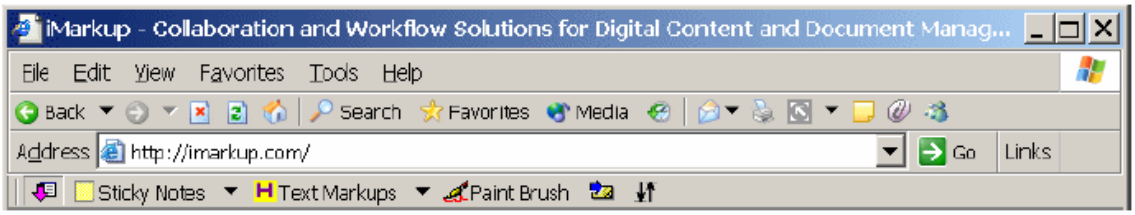

Figure 21 : View of the IMarkup toolbar in Internet Explorer.

Advantages:

- Has multiple types of methods to display annotations
- Allows annotation sharing
- Annotations are searchable
- Annotations can be emailed to other users.

Disadvantages:

- It is a commercial solution and thus costs money.
- Uses geometric positioning to insert annotations thus makes it less fault tolerant. (in case of a screen resize)
- Geometric positioning makes it fail in cases of dynamic content.

3.4.5 Web Annotator

The web annotator is a toolbar made for the internet explorer. It is ported as an installer. The user of this toolbar can insert annotations in the document or web content. Annotations allow deleting and adding both text and images. Fonts can be changed, and justification and formatting can be applied. There are buttons for undo/redo, cut, copy, paste, etc. The Web Annotator requires IE 5.5 or higher to run. [22]

The program is fully functional as a standalone product, as the ability to share filters through a server is not required. The server database uses mySQL, a free database product

The figure below shows this toolbar installed in the internet explorer browser.

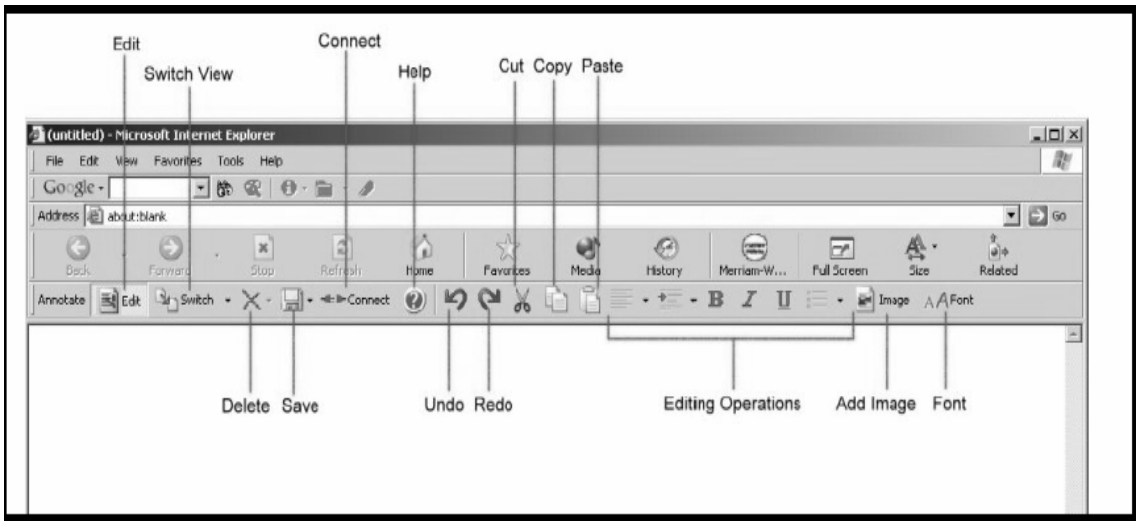

Figure 22 : The web annotator toolbar in internet explorer.

Advantages:

- Has multiple types of methods to display annotations
- Has kind of a document lock because documents cannot be edited till the edit button is clicked.
- Has undo and redo buttons which makes the task of editing changes easier.
- The toolbar has the switch button which switches the view of the document one with the annotations and other without the annotations.

Disadvantages:

- Uses geometric positioning that makes it less tolerant to dynamic content.
- Has no support for other operating systems.
- Has no support for other browsers like Netscape.

3.5 Evaluation of related works.

The goal of the title of the thesis is to make an annotation engine that can remotely tag, store, retrieve and display user annotations in a browser based environment. For this either a new system can be made altogether of one of the above discussed systems (from the related works) can be used as a base or a starting point and its functionality can be extended to achieve the final goal. Here let's first discuss about choosing a possible candidate solution to accept as a base to start the work at.

As seen from the advantages and disadvantages of each of the related works mentioned above one can argue that if a remote tagging, storage, retrieval and display system is to be made for an browser environment then independent systems like CoNote or Melita do not contribute a lot towards the goal. If one thinks to use them as a base for starting the work and extending their functionality it does not lead him a lot towards the goal because he does not end up making a system for a browser environment. Though some good features can be adopted from these independent systems. Feature like suggestive annotation or a learning engine as used in Melita. Eventually one can say that independent systems cannot be the basis to start work.

Commercial solutions like IMarkup or Intellitxt also cannot be a good starting point for the work neither can dead systems like the third voice be used because their codes are not available. Thus surely only the solutions that are open source like Annozilla, Yawas, Webannotation must be considered as a possible starting point or basis for making the needed system.

As we have seen with the markup language solutions like RUBY need to be hard coded in the web content for the annotations to be displayed. This does not satisfy the need for the user to insert annotations and further retrieve them. RUBY only satisfies the need to display annotations when they are hard coded into the web content. Moreover RUBY must be supported by the browser for it to be considered a solution. Thus considering all these elements RUBY or overall a markup language approach is not a good solution to start work at.

Server based solutions like Wendy Seltzer's annotation engine or EMaked or Marginilia could be possible candidates to start work at but the disadvantages of each must be considered. With Wendy Seltzer's annotation engine the annotations are rendered in a separate screen region by the use of frames thus reducing the effective usable screen region of the user. Also the webcontent must be first loaded in the given screen area for the user to insert annotations. It also has the following drawbacks of not having any user accounting and it lacks the possibility to delete annotations. Thus it is not a very user friendly approach. Similar is the case with EMarked where user has to first go to the website, create a account, create a category, add the desired url to the category, then browse the url via the proxy embedded in EMarked and then the user can insert or see annotations. Marginilia also uses separate screen regions and has to be combined with other systems to be used effectively. Thus the server based approaches do not prove to be good candidates as a base system.

Solutions like the Webannotation and iMarkup use graphical positioning to place annotations. This makes the annotations less tolerant to dynamic content. Graphical positioning also fails in case of content or browser resizing. Thus a robust content addressing scheme must be used like the one used by Annozilla. Dynamic content creates annotations whose base or anchor text cannot be found. These annotations are called orphan annotations. This also leads to the conclusion that an intended system must have

support for orphan annotations. Systems like Marginilia do not have support for orphan annotations.

User accounting plays an essential role in tracking junk or foul languaged annotatations or other malpractices. Systems like Wendy Seltzer's annotation engine, Marginilia, YAWAS, Annozilla have no user accounting.

Some systems like YAWAS do not support annotation editing or annotation management. Annotation deletion is not supported by Wendy Seltzer's annotation engine. Thus user has no opportunity to make changes to his own annotations.

Multiplatform and multi browser support is one essential factor that is missing in solutions like YAWAS or Annozilla. A user must be able to use the annotation engine on his chosen platform or his chosen browser.

From all the discussion above it is seen that the solution Annozilla has the least negative aspects. It also supports open standards. Annozilla proves to be a strong candidate to act as a base system for further work but the following points work against the favor of Annozilla in being selected as a starting point of the work.

- Annozilla is only made for the Firefox and Netscape browsers because it is made by the mozilla foundations that are responsible for the development of Firefox browser.

- The current client Annozilla is only compatible with the given server part in terms of storage of annotations. Adding new functionality to the system means change in the server or storage part to accommodate new parameters. This is not supported by Annozilla.

- Annotations are totally shared thus everybody has access to everybody's annotations. Thus is a user opts to load all shared annotation in the browser content then this clutters up the users screen with annotation when the user loads a heavily annotated content.

- Annozilla uses Xpointer and W3C Xpointer specification does not provide suitable methods to specify character offset [reference 24 page 12]. The character offset is essential for robust and precise addressing of content.

3.6 Basis of the work.

The discussion from the above section derives some conclusions or guidelines for the basis of the work that has been carried out in this thesis. The guidelines can be summarized as follows.

The new system

- Must not be an independent system but must depend on the browser environment.

- Should not be a commercial system but an open source system.

- Should not be based on a markup language approach

- Should not consist of screen regions that narrow down the users perspective
- Must have user accounting to avoid malpractices.
- Must be user friendly
- Must have support for manipulation of user annotations (edition or deletion)
- Should not use graphical positioning but must use robust addressing schemes.
- Must work with different browsers on different operating platforms.
- Must have multiple types of annotations and multiple methods to display the annotations.

Chapter 4 – System Design

4.1 Design elements.

In this part the design elements are discussed. By the term “Design Elements” it is meant the various elements that are needed to make the whole system.

The code is intended to work in a browser environment thus for the purposes of implementation, one browser must be chosen, the programming behind must be however browser independent so the same code can run on different browsers. The annotations inserted by the user will be inserted though a user interface which needs to be a user friendly interface. The annotation display approach is one element that is of much concern because it will decide the overall experience of the user and thus the later usability of the whole system. The annotations must be stored in order to be retrieved and displayed thus a proper storage and transport methodology must be designed.

This leads to the following design elements whose decisions must be taken

- Browser
- User interface approach
- Annotation display approach.
- Storage type.
- Expected Functionality

The actual decisions on the selections of each of these elements are discussed in the next section.

4.2 Design decisions

4.2.1 Browser

One browser must be chosen for the sake of implementation of the thesis. Such a browser is to be chosen that would guide the development process in the implementation phase. I have chosen firefox browser for the sample implementation of the thesis.

Why firefox browser?

- This is an open-source code and has a version for almost all operating systems.
- This browser runs independent of the operating system so a flaw in the browser software won't necessarily expose the entire computer.
- Supports xul (xml user interface language)
- Supports toolbars to be written in xul and thus the browser functionality can be extended

4.2.2 User interface approach

Various approaches for a user interface are possible like the side pane window, a server based approach, independent browser / software approach or a toolbar approach. The choice of a toolbar approach is done for the sample implementation of the thesis. The reason for choosing the toolbar approach is discussed by discussing the various approaches.

A side pane window in browser

A side pane window is one window that appears on the left hand side of the browser window like a frame. It can be used to display annotations and can be used to insert, edit, modify or delete annotations. The problem with this approach is that it reduces the view area of the user by reducing the actual display area of the browser. The user will always have less space in his browser window for the actual web content. Thus though this is an approach that can be used but because it reduces the view area of the browser it seems not to be an apt approach for the annotation engine.

A server based approach for side panes

A server based approach like the one used by Wendy Seltzer in her annotation engine can be used. This approach has a webpage with frames that the user must load first in his browser window. This satisfies the needs of insertion, edition, modification and deletion of annotation in the web content but the issue is that again it uses the frames method which eventually reduces the view point of the user for the actual web content that is loaded in the browser. Moreover the annotations are loaded only when the specific web page is loaded in the assigned frame on the assigned page on the given server. It is not a generalized approach and thus needs the user of the annotation engine to go to the server or webpage to insert, view, modify or delete his annotations. Thus this approach also does not satisfy all the needs of the annotation engine.

Independent browser / software approach

An independent browser or special software could be made to insert, view, modify or delete the annotations. The limitation with this kind of approach is the same that is with the server based approach. The user of the annotation engine must always use the specific software or the independent browser to access the web pages or web content that he or she intends to insert, edit, modify or delete annotations to. Moreover the approach of making an independent browser for an annotation engine is not feasible as it requires a lot more efforts in the design and compatibility of the browser features rather than the actual annotation engine itself.

A toolbar approach

A toolbar approach leads to making a toolbar for a browser. This toolbar needs to be installed in the browser as a plugin or toolbar. The toolbar or plugin can then be used as a user interface for the whole annotation engine.

Advantages posed by this approach are as follows

- As most browsers support functionality extension by the use of 3rd party software thus the task of making an annotation engine toolbar is somewhat simplified.
- The toolbar or plug-in can make use of the native functions and methods of the browser (like alert etc).
- Security issues and access rights can be addressed by the browser. The design or code of the annotation engine need not take care of it.
- The installation of the toolbar or plug-in is easier for the user and also the un installation has a standard procedure.
- The view, look and feel of the toolbar can be adjusted so to blend with the look and feel of the browser itself. No extra coding in terms of look and feel need to be done.

Thus the toolbar approach is chosen for the implementation of the annotation engine.

4.2.3 Storage type

Local storage (storage on the client's local machine)

A local storage on the client's local machine could be helpful in the sense of faster access to the annotations and thus faster execution of the annotation engine and the browser in general. The issue that leads to not using this approach is that if the annotations are stored locally then the user does not have access to the shared annotations of other users nor does he have the chance to share his annotations to other user or his groups of friend with whom he likes to collaborate on the document. The further issue is of security. If the annotation engine is given access to the local storage of the user then in case of security vulnerability in the annotation engine the local computer of the client is at risk.

A local storage is also a bit more difficult to implement because of access rights posed by the browsers and operating systems security policies.

Server storage

Server based storage allows the user to access his annotations from anywhere. Moreover he can share his annotations with other users of the annotation engine or his own group of colleagues, with whom he wishes to collaborate on the document. There won't be any issue of security because the annotation engine accesses the server with the assigned username and passwords. The only disadvantage here is that in case of a slow connection speed the annotation engine is slower in performance and thus the overall performance of the browser drops down.

A server storage is a bit more simple to implement because of readily available databases that are easy to manage and the insertion and retrieval methods are easier as compared to the local file storage. Thus for the sample implementation of the thesis I have chosen a server based approach. This approach can be further extended to a local storage approach with proper synchronization to the server from time to time.

I have chosen a centralized server based storage.

Why centralized server?

One of the features that the annotation engine must offer is the ability to share annotations with other users of the network and also with a group of colleagues with whom the user wishes to collaborate on a document. Thus all annotations relating to one document need to be stored on one centralized server so that they will be accessible at all times from all places to all users who are authorized to access them. Later for the need of speed or security the annotations could be spread across multiple servers.

All these features can also be achieved by using decentralized servers and proper mirroring or reflecting capabilities but that is a bit difficult step and thus is not possible to be implemented in the give time frame of the thesis.

Why not peer to peer approach?

This approach is not chosen because it is more difficult to implement as compared to the centralized server approach. This is an approach that could be implemented in the future.

4.2.4 Annotation display approach.

The annotations that are inserted by the user need to be displayed to him in some way. The following methods could be candidates for the selection of a presentation method for the annotations.

- Separate screen region.

  The annotations will appear in a separate screen region. This again causes the problems of reducing the view area of the user.

- Links represented as icons, underlining, etc.

  The annotations could be inserted inside the content by use of icons or underlines to indicate the presence of an annotation at that place. The icons or underlines could be

links to the actual annotations that would reside at a location different from the actual content. The click on the icon or underlines leads the user to see the annotations in a different window.

- Previews and Superimposition

This approach is one in which the annotations would be superimposed on the actual content or the anchor text. This might hide the actual content and thus will defeat the very purpose of the annotation been inserted in reference to the anchor text.

- Mouse over Popup.

This is the approach in which the annotation will be loaded in a popup which is hidden to the viewer. When the user rolls his mouse over the anchor text the popup is made visible.

In the mouse over popup approach the base text is colored in some color that is selected by the user and thus is identifiable from the other surrounding text.

This Mouse over Popup is the approach that is chosen because it seems to be the easiest for the user to view his annotations nor does it block the anchor text content in any way. Other approaches could also be implemented later in case the users would ask for them.

4.3.5 Expected Functionality

When a annotation engine is made the user would atleast expect the following functionalities

1. Ability to insert annotation on web content

2. Ability to save and retrieve the inserted annotations

3. Ability to display inserted or retrieved annotations

4. Ability to manage annotations (edit and delete annotations)

Optional functionality would consist of

1. Setting various display parameters of the annotation.

2. Sharing annotations with other users.

3. Annotation accounting

4. Inviting other users to use the user’s annotations.

4.3 System Architecture

The figure 23 shows the system architecture. The main components of the system architecture are

- Server
- Toolbar

4.3.1 Server

This is the component of the system that is responsible for the storage needs. It also houses the codes or the application programmers interface (API) to communicate with the toolbar or with third party software clients which like to use the server. The server also houses the front end for user management and other user interfaces where the user can manage his account data as well as his annotations.

4.3.2 Toolbar

This component of the system deals with the insertion, retrieval and display of the annotations. The front end acts as a user interface, which gives various functionalities to the user for annotation insertion and manipulation. The backend of the toolbar acts as a communication tool that communicates with the server for the transmission of the data for insertion in the database or for the retrieval of the annotations form the server to be displayed in the browser window.

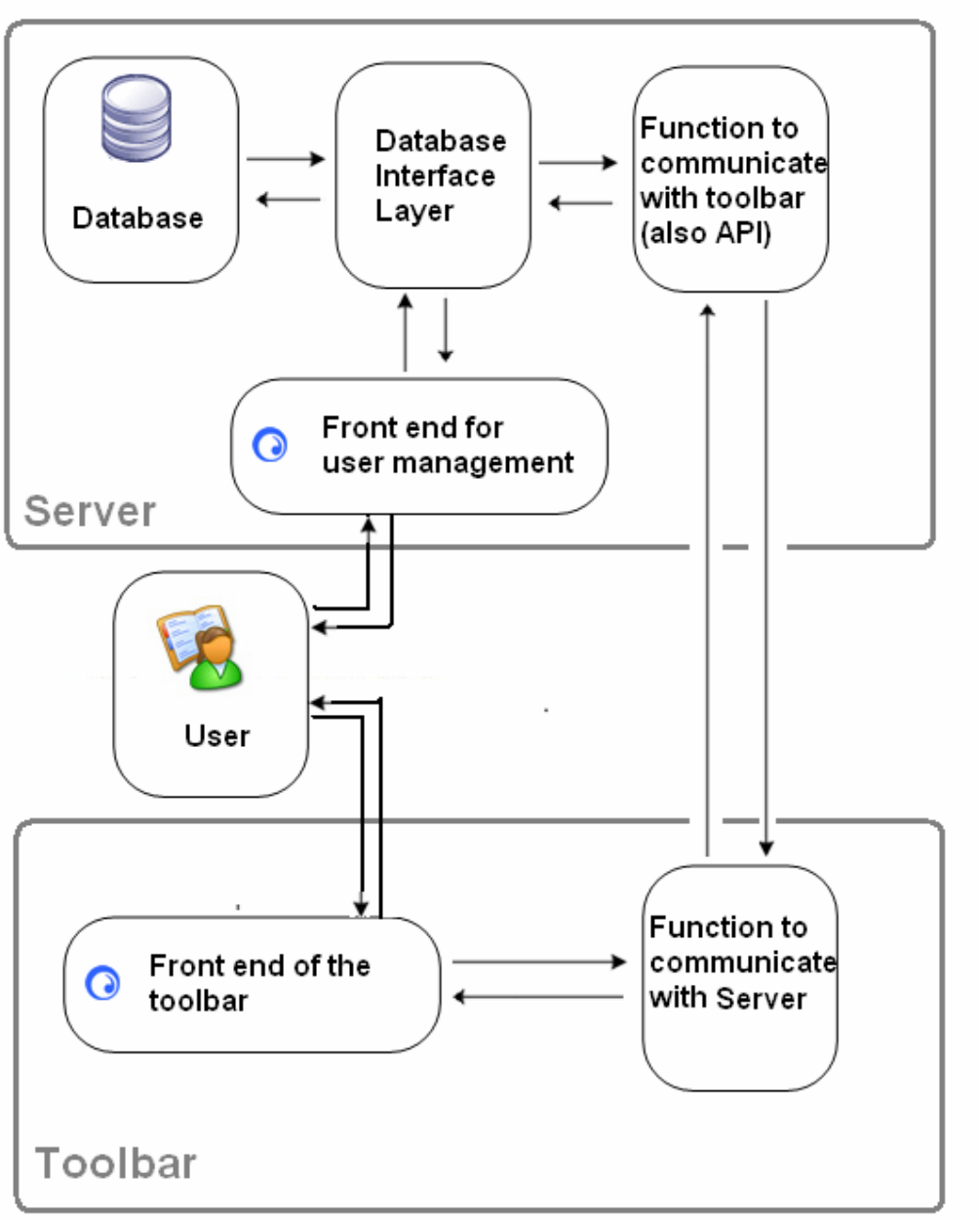

Figure 23 : System Architecture

4.4 Server

4.4.1 Backend – Database design

Data stored and retrieved from tables on the server are as follows

Annotations table

The annotation data structure is one which holds all the parameters of an annotation. The parameters include the annotation text, base text, base text addressing, annotation display properties, access rights, book keeping properties. Based on the need to cater to all these properties the annotation table has the following columns.

- **user –** This is the user name of the registered user who has inserted the annotation.
- **Id –** Annotation identifier.
- **text1 –** Actual annotation text
- **anchoroffset** – The offset of the start of the selected text within the anchor node. The anchor node is the node where the selected text lies.
- **pname** – Name of the parent node. (example P is the name of the node for a paragraph node) The parent node is the node which is the parent of the anchor node.
- **ptype** – Type of the parent node. This is a number. Each type of node has a number assigned to it in the Document Object model. (example : text node has a type number 3)
- **pvalue** – value of the parent node. This is the actual content of the parent node.
- **atype** – type of the anchor node. This is the type of the anchor node. (Example : text node has a number 3)
- **anchornodevalue** – value of the anchor node. This is the actual content of the anchor node.
- **selectedtext** – the actual text of the selected text. This is the base text on which the annotation will appear.
- **website** – the web address on whose text the annotation has been inserted
- **type** – This value will have the type of the annotations. The types currently are private and shared annotations.
- **popupid** – popup id will be used to refer the annotation once it is loaded in the browser environment
- **acolor** – The base text must be displayed in a different color than the surrounding text to indicate the presence of an annotation there. Thus this is the color of the anchor text that is selected by the user.
- **anntype** – This is the type of the annotation, 2 means bubble annotation 1 means inline annotation
- **owner** – this is the username of the user to whom the annotation actually belongs to in case of an adoption this field will tell who is the original owner of the annotation
- **date** – the date on which the annotation was inserted
- **time** – the time on which the annotation was inserted

Invitation table

For the sake of accountability of the annotations the annotation engine can be used only by registered users. In order for people to join the network they must be invited by users who

are already registered users of the network. This table will store all the invitations to join the doodle network sent out by existing users of the network to their friends.

Fields in the invitation table are

- **Id -** This field is the primary key of the table.
- **From -** this field has the information about the user who has sent the annotation.
- **to_name** - this field has information about the name of the person to whom the annotation is sent to.
- **to -** this field contains the email address of the person to whom the invitation is sent to
- **link –** this field has the registration link that is sent to the person in the invitation email
- **code –** this field has the code that is linked to the specific image in the images table. The image from the images table will be used for visual verification from the user in order to avoid bulk and junk registrations
- **active –** this field holds a value which tells the server if the invitation is active or not. Only active invitations are allowed to register.

Table `stats`

This table will store the statistics which can be used for remote evaluations

- **no_user –** this field has the count of the number of users that have registered on the server
- **no_doodles –** this field has the count of the annotations inserted by all the users
- **last_user –** this field holds the username of the user who has registered the latest
- **last_doodle –** this field holds information about the last inserted annotation
- **max_user –** this field holds the number of maximum users that will be allowed to register
- **max_doodle –** this field holds the maximum number of annotations that will be allowed to be inserted in the server database.

As discussed above we have the following fields for the annotatetext table

User table

This table holds all the information entered by the user at the time of registration. It also has extra field for the sake of accounting (number of inserted annotations etc.)

- **id –** this field is the primary key of the user table
- **username –** this field holds the username of the user with which he has registered
- **password –** this field holds the password of the user
- **rusername –** this field holds the username of the user who has invited the user to register
- **email –** this holds the email address of the user
- **dob –** this field has the date of birth of the user that the user has entered while the process of registration
- **name – this** is the name of the user

- **surname** – this is the surname of the user. The name and the surname will be used to address the user in further information emails.
- **type –** this is the type of the user account, this tells if the user is a normal user or is he a administrator
- **invites –** this holds the count of the number of invitations that the user can send out to his friends
- **secretquestion –** this is the secret question which will be asked in case the user forgets his password
- **secretanswer –** this is the answer to the above said secretquestion
- **friends –** this is a list of users that have joined the by accepting his invitations
- **backupemail** – this is a email address which can be used as a secondary email address in case of the primary email address does not work
- **city –** this is the value entered by the user in the process of registration this will be used to identify the user by his city.
- **country –** this field holds the value of the country that the user enters while registration.
- **language1 –** this is the spoken language of the user. Later version would be Multilanguage version
- **no_doodles – this** is the number of annotation inserted by the user
- **no_shared_doodles –** this hold the count of the number of annotations that are shared by the user
- **no_private_doodles –** this is the count of the annotations that are marked private by the user

Images table

The contents of this table are used in the registration process as a visual verification taken from the user in terms of an image.

- **number** – this is the value of the code that is sent out in the invitations
- **value** – this is the actual image that will be shown for visual verification

Logged users table

This table will store information of users that have logged in. This table will have no more use once sessions are implemented.

- **username** – username of the logged user
- **userhash** – hash value of the user name

Following diagram shows how the various tables are related and used

- Square boxes are the table entities

- The diamonds are the relations between the table entities

- The ellipses are the field of the tables that are actively involved in the particular relation.

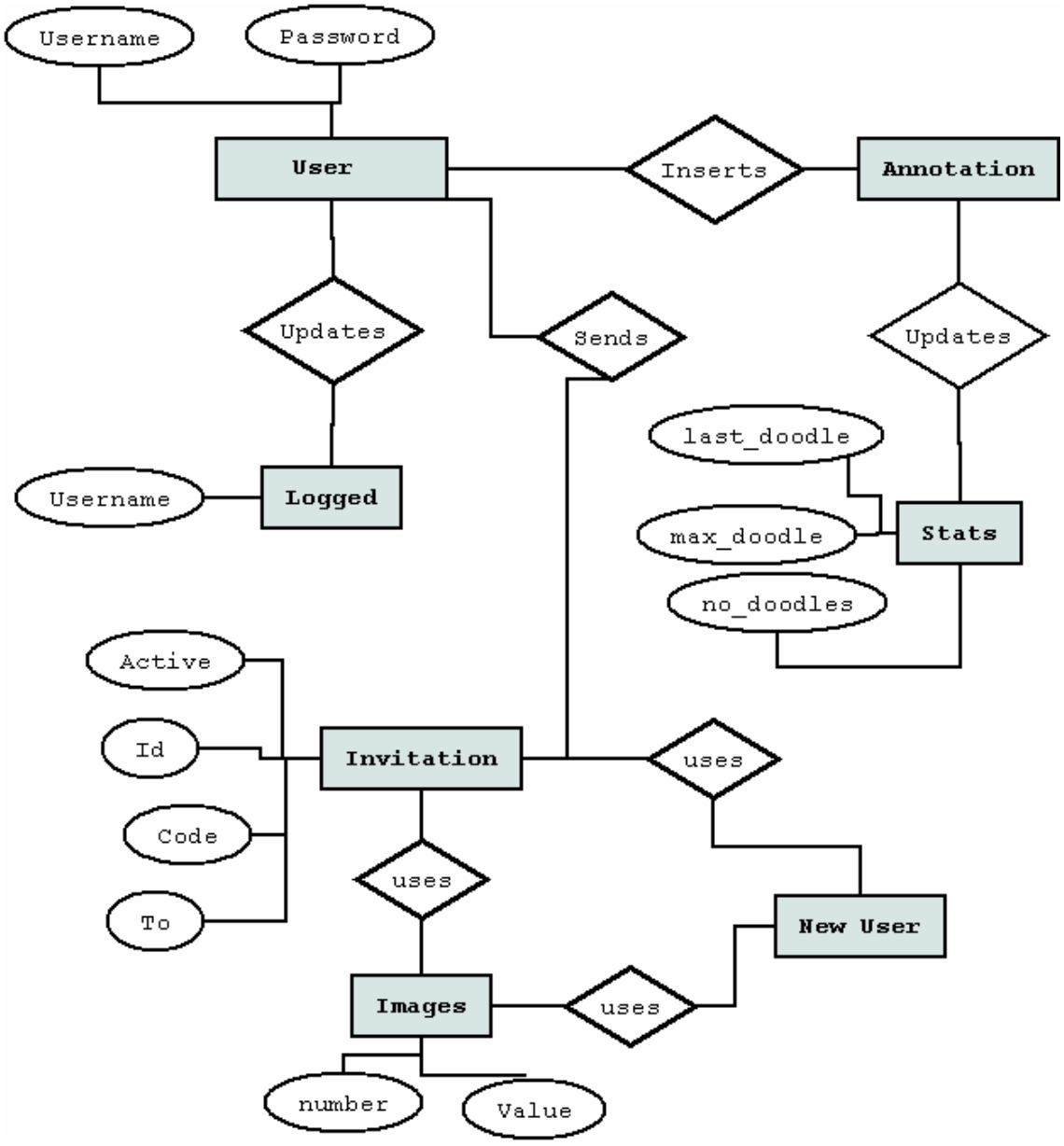

Figure 24 : Tables and their relations

4.4.2 Server functions to provide service for toolbar requests

The toolbar and the server communicate by sending back and forth data. This section lists all the functions that the server perform in accordance to service all toolbar requests.

01. Accept incoming annotation and store it

Server must accept the annotation that has been sent by the toolbar for the purpose of storage and the server must store this annotation in the proper table in the database depending on the username.

02. Delete an annotation

The server must accept an annotation that is sent by the toolbar and which is the one that the user intends to delete from the database. The server must select the proper table from the database and delete the annotation depending on the id of the annotation.

03. Search and send an annotation for adoption

The toolbar would request the server to search a annotation for adoption by the user. The server must search this annotation in the proper table in the database and retrieve the content of this annotation and return it to the toolbar.

04. Search and send a list of shared annotations for a webpage

This server is responsible to search all annotations from all tables in the database which are marked shared and which have the website matching the requested website. After searching these annotations the server must retrieve the content of all these annotations and send them to the toolbar.

05. Update an annotation

This server is responsible to accept an annotation, search this annotation in the proper table in the database with respect to its annotation id and then update the annotation content with the new content and return an error or success code to the toolbar.

06. Search and send friends

This server searches for users in the user table for the requested username and retrieves the data of the user if found and returns this data to the toolbar.

07. Search a user and send information

This server is responsible to accept the username that is sent by the toolbar, search for the username in the user table and if the user is found in the user table then retrieve the information of the user from the table and return this information to the toolbar. If the user is not found then an error must be returned to the toolbar.

08. Add users as friend

The server accepts a user name that is sent by the toolbar to the server, searches for the user in the users table and retrieves the id of this user and inserts it in the friends field of the requesting user and returns a error or success code to the toolbar.

09. Send one annotation

This server accepts the annotation id, searches for the annotation in the respective table based on the annotation id, retrieves the content of the annotation if found and send the content to the toolbar or else sends the toolbar a error code.

10. Get popupid at time of first use

This server is responsible to accept the username from the toolbar and search in the database if the user already has an existing last used popupid, it returns this popupid to the toolbar in case it is found or else a default value considering the user is a new user.

11. Send feedback / email

This server accepts the username, subject of the email and text of the email from the toolbar and sends an email to the author of the system.

01- Accept incoming annotation and store it

The server accepts the annotations that have been sent by the toolbar and saves them to the database and returns a success or error code to the toolbar.

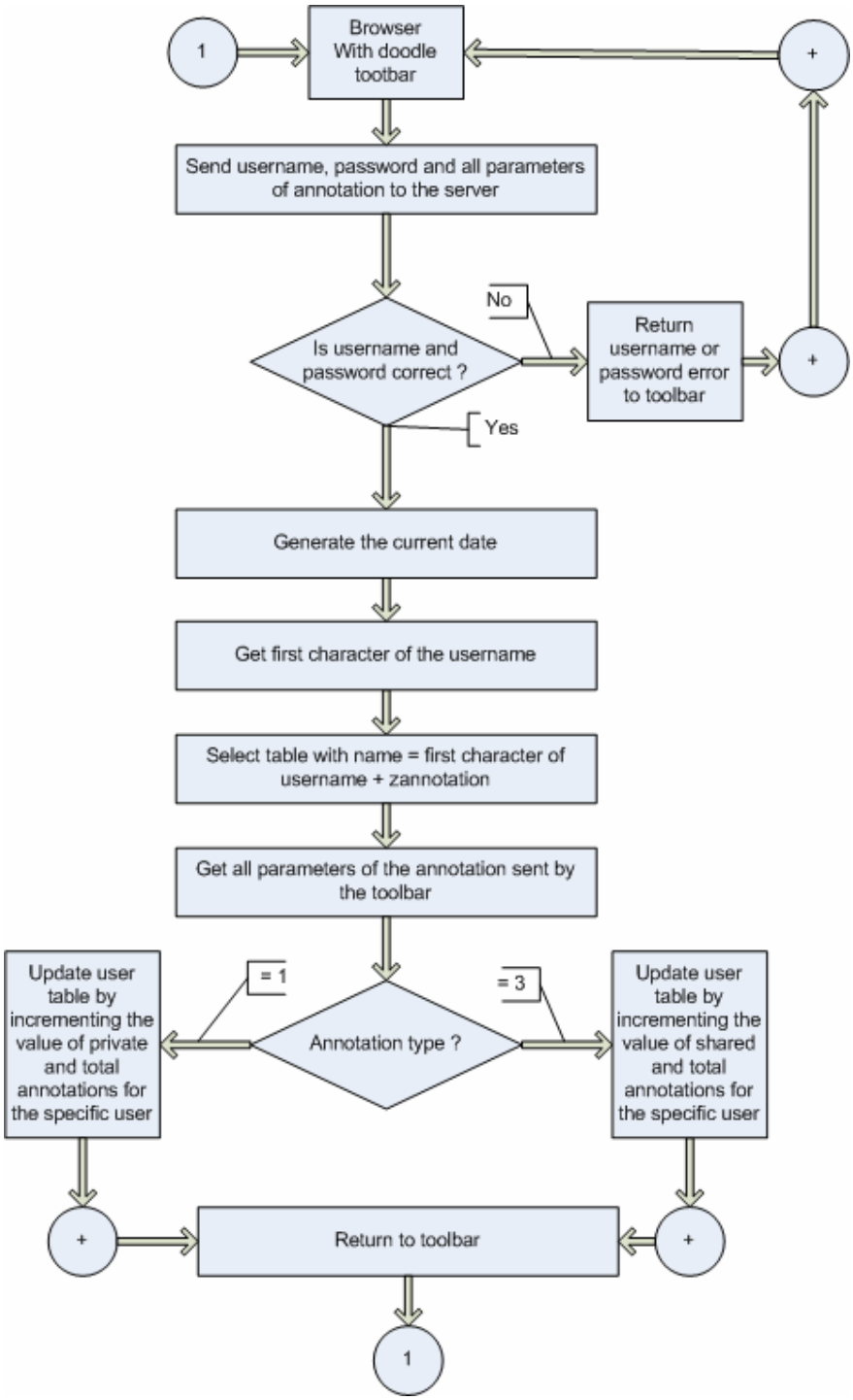

Figure 25 Flow chart for: accept incoming annotation and store it

02 delete an annotation

This piece of server software accepts an id and text of the annotation and searches the annotation from the database and deletes it.

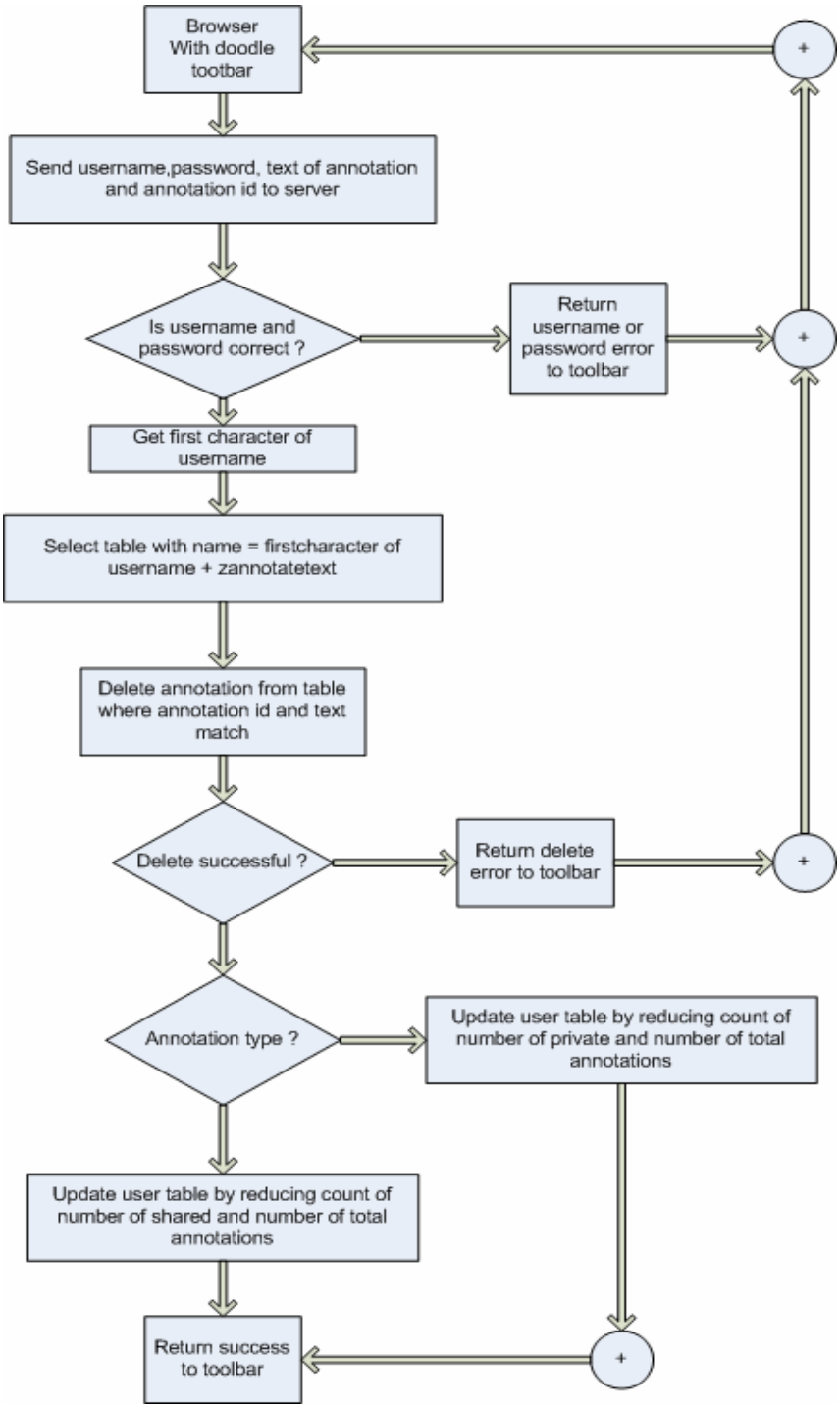

Figure 26 Flow chart for: delete an annotation

03 search and send a shared annotation for adoption

This server code accepts a donor name and the popupid, searches the desired annotation and inserts it into the annotation table of the requesting user.

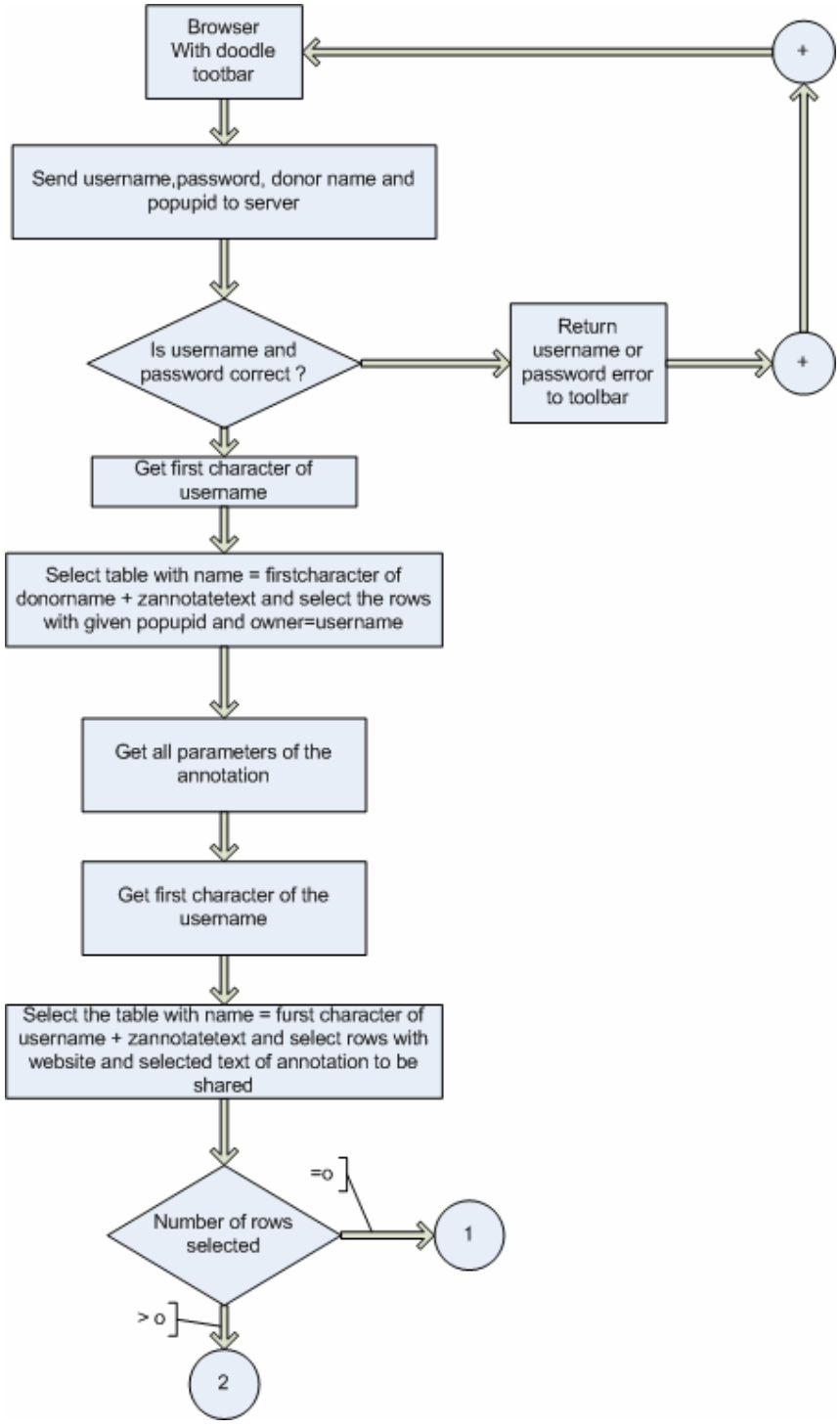

Figure 27 Flow chart for : search and send a shared annotation for adoption. (1)

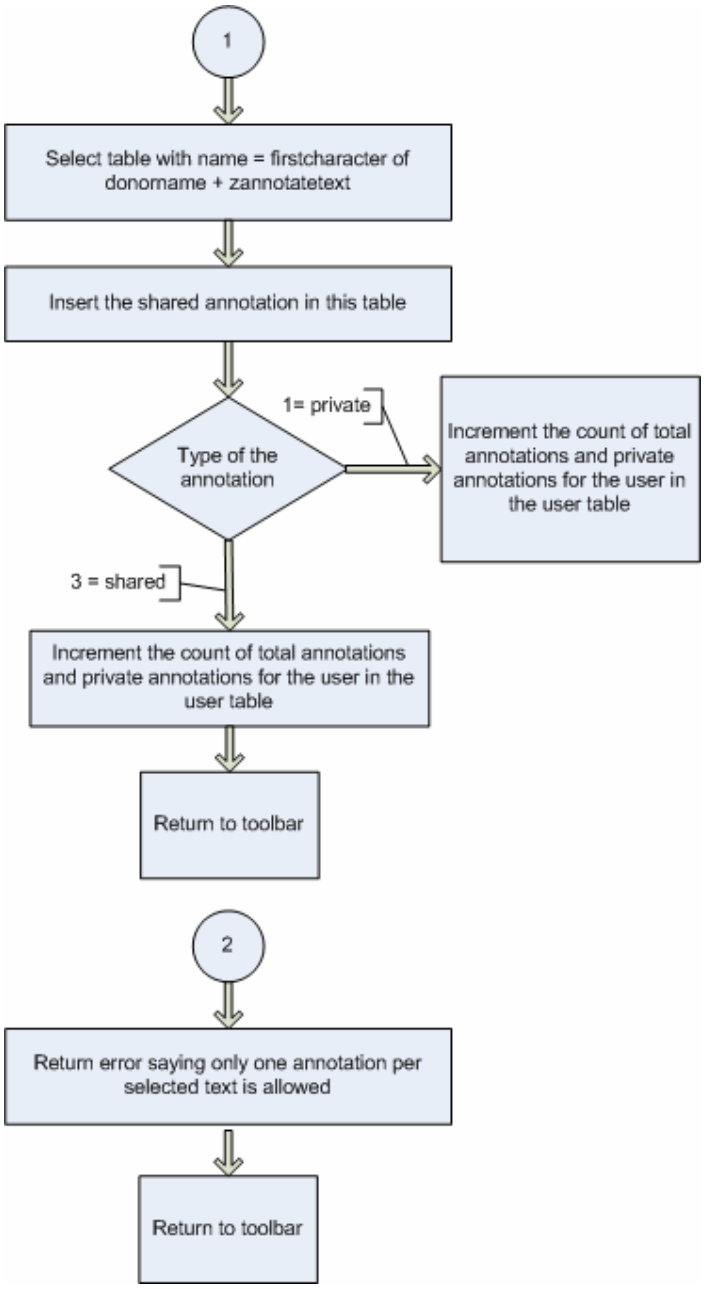

Figure 28 chart for : search and send a shared annotation for adoption.

04 Search and send a list of shared annotations for a web page

This server code searches all the database for shared annotations that have been inserted by other users and returns a list of them to the requester.

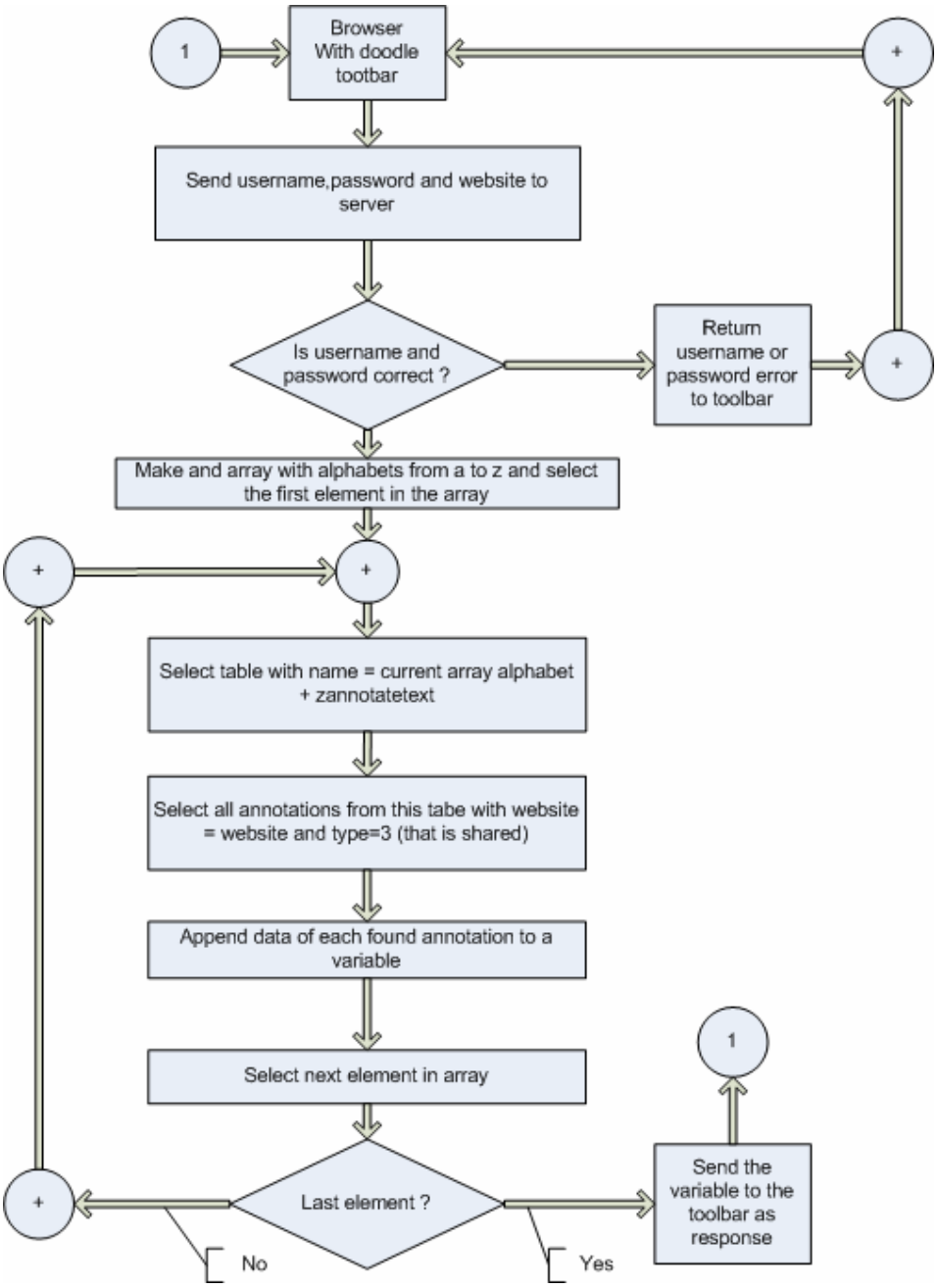

Figure 29 Flow chart for :search and send a list of annotations for a webpage.

05 update an annotation

This is used by the toolbar to update an annotation in the database when the user changes any of the properties of the annotation.

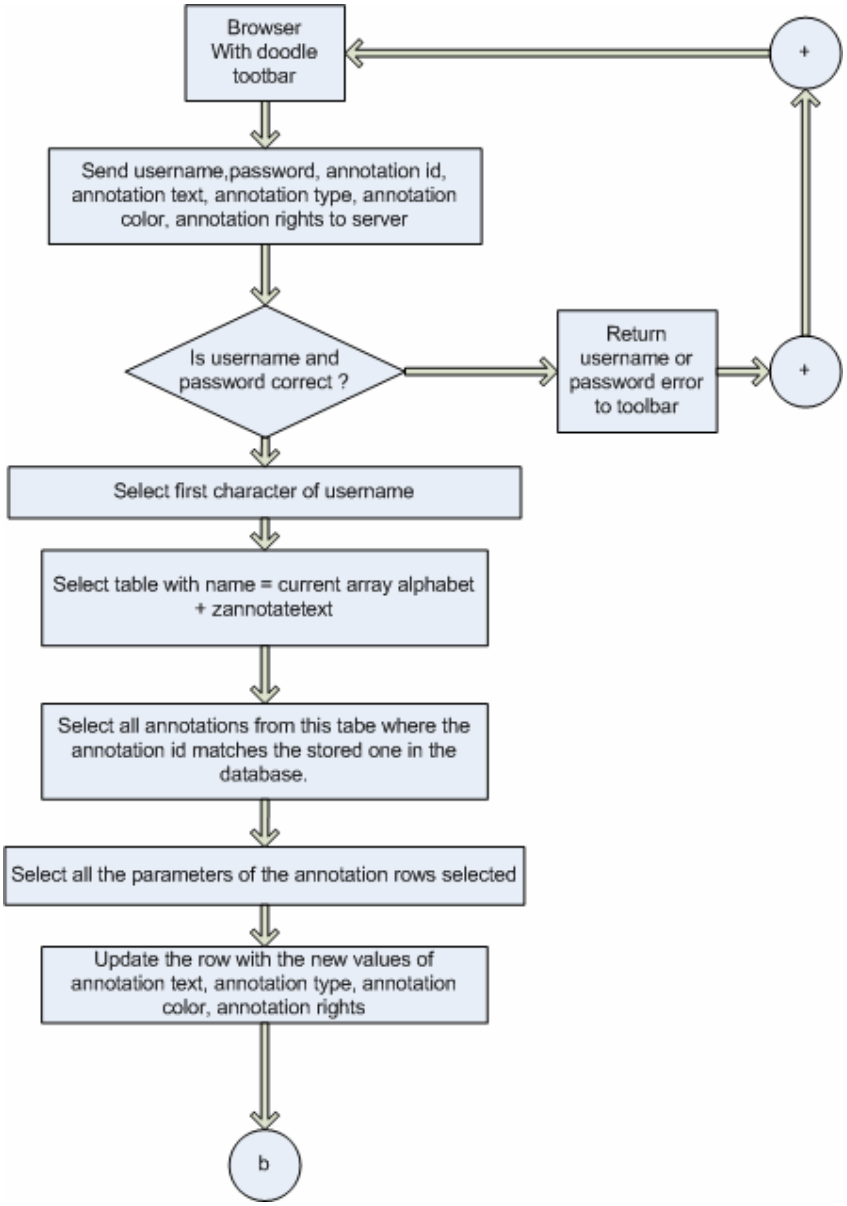

Figure 30 Flow chart For: Update an annotation. (1)

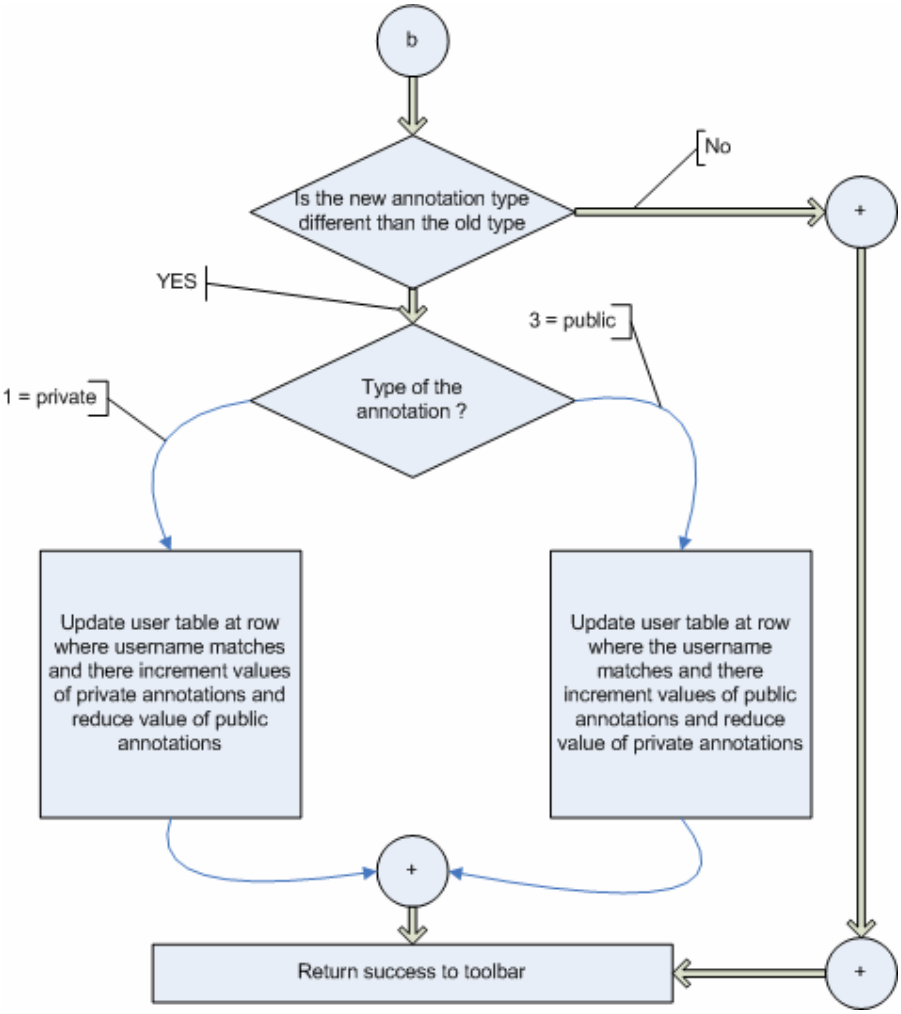

Figure 31 Flow chart For: Update an annotation (2).

06 search and send friends

In order to make user groups a user must be able to add users to his friends list thus this api helps the user to search friends from the database and sends the information of the user to the toolbar.

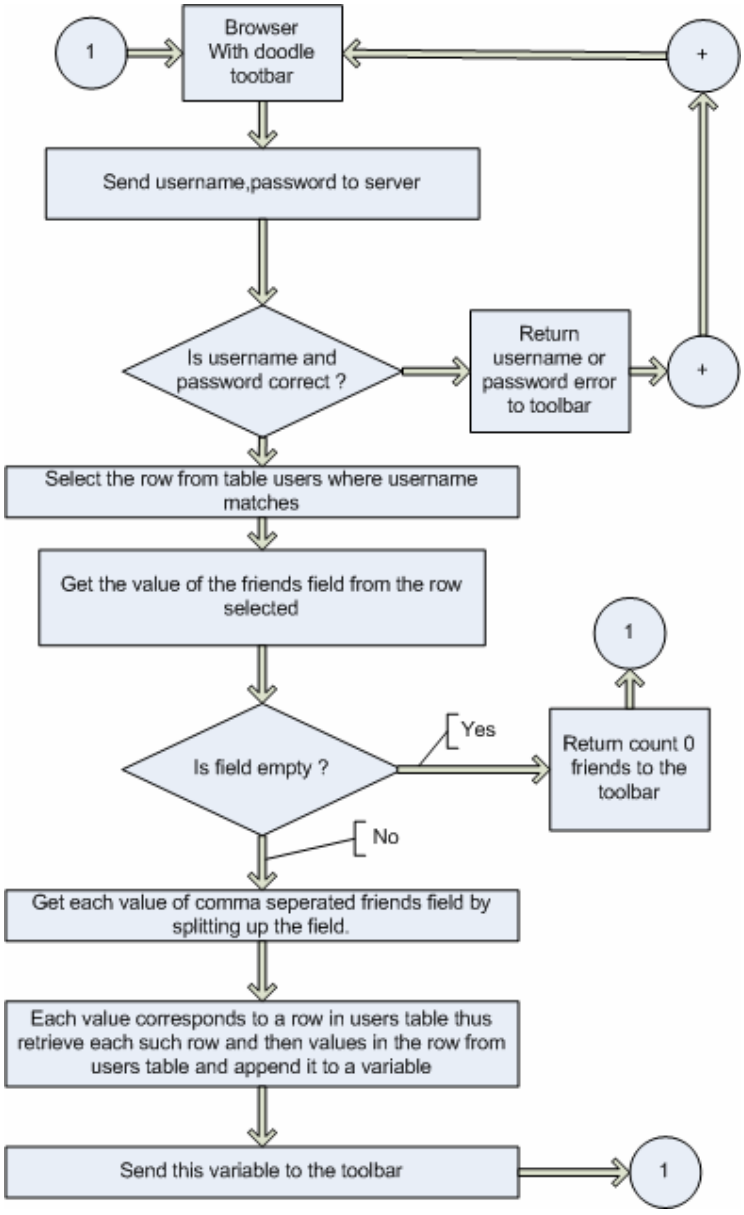

Figure 32 Flow chart for: Search and send friends.

07 Search a user and send information

This server code helps the user to search a specific user and retrieves the information from the database and returns this information to the toolbar.

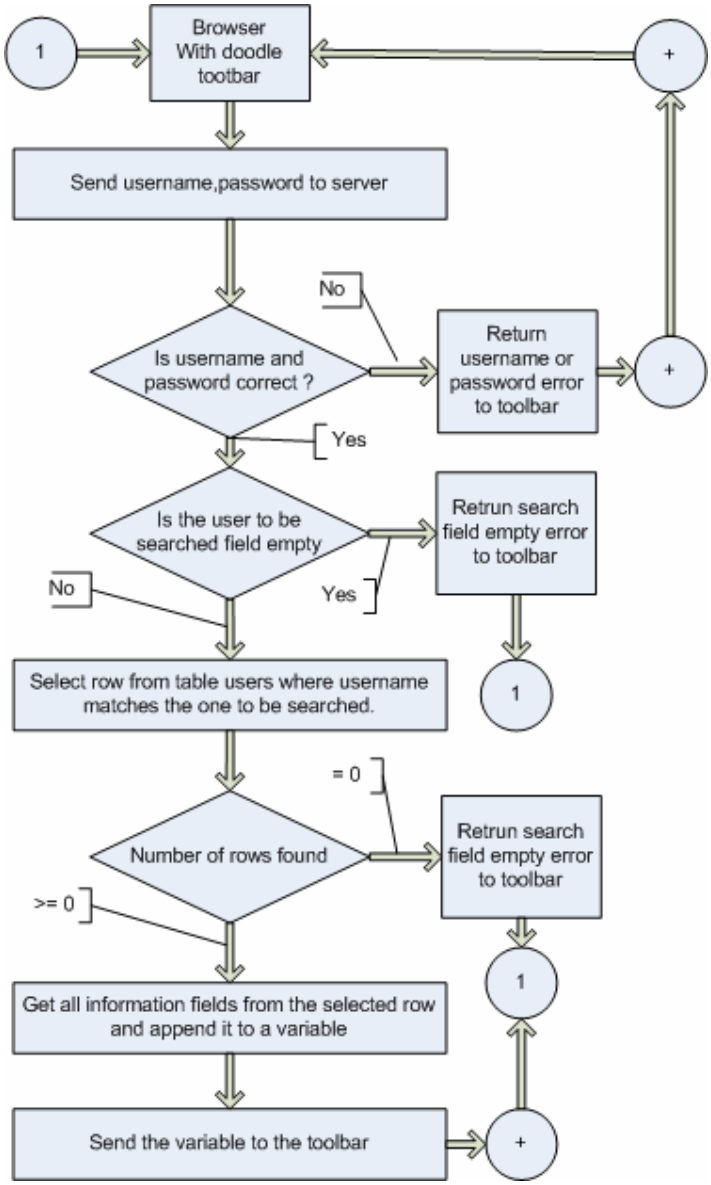

Figure 33 Flow chart for: Search a user and send information.

08 Add users as friend

Once the user searches and gets information about users then this server code helps the user to actually insert a user in the database as a friend.

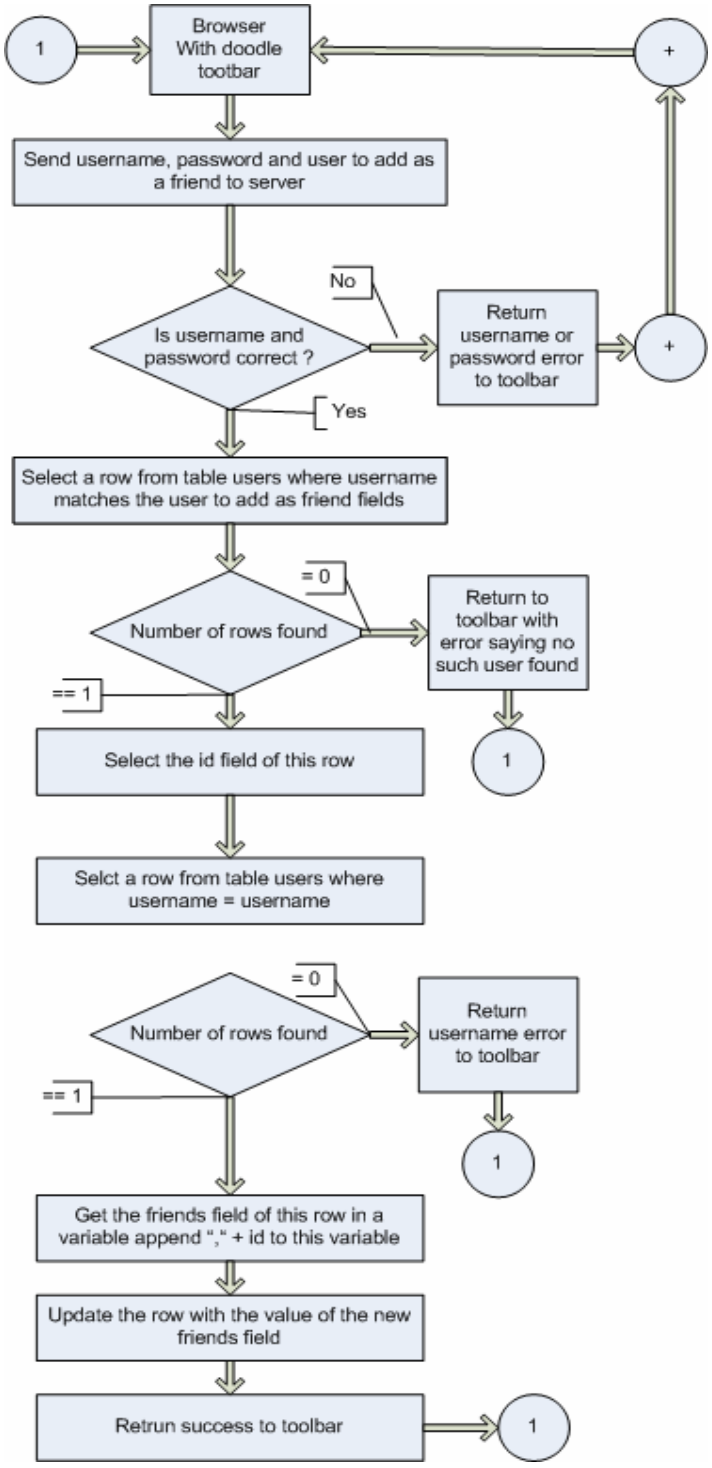

Figure 34 Flow chart for: Add user as friend.

09 send one annotation

This server code accepts username and password and sends the annotation popupid to the toolbar

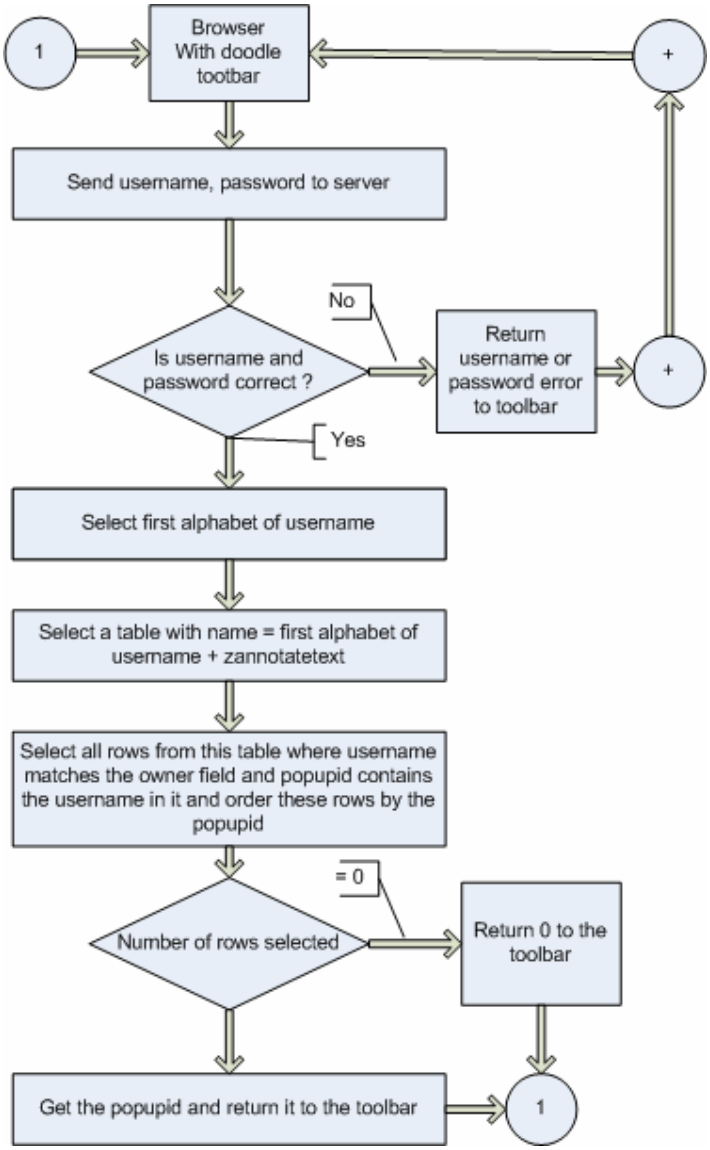

Figure 35 Flow chart for: Send one annotations.

10 get popupid at time of first use

When the toolbar is installed and run for the first time then at that time this server code checks if the user already has some inserted annotations before. If yes it returns the last used popupid else it returns a zero

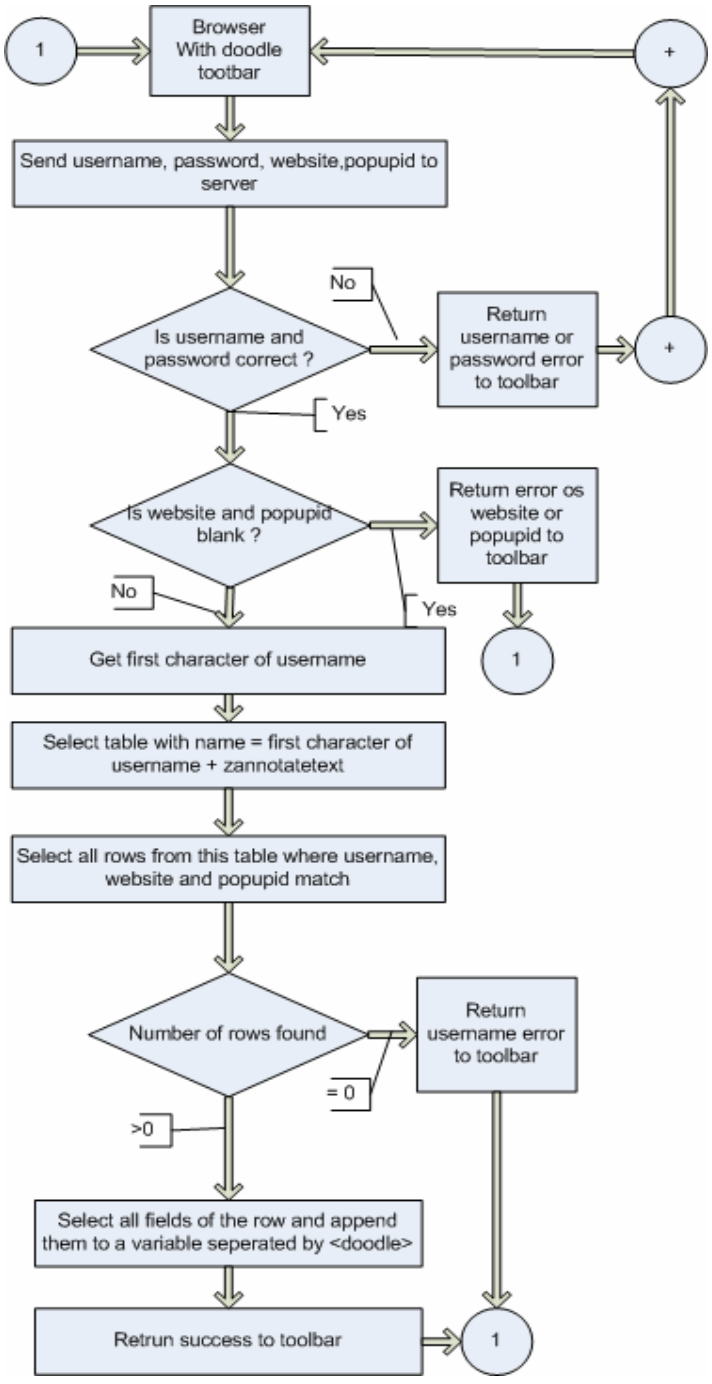

Figure 36 Flow chart for: getting pop up id at the time of first use

11 send feedback / email

This code helps users to send feedback to the author of the annotation engine (in this case to parag@walimbe.com)

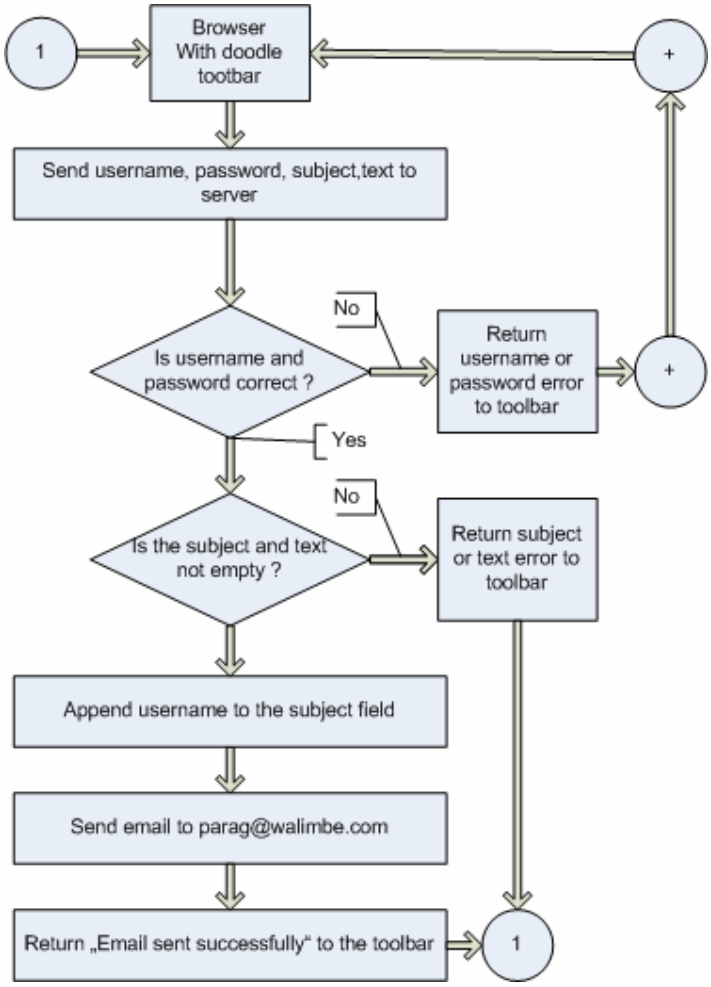

Figure 37 Flow chart for: Sending feedback or email.

4.4.3 Front end of the server.

This part of the work will be only used by the users as a web interface. It is not the most important concept in the implementation of the thesis so is not discussed in detail. The front end basically acts as a interface for user accounting and also has a redundant means for users to manage their annotations. This front end must have pages for user registration, annotation handling and other user related activities. The ability to add users as friends adds the social networking aspect to the front end.

Registration page

This page must accept all user data needed for registration and then check its validity or correctness and then store the data in the database and declare the user as registered. It must also notify the user of a successful or error in registration.

Login page

This page must allow a registered user to login with his username and password. This page prevent unauthorized users to login to the page or the system.

See / delete annotations page

The see / delete annotations page must enable the user of the doodle network or system to see his inserted annotations and must be able to delete them. The annotations must be listed as per the id of the annotations or as per the date that they have been inserted. First the user must be presented with a list of his annotations and then he must be given an option to see the annotation in detail by clicking on the annotation title or any other field of the annotation. The same must be the case with deletion of the annotations. The user must be given an option to click on a delete button associated with the annotation and must be able to delete his annotation.

Share / group share / make private page

This page will allow the user to share or make private his annotations. The page must show the user a list of his annotations and then he must be given an option from where he can make his annotations private if they are shared and make shared if they are private before. The actual toggling between the shared and private states of the annotation are to be done by an intermediate webpage which will return back to this page after the toggling. To toggle the shared or private state of the annotation the field in the database must be set 1 = private and 3 = shared.

See friends page

The server registration is strictly invitation based. That means a user cannot register unless he has been invited to register by another current user. Thus when a user invites a friend to join and the friend joins then the new user is added to the list of friends of the user who has invited him. Thus the user must be given an option where he can see a list of his friends and some brief description about them. The information about the friends is stored in the users table in the column “friends” of each of the row. This field is a comma separated field. Thus

when the list of friends is to be displayed then this comma separated field must be separated to different numbers which correspond to the id of the users.

See friends profile page

This page is to be used when the user clicks on the name of the user from the see friends page. The input to this page is the id of the user and thus the code behind must retrieve all information about the user from the user table depending upon the id of the user. The user can go back to the see friends list page from here.

Search and add user as friend

Annotations can be set as shared. Now other users can adopt these shared annotations and thus understand what the author of the annotation intended to say by the annotation text. Now there is also an option where user wants to share his annotations only with a specific set of people like share his annotations only with his friends and not with all the users of the network. In this case the user of the network would like to search users of the network with parameters like the id of the user or the username, name or last name of the user of the network. Thus this page allows the user to search for users in the network and then consecutively add the found users to his friends list. A number of other operations can be done with the friends list like a group can be split in to subgroups and the annotations can be shared with the whole group or only the subgroup etc.

Send an invitation page

As discussed earlier, the doodle network is an invitation only network. Thus users already in the network must sent invitations to other friends to join the network. Thus this page will enable users to send invitations. Each user of the network has 3 invitations to be sent. The user puts the name and email address of the friend he wishes to invite to the network and the invitation is thus sent to the friend on by the network on behalf of the user.

This figure shows the general structure of the server front end. The proc files are the process files that do the actual processing.

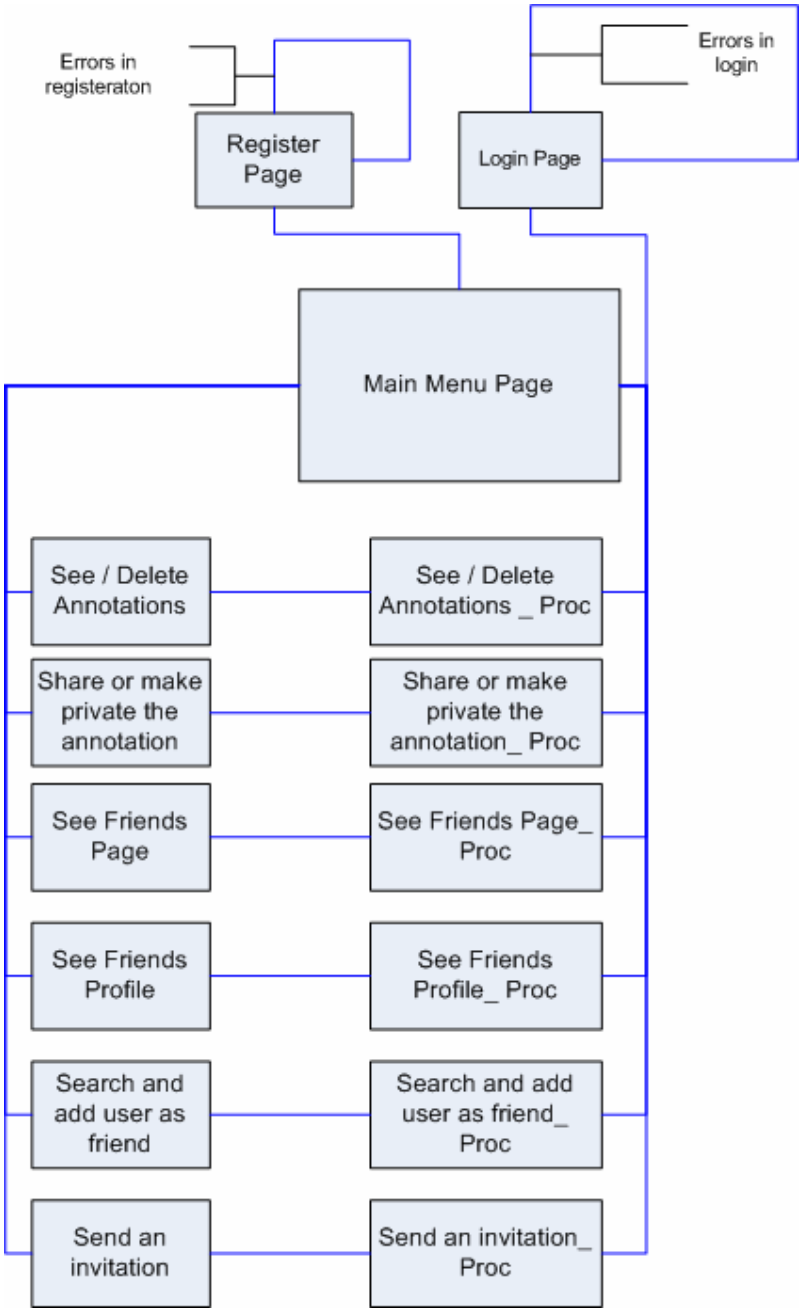

Figure 38 Front End of user management on server.

4.5. Toolbar

4.5.1 Front end (XUL)

The front end of the toolbar is written in XUL (XML user interface language).

What is XUL?

XUL was created to make development of the Mozilla browser easier and faster. It is XML languages so all features available to XML are also available to XUL.

XUL has all the advantages of other XML languages. For example XHTML or other XML languages such as MathML or SVG can be inserted into it. Also, text displayed with XUL is easily localizable.

XUL provides the ability to create most elements found in modern graphical interfaces. Some elements that can be created are:

- Input controls such as textboxes and checkboxes
- Toolbars with buttons or other content
- Menus on a menu bar or pop up menus
- Tabbed dialogs
- Trees for hierarchical or tabular information
- Keyboard shortcuts

The displayed content can be created from the contents of a XUL file or with data from a datasource. In Mozilla, such datasources include a user's mailbox, their bookmarks and search results. The contents of menus, trees and other elements can be populated with this data, or with your own data supplied in an RDF file.

XUL can be used to make firefox extensions or toolbars. An extension adds functionality to the browser itself, often in the form of extra toolbars, context menus, or UI to customize the browser UI. This is done using a feature of XUL called an overlay, which allows the UI provided from one source, in this case, the Firefox browser, to be merged together with the UI from the extension.

Toolbar functionality.

At the end of the related works section a discussion is done to find out the advantages and disadvantages of other related work. The advantages of each system need to be implemented in the new work and the disadvantages need to be avoided. From that discussion the following functionality for a toolbar is derived.

- Ability to select text and insert annotation with reference to selected text.
- Ability to send the inserted annotation to the server.
- Ability to retrieve annotations from the server on demand.
- Ability to display the retrieved annotations according to user preferences.
- Ability to edit annotations by clicking on the edit button.
- Ability to send the updated content of the edited annotation to the server
- Ability to minimize toolbar

- Ability to restore toolbar to original mode from minimized state.
- Ability to search and adopt shared annotations.
- Ability to see friend list and friend profiles.
- Ability to search user by ID, name and ad as friend.
- Ability to create user groups.
- Ability to go to online account.

Basically the functionality requires that after an annotation is inserted by a user and he quits the insert window then the annotation must be automatically inserted in the DOM content that has previously been loaded in the browser. Similar is the case with edition, deletion, adoption of annotations or just a normal load of annotations.

As the browser is an event driven environment thus some events are to be tapped and the proper functionality of the annotation engine is triggered.

Three events are tapped which are as follows

- When the user loads a webpage in the browser window
- When the user clicks the refresh button
- When the specific window or tab gets focus.

Each of these hooks triggers different functionality because they are different events.

There are 9 possible states that the toolbar can be in they are.

1. Wait for event
2. Browser is focused
3. Browser content is reloaded
4. DOM content is loading is complete
5. Returned after inputting an annotation
6. Loading annotations
7. Returned from the editing window
8. Fetching annotations from server
9. No annotations to load.

These 3 event tapping with the 9 states work in the following way.

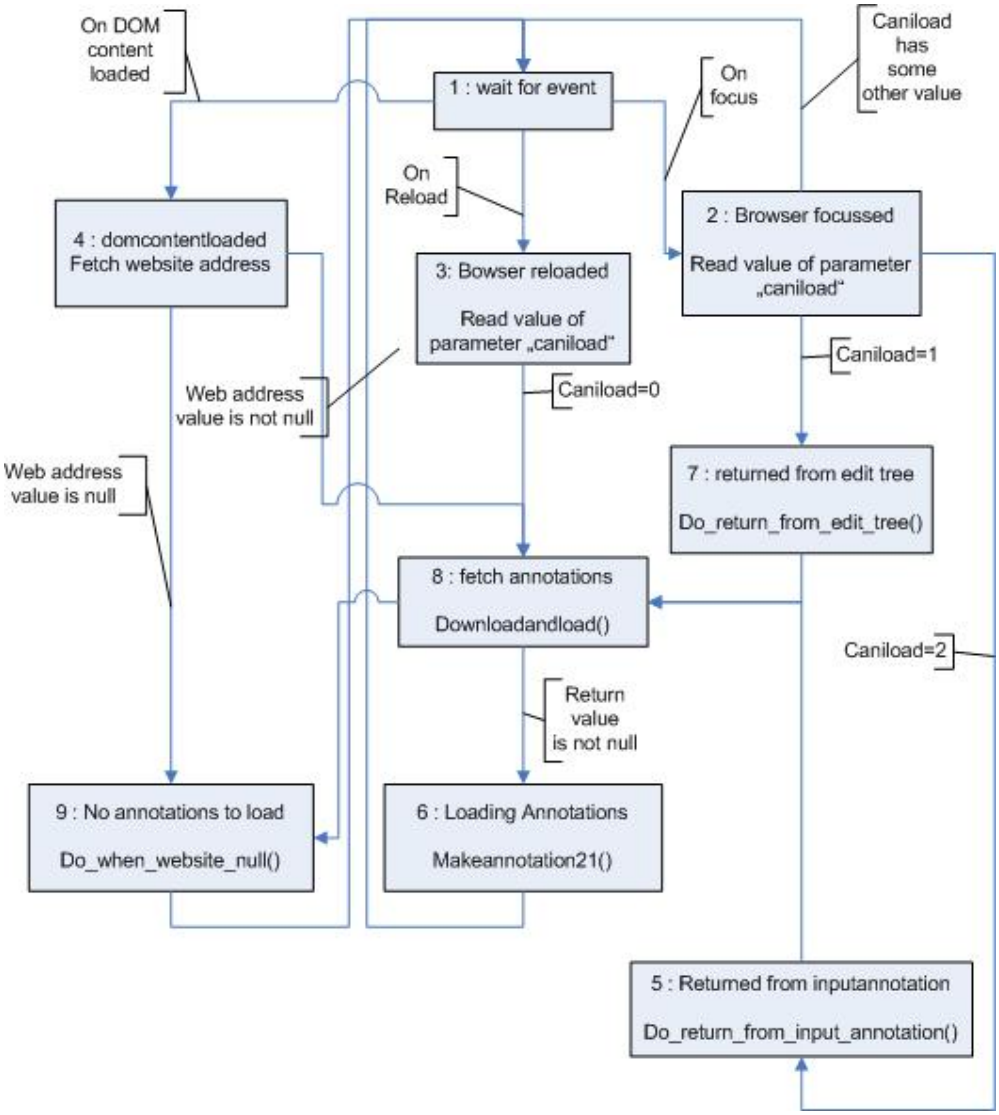

Figure 39 : State chart for toolbar functionality.

Remaining functions

These are some of the remaining functions which are useful but not the most important thus only their functionality is stated here.

- **get_all_from_server**

This function must send username, password, Web site to the server and must wait for an response from the server.

- **search_annotation**

Should accept selectedtext, pname, anchornodevalue, atype and should search an candidate node and return failure or the node reference on success

- **get_one_from_server**

Must accept username, website, password, popupid and send to server and must wait for response

- **makeannotation21**

Must accept as input the following, id, annotationtext, selectedtext, popupid, acolor, anntype, time, owner, date, whichnode, anchoroffset and must search the anchor node and create a popup at the specified location of the selected text.

- **inputvalues**

Must prepare the values username, password, anchoroffset, pname, ptype, pvalue, atype, avalue, selectedtext, website, popupid and send it to the input annotation window.

- **openprefs**

Must open the user preferences window where the user can set his username and password

- **gomini**

Must collapse doodlebar and show the small doodlebar icon in the navbar

- **gofull**

Must delete the small doodlebar from the navbar window and show it in full mode

- **getpopupid**

Must get the user preference doodlebar.doodle_popup_id and return it to the calling function.

- **deleteunwanted**

Must get user preferences value doodlebar.deleted and delete the loaded annotations from browser window which are stated in the deleted variable

- **openwiz**

Must open the first run wizard window which will set all default parameters and will also set the username and password.

- **setvalues**

Must set default values of user preferences on first run of the toolbar. The user preferences to be set are

- doodle_popupid
- caniload
- deleted
- wait_load
- autoload

Chapter 5 - Implementation

5.1 Server

5.1.1 Backend (MYSQL)

The backend of the server needed to create various tables to store, retrieve, insert and display annotations. The most important tables are the invitation, users, images and the annotatationtext table. The codes for the creations of the other tables are provided on the cdrom where all the source codes exist. The users table is used to store the information about the users. The images table has images which are used for visual verification from the user in terms of an image. The invitations sent by users to their friends will be stored in the invitation table. The figure below shows the tables in the database and their relation.

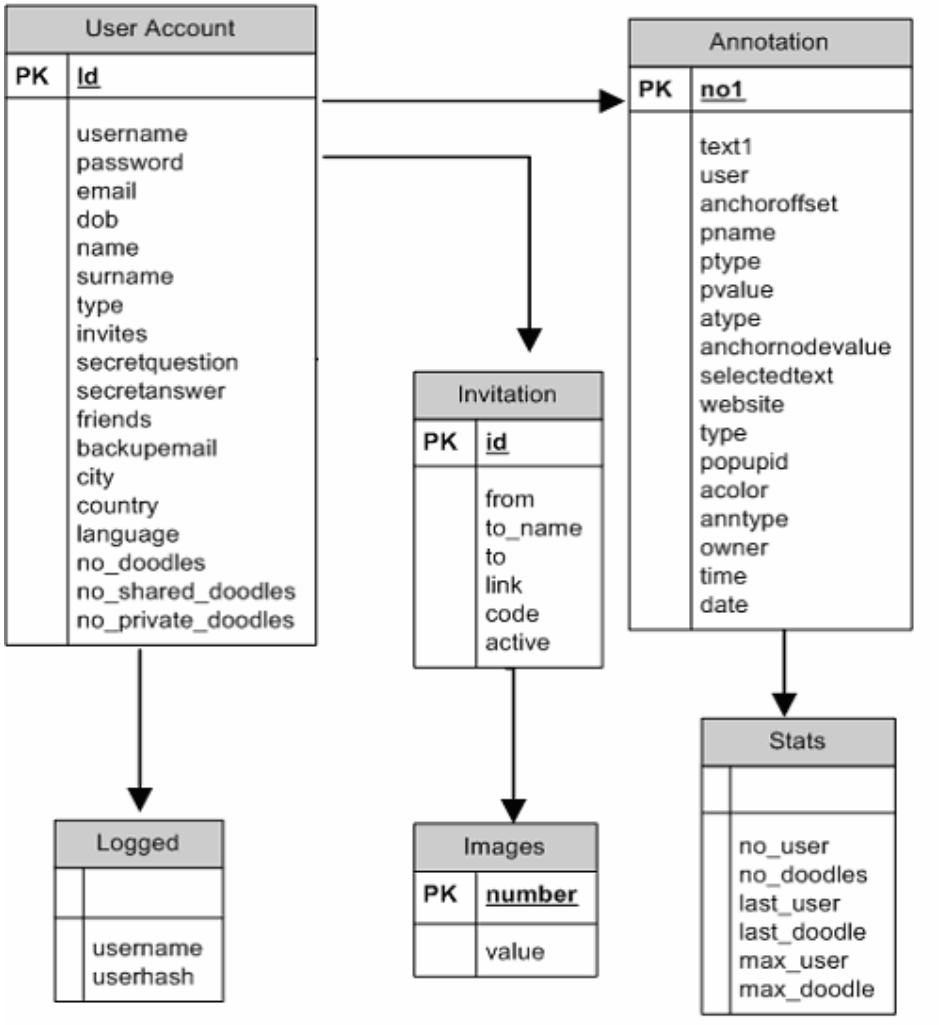

Figure 40 Tables used in the database

5.1.2. Server functions to server toolbar requests.

This is the part of the implementation that can be accessed and which interacts with the toolbar. These codes can also be used by 3rd party client software to interact with the server. The following diagram shows the files that exist on the server.

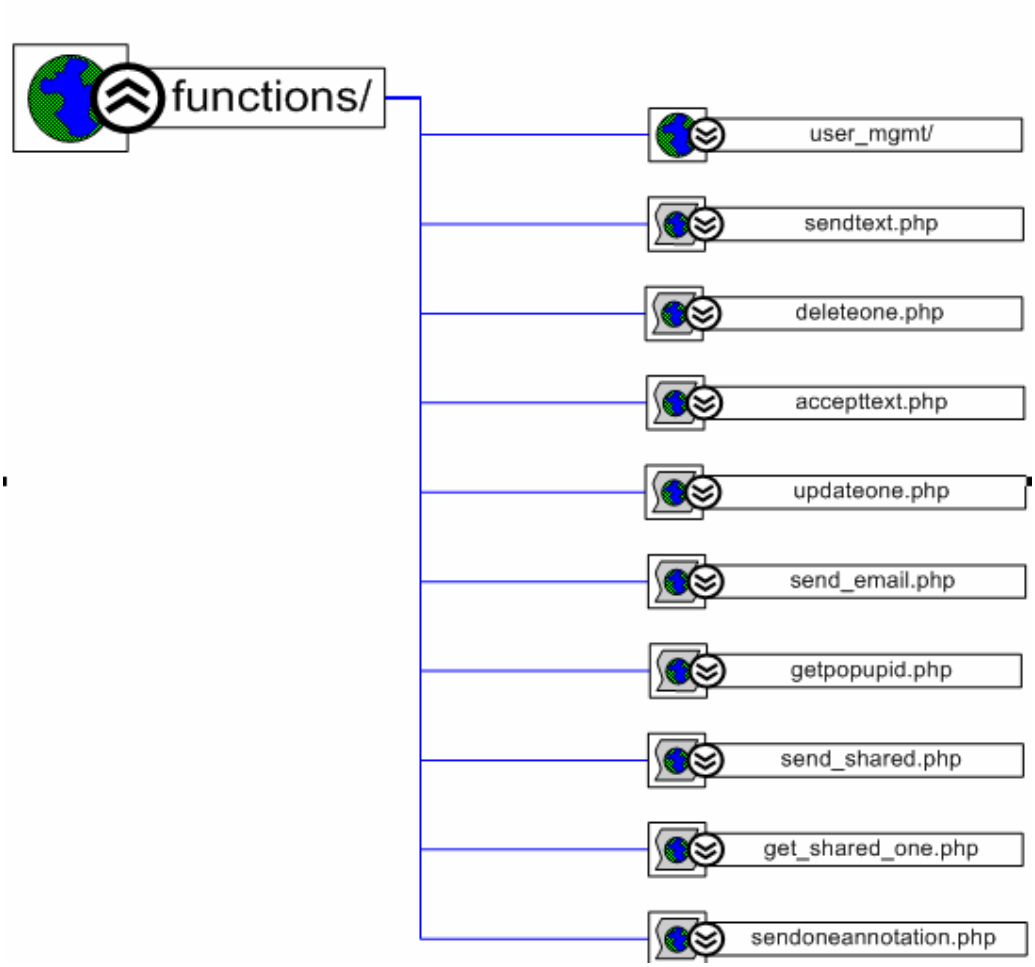

Figure 41 : Server functions to serve toolbar requests

Here I list the most important code samples of the most important functions of the server.

sendtext.php

This file has functions to send annotations to the toolbar. This is how annotations are send by the server to the toolbar. The contents are appended with <doodle> inserted between the fields. And the count inserted between the annotations

```
$send.= "<doodle>".$firstofit.$row1['no1']
."<doodle>".$row1['text1']
."<doodle>".$row1['anchoroffset']
."<doodle>".$row1['pname']
."<doodle>".$row1['pvalue']
```

```
."<doodle>".$row1['ptype']
."<doodle>".$row1['atype']
."<doodle>".$row1['anchornodevalue']
."<doodle>".$row1['selectedtext']
."<doodle>".$row1['website']
."<doodle>".$row1['type']
."<doodle>".$row1['popupid']
."<doodle>".$row1['acolor']
."<doodle>".$row1['anntype']
."<doodle>".$row1['owner']
."<doodle>".$row1['time']
."<doodle>".$row1['date']
."<doodle>"
.$howmany;
```

- **deleteone.php**

This piece of code takes annotation id and annotation text as input and deletes the corresponding annotation from the database and returns either a success code or an error code to the toolbar. The query for the same is as below

DELETE FROM `annotatetext` WHERE `text1`='". $ann_text1. "' AND `no1`='" $ann_id."'

- **accepttext.php**

This code file on the sever accepts various parameters sent by the toolbar like annotationtext, username, anchoroffset, pname etc and inserts that into the table in the database. The query for the same is as under.

INSERT INTO `annotatetext` (`no1` , `text1` , `user` , `anchoroffset`, `pname` ,`ptype`,`pvalue`, `atype` , `anchornodevalue` , `selectedtext` , `website` , `type`,`popupid`,`acolor`,`anntype`,`owner`,`time`,`date`) VALUES (NULL, '$text1', '$username','$anchoroffset','$pname','$ptype','$pvalue','$atype','$avalue','$selectedtext','$website' , '$type','$popupid','$acolor','$anntype','$username','$time','$date')";

- **updateone.php**

This file has functions that updates the values of the annotation that has been sent by the toolbar. The query for the same is as below.

UPDATE `annotatetext` SET
`text1`='" . $ann_text1. "' , `anntype`='" . $ann_type. "' , `acolor`='" . $anchor_color. "' , `type`='" . $ann_rights. "'
WHERE `no1`='" . $ann_id ."'

- **send_shared.php**

this piece of server software accepts the website address from the toolbar and searches the database for shared annotations that are linked to the website address. If annotations are found then it formats the various data separated by tags of <doodle> and then send them to the toolbar. The query for the same is as under.
SELECT * FROM " .$tablename ." WHERE user!='".$username."' AND website='".$website. "'" ." AND type='3'"

5.1.3. Front end (User mgmt)

The front end of the server is basically for users to manage their account data and to manage their annotations online. It is just a redundant system to edit account and annotations. All the functions can be achieved by the toolbar also.

The screen shot below shows the front end of the server. Where the user can do a number of activities, like changing account information, Inviting friends, managing annotations, uploading a profile photo etc. Social networking is possible when users share their annotations and add other users as friends.

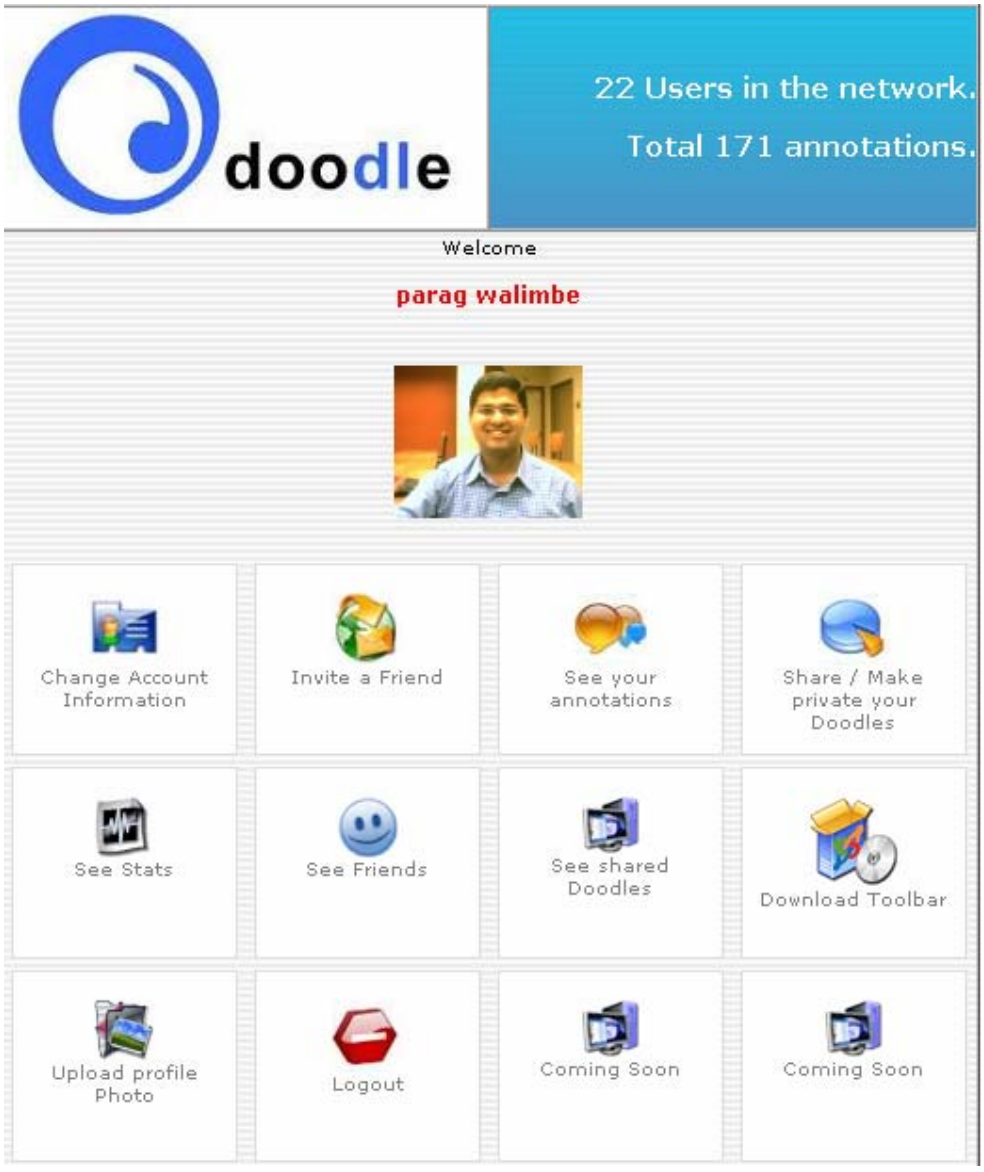

Figure 42 Front end of the server

Codes for the functions can be found on the cdrom that is attached with the report.

5.2 Toolbar

The front end of the toolbar is made in XUL (xml user interface language). The background functionality is written in JavaScript. The communication with the server is achieved via the http protocol.

5.2.1 Process of making the toolbar

- Place all the code files (xul, js) into the content folder
- Place all the image files and the css in the skin folder
- Each of these folders will have a contents.rdf file
- Place both the folders in a chrome folder
- Make a jar file from the skin and the content folder
- Place the chrome folder in the paragext folder
- The paragext folder will also have the install.rdf file whose code is given below
- Make a .xpi file out of the chrome folder and the install.rdf file
- The xpi file is the actual file that is recognized by the firefox browser and will install the toolbar in the browser.

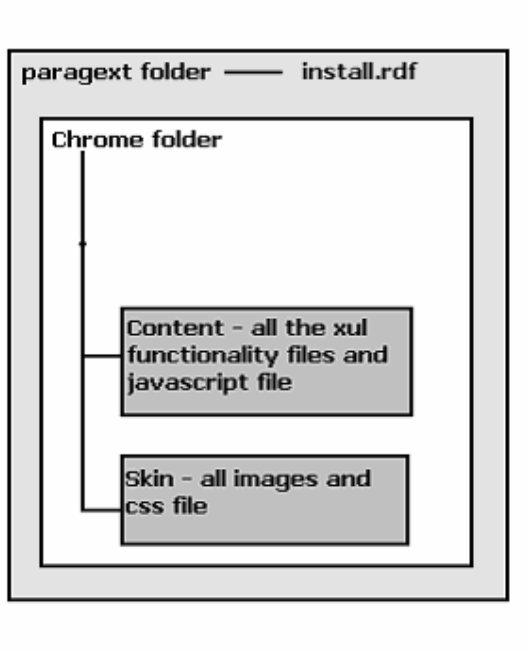

Figure 43: Toolbar directory structure

5.2.2. The install file

Install.rdf

This is a file that tells the firefox browser about the toolbar that is contained in the .xpi file. The em tags have information about the creator, description etc about the toolbar. The em:id must be a unique id

```
<?xml version="1.0"?>
<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:em="http://www.mozilla.org/2004/em-rdf#">
<Description about="urn:mozilla:install-manifest">
<em:creator>parag walimbe</em:creator>
<em:description>The annotation engine for web content</em:description>
<em:homepageURL>http://walimbe.com/</em:homepageURL>
<em:id>{3d21d690-246b-11db-a98b-0800200c9a66}</em:id>
<em:name>Doodle Toolbar</em:name>
<em:version>1.2</em:version>
<em:iconURL>chrome://paragext/skin/logo_blue.png</em:iconURL>
<em:optionsURL>chrome://paragext/content/prefs.xul</em:optionsURL>
<em:aboutURL>chrome://paragext/content/about.xul</em:aboutURL>
<em:updateURL>http://www.walimbe.com/sis/paragext/update.rdf</em:updateURL>
<em:targetApplication>
          <Description>
<em:id>{ec8030f7-c20a-464f-9b0e-13a3a9e97384}</em:id>
<em:minVersion>1.0</em:minVersion>
<em:maxVersion>1.5.0.*</em:maxVersion>
</Description>
</em:targetApplication>

<em:file>
<Description about="urn:mozilla:extension:file:paragext.jar">
<em:package>content/</em:package>
<em:skin>skin/</em:skin>
</Description>
</em:file>
</Description>
</RDF>
```

5.2.3. Contents of the content folder

The contents folder has all the source files for the widgets that appear in the front end of the toolbar. This folder also contains the functions in form of javascript file that form the backend of the toolbar.

- about.xul

  this file is an xul file that has code to tell the user of the toolbar information about the toolbar. This has information about the title, version, creator, contact details. It looks like the following snap shot.

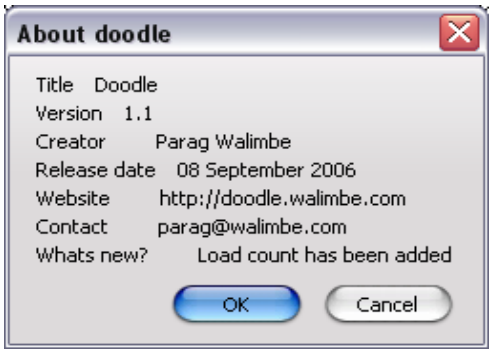

Figure 44 : about.xul in the browser

- advanced_options.xul

This file contains the advanced options of the toolbar where the user can search for shared annotations by other users and can also send a bug report via email to the author of the toolbar. Figure 46 below shows this xul file in action.

The source codes of the other files can of this folder can be found in the CDROM attached with the report.

5.2.4 Contents of skin folder

This folder contains all the images that appear in the toolbar or will be used by the toolbar. It also has the various cascading style sheets required by the toolbar.

5.2.5 Screen shots

This is how the toolbar looks when installed in the firefox browser.

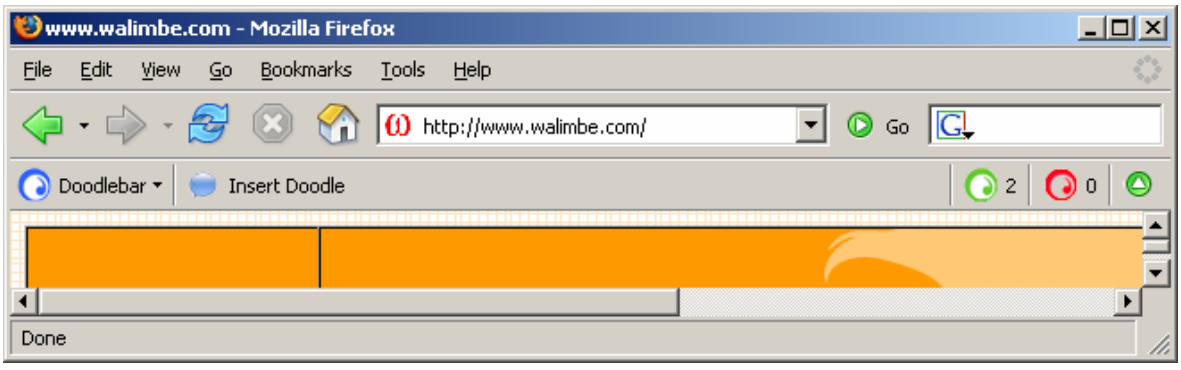

Figure 45 : The doodle toolbar

This is how the annotation popup looks when the mouse is rolled over the anchor text.

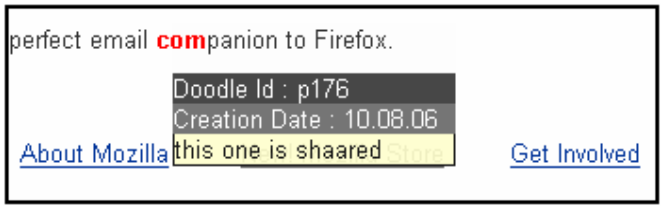

Figure 46 : Actual popup of the annotation with the colored anchor text

This is the annotation insertion window where the user can insert his annotation text for the selected anchor or base text.

Figure 47: Insert annotations window

This is the annotation edition window.

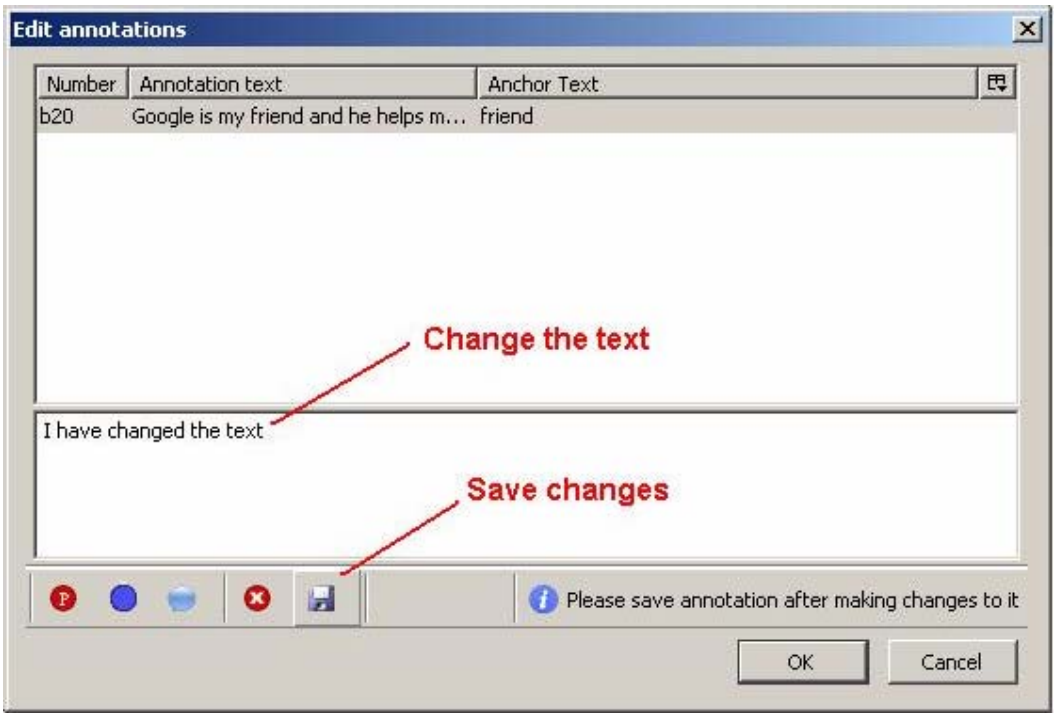

Figure 48 : Edit Annotations window

This is the advanced options window where user can adopt shared annotations or send bug reports.

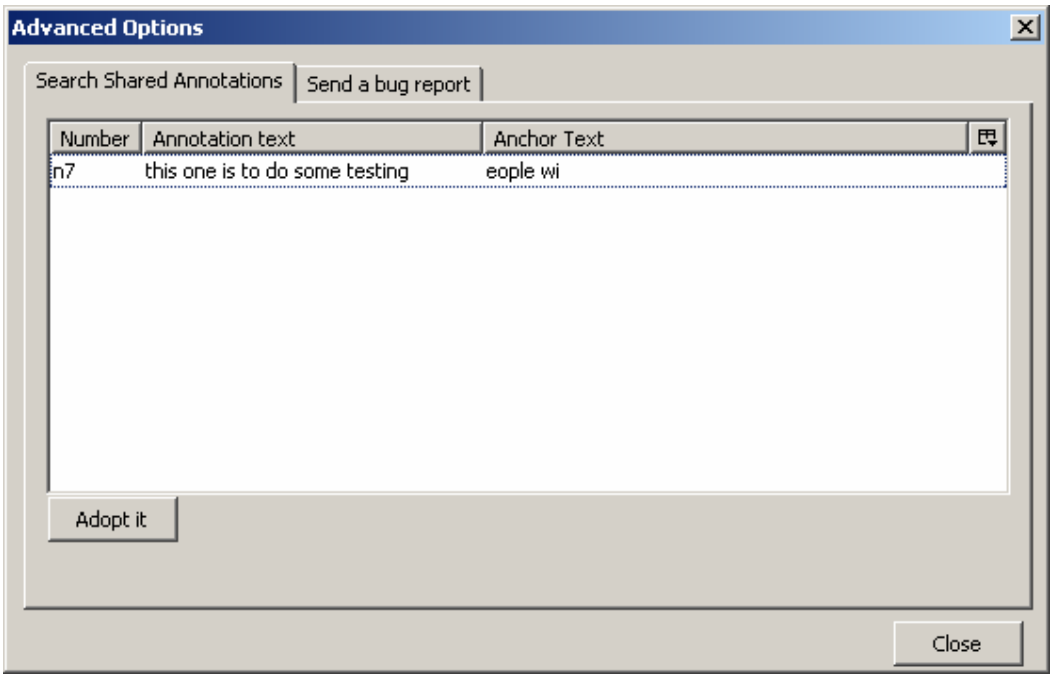

Figure 49 : Advanced options

5.2.6. Sample codes from the toolbar code files

5.2.6.1. Setting user properties in the browser

The following code sets variables as user properties in the browser.

```
var tabPrefs = Components.classes["@mozilla.org/preferences-
service;1"].getService(Components.interfaces.nsIPrefBranch);
tabPrefs.setCharPref("doodlebar." +  "caniload",'0');
```

5.2.6.2 Retrieving user properties in the browser

The following code retrieves variable from user properties.

```
var tabPrefs = Components.classes["@mozilla.org/preferences-
service;1"].getService(Components.interfaces.nsIPrefBranch);
var okok1=tabPrefs.getCharPref("doodlebar." +  "caniload");
```

5.2.6.3. Connecting to server, sending data to server and waiting for response

```
try {
       var httpRequest = new XMLHttpRequest();


      }catch (e){
       alert('Error creating the connection!');
       return false;
      }

         try {

       targetURL=server+ "functions/sendoneannotation.php";

       var txt="?"+ "username="+  username +  "&website=" +  website +
"&password=" +  password +  "&popupid=" +  popupid;


       httpRequest.open("POST", targetURL+txt, false, null, null);
       httpRequest.setRequestHeader("Content-Type","application/x-www-form-
urlencoded");
       httpRequest.send('');


       if(httpRequest.readyState==1 || httpRequest.readyState==2 ||
httpRequest.readyState==3)
```

```
            {
            alert('Bad Ready State: '+ httpRequest.status);
            return false
            }

            if(httpRequest.readyState==4)
            {
            if(httpRequest.status !=200) {
            alert('The server respond with a bad status code: '+httpRequest.status);
            return false;
            }
            else {
            var response1 = httpRequest.responseText;
            return response1;
            }
        }


    }

    catch (e) {
    alert('An error has occured calling the external site: '+e);
    return false;
          }
```

5.2.6.4. Action listeners

Action listeners are the events that are to be listened for. These events can be seen in the state diagram chart in Figure 37. The following code shows how one of these events is listened for.

```
if (document.addEventListener) {
    document.addEventListener("DOMContentLoaded", downloadandload, false);
}
```

5.2.6.5. Open a popup xul window by passing parameters to it

```
window.openDialog("chrome://paragext/content/prefs.xul","clustybarPrefsDialog",
"chrome");
```

5.2.6.6. Accessing the nav-bar

The following is the code to access the navigation bar.

```
var mainNavBar = document.getElementById("nav-bar");
```

5.2.6.7. Accessing parameters passed to the xul popup window

The following is the code to pass parameters to a xul pop window.

```
var username=window.arguments[1];
```

5.2.6.8. Creating an xpi installer link

A registered user must be able to install the toolbar on a single click. The code for doing the same is as follows.

```
<a href="#" onclick="xpi={'Doodle Toolbar - Version
1.0':'http://www.walimbe.com/thesis/paragext/v1.0/paragext.xpi'};
  InstallTrigger.install(xpi);">Version 1.0</a>
```

Chapter 6 - Testing

6.1 Functionality testing

Successive user feedbacks have been recorded as bugs for the sake of functionality errors and have been discussed in the further works section

6.2 Operation systems test

Three major operating systems have been used for testing and the results of them are as under.

Windows Xp with sp2

Tested by user: Dennis Smitt
Tested on: 21 August 2006
Which functionality did not work?
Did not work on website with frames.
Reported errors:
No errors reported

Apple Mac

Tested by user: paragmac
Tested on: 22 sept 2006
Which functionality did not work :
Did not work on website with frames.
Reported errors:
No errors reported

Linux (Ubuntu)

Tested by user: Anup Patil
Tested on: 25 September 2006
Which functionality did not work:
Did not work on website with frames.
Reported errors:
No errors reported

6.3 Testing matrix

Feature	WinXP	MAC	Unix
Installation	**OK**	**OK**	**OK**
Annotation insertion window	**OK**	**OK**	**OK**

Annotation reloading and viewing	**OK**	**OK**	**OK**
Annotation editing	**OK**	**OK**	**OK**
Annotation deletion	**OK**	**OK**	**OK**
Annotation count display	**OK**	**OK**	**OK**
Bug reporting	**OK**	**OK**	**OK**
Shared annotation search and adoption	**OK**	?	**OK**
Annotation modification	**OK**	**OK**	**OK**
Toolbar minimize	**OK**	**OK**	**OK**

Table 1 : Table for comparison of test cases on various OS

6.4 User feedbacks

The following user feedbacks were collected from users who have tested the beta version.

Feedback Number 1

Username: nmhatre
Real Name: Namrata Mhatre
Location: Mumbai, India
Tested on platform: WinXP

Positive feedback

- "I will use it whenever I browse the web"
- Really helps me to understand the same content faster when I read the content with my own annotations.

Negative feedback

- Colors are insufficient (only 4 of them)
- Cannot delete orphan annotations
- Install process is too long.

Feedback Number 2

Username: vaibhav
Real Name: Vaibhav Joshi
Location: Pune, India
Tested on platform: WinXP

Positive feedback

- "Cool spark dude".
- A quick turn off and turn on button for the doodles on a page.

Negative feedback

- Popupid could not be set
- Popup hides behind some content sometime as on www.yahoo.com
- Insufficient help

Feedback Number 3

Username: pratibha
Real Name: Pratibha Bhatanglikar
Location: Pune, India
Tested on platform: WinXP

Positive feedback

- Good tool I will refer it to my colleagues

Negative feedback

- The popup is hidden and is not visible
- Does not work properly when I am not connected to the internet

Feedback Number 4

Username: n1cole
Real Name: Nicole Eiserloh
Location: Dublin, Ireland
Tested on platform: WinxP

Positive feedback

- It helps me remember things, I am using it as post it stickers on web pages and content

Negative feedback

- When will you make it for Internet Explorer

Chapter 7- Further Work

7.1 Solving Known problems

7.1.1. The annotation engine fails on pages which are composed of frames.

WebPages where the actual view area is divided into frames and the content of each frame is loaded from a different location or is a different url. The annotation engine fails in these cases because of the following reasons.

- The content of the webpage is just a reference to different urls that are loaded inside one main page.
- Thus the content of the frames is not actually the content of the base page.
- Now when the user inserts an annotation on some text in the frames then he has actually inserted an annotation on the content on web content whose urls is not the same as the base page url.
- Thus on a reload when the annotation engine is triggered on the event of DOM content load end then it does not find the selected text and the anchor text because that content is not part of the base page.

The solution to this problem is as follows

- The annotation engine must be redesigned to act differently in case of a page with frames
- If frames are present then the annotation engine must save the url of the content page that is the base page and also the url of the content that is loaded in the frames. Especially of the frame from which the user has selected the text.

7.1.2. Span problem

In webpages that have a css (cascading style sheet) and in which a span is displayed with the rule **display:block** in these cases the display or the reloaded display of the annotations fails. It does not fail totally but the display looks like the following diagram

Figure 50 : Span problem : Before annotation insertion

Figure 51 : Span problem : after insertion of annotation

In the above example of www.yahoo.com you can see that the text gets broken and the annotation pop up does not appear at the correct position.

7.1.3. Multi load pages

WebPages which have a number of tags like **div** and **span** whose content comes from another remote location other than the location from where the base page is loaded, triggers the **onDOMcontentloaded** signal a number of times to the browser.

Now the annotation engine has a hook on the **onDOMcontentloaded** signal of the browser. The reloading of the annotations is triggered on this signal. In this case where there are multiple divs which have their content loaded from remote locations would trigger multiple of these signals prompting the annotation engine to be triggered number of times and thus falsifying the count of the orphan annotations and the loaded annotations.

Example of such a website is

http://timesofindia.indiatimes.com/?

The code that makes the onDOMcontentloaded signal to retrigger is as follows

```
<div >
<a href="/articlelist/424458.cms?toi_leftnav">City</a>
</div>
```

Figure 52 : Div with content being loaded remotely.

In the above example we see that the content of the div tag is loaded from another URL

The following figure shows the falsified count of the orphan annotations and the loaded annotations. (there is actually only one orphan annotation on this webpage but because of the multiple load event triggers going off this count is falsified)

Figure 53 : False count of orphan annotations and real loaded annotations

7.1.4. Web addresses interpretation.

Currently the annotation engine interprets the address **http://walimbe.com** and **http://www.walimbe.com** as two different web addresses. Due to the variable interpretation the retrieval and reloading of the annotations fails. If annotations are loaded on a page with the web address **http://walimbe.com** then this address is stored in the database and annotations will be loaded only when this address is entered in the browser. Now if **http://www.walimbe.com** is loaded into the browser the address is interpreted as it is and the annotations are not retrieved. This is the problem because the two web addresses lead to the same web content being loaded in the browser window but the annotations don't load because of the different addresses.

7.1.6 Tabbed browsing

The annotation engine has a bug that makes the count of the annotations loaded and the count of the orphan annotations to fail in case of multiple tabs opened and loaded with different web pages and their respective annotations.

The annotation engine is a single entity for the browser and for all the tabs inside it. When one webpage is opened in one tab of the browser then the annotations related to it are loaded on the webpage. In this case the count of loaded and orphans annotations is shown correctly. But when another tab is opened and another webpage is loaded inside it then the annotations for the webpage are retrieved and loaded inside it but now the count of the loaded annotations and orphan annotations is falsified.

This happens because when the annotation engine loads annotations it starts by first setting the count of the loaded and orphan annotations to zero and then later counting the number of annotations loaded and orphaned and set the respective figures.

Solution to this would be to have a separate array which would remember the counts of loaded and orphaned annotations for each tab. Full account for tab context must be achieved.

7.2 Removing reported bugs

7.2.1. Sometimes the pop up id is not set

Users while testing the annotation engine at their computers reported that the pop up id has not been set. The pop up id is a variable that is set inside the browser preferences. This is used later on to reference the annotations and load them. The pop up id could not be set due to the following reasons

- security restrictions that prevented the pop up id to be set

- server returned error or did not accept request or returned wrong popupid

The solution to this is that either the user manually sets the pop up id in a given field or there must be implemented a way that would dynamically generate a popupid and would not need it to be set in the user preferences of the browser.

7.2.2. Server Error.

At some time of the operation the server reports the following error. This happens when the annotation engine client sends a request to the server and the server does not respond for a long time. The exact reason for this bug is not know yet but this might happen because of the shared nature of the hosting server. The hosting server being shared, accept only specific number of requests per unit time and at other times it does not serve the requests resulting in the following error show in the figure.

Figure 54 : Server returns this error once a while

Solution seems to be change to a dedicated managed server.

7.3 Implementing unimplemented features.

7.3.1 Making user groups

User groups could be made for users to share their annotations to. This helps in collaborative learning. Currently the annotations can be shared globally (all users of the network can access it) or can be kept private (accessible only to the author of the annotation). This means that anybody who is part of the network can access the shared annotations. But some users like to share the annotations only to a group of friends. Thus a group making functions needs to be implemented and the annotations need to be shared to a group.

7.3.2 Adding user as friend

A user can make user groups to share his annotations to. He can make these user groups out of his friends list. Currently user's friends are the ones whom he has invited into the network. Thus the user must have an option where he can search a user by name or id and then add him as a friend to his friend list. This will enable him to make user groups out of his friends list.

7.3.3 Dublin core Meta data.

Meta data helps provide some information about a webpage. The insertion and usage of Dublin code Meta data is optional as of now. The author of the webpage can either opt to use part or none or the entire Dublin core Meta data. In cases where the author has not filled in the Dublin core meta data but the reader of the webpage knows some information about the meta data and wants to fill in the same then he must be given an option to fill in the Dublin core meta data for the web page. This meta data will be an augmentation to the page and wont be actually a part of the page. This meta data can be later shared with other users or can be kept private.

7.3.4 Meta data based search engine

A webpage will have the following types of Meta data when the annotation engine is running

- Annotation Meta data

- User Meta data

- Dublin core Meta data

- Derived or interpreted Meta data (e.g. popularity of the shared annotation)

Now all these Meta data can be used to better index and search pages. Thus an search engine needs to be built that will search through the Meta data of the annotation.

An another approach would be to make annotation and their meta data as entities that can be crawled by web crawlers.

7.3.5 Remote evaluation

The meta data search engine mentioned above would work well if lots of meta data is available. This includes the derived or interpreted meta data. This kind of meta data can come from remote evaluation. Thus remote evaluation must be done to find out the following meta data for each annotation and in general

- Popularity of the annotation
- Overall rank of the annotation
- Group size of a group
- Number of views for an annotation
- Webpage with most number of annotations
- User with most number of shared annotations
- Ratio of shared annotations to private annotations

etc

7.3.6 Local Storage

Currently the storage is centralized server storage. A scheme of local storage and retrieval needs to be devised. This will enable the storage of the annotations on the local file system

of the user. This involves local file system security issues as well as the browser security issues. A scheme for decentralized servers must also be worked on.

7.4 Extending support for different browsers.

7.4.1 Internet Explorer

A toolbar for Internet explorer is made using VC++ and CWindowImpl ATL Object Wizard. A detailed tutorial about how to make a toolbar would be found at the following reference [42] (Internet Explorer Toolbar (Deskband) Tutorial)

This is just one of the tutorials available on the internet. The back ground functionality stays almost the same with a few minor modifications. What is to be changed is the front end that has to be rewritten for internet explorer because internet explorer does not understand XUL at the moment.

7.4.2 Opera and other browsers

No information could be found in the area of extending the functionality of other browsers thus it has not been stated here.

Chapter 8 - Conclusion

The title of the thesis is “Doodle – Remote Tagging, Storage, Retrieval and Display of User Annotations within a Browser Environment.” The main objectives at the start of the thesis were as follows

- To make a data structure that can comprehensively define and hold the annotations in terms of addressing, storage and user properties.
- To make a system or method by which users of the system or method can select some text loaded in the browser environment.
- With reference to the selected text users can enter some annotation text which is to be stored in the storage with an association to the selected text.
- The users must be then able to retrieve the annotation text on demand and view it with reference to the selected text at some later stage.
- This viewing of the annotation text must be done by mixing the content of the annotation text with the original text.
- The users must be able to share their annotations with other users for them to view them.
- The users must be able to view and adopt the shared annotations of other users.

At the end of the thesis the following things are achieved

- A toolbar is made for the firefox browser which is a system or method by which users can insert, store and see annotations.

- A data structure that comprehensively defines, stores annotations is devised. (While designing this data structure the Annotea annotations schema was a very close match, But the schema did not have any possibility for user accounting, display properties of the annotation, rights management and type of annotation. Thus a new data structure has been defined.)

- The user can select some text that is loaded in the browser and with reference to the text insert some annotation text (with optional properties to share the annotation, change the color in which the selected text will appear once the annotations are loaded) (insertion of annotations)

- Once the user has inserted the text the toolbar in the backend stores the annotation content to the server. (Storage of annotations)

- When the user loads the same content again in the browser then the toolbar retrieves the user annotations from the server (retrieval of annotations)

- With the retrieved annotations the toolbar inserts the annotation in the original text that is there in the browser (remote tagging)

Extra features that have been implemented

- Password protecting the annotations by the necessity to create user account.
- Sharing of annotations.
- Setting display properties (base color of anchor text, type of annotation inline or popup)

- Adopting annotations that are shared by other users.
- Minimizing and maximizing the toolbar.
- Browsing to online account where the annotations can be managed.
- Forming an online community to share and adopt annotations. (Incomplete till date)

Further work involves solving the reported errors, implementing unimplemented features, making server software that is atomic and easily installable, defining and making public the api to the server so that custom made client software is able access the annotations, Decentralization of the server for faster access and security.

Finally I can say that as the title of the thesis is “Doodle – Remote Tagging, Storage, Retrieval and Display of User Annotations within a Browser Environment” which has 4 important aspects as

- Remote tagging of user annotation within a browser environment.
- Storage of user annotations within a browser environment
- Retrieval of user annotations within a browser environment
- Display of user annotations within a browser environment

Have been fulfilled by this thesis. Uncompleted work has been mentioned in the further works section.

References

[01] Ninth Graders' Use of a Shared Database in an Internet Research Project: Issues of Collaboration and Knowledge-Building
Authors: Jeff Kupperman, Raven Wallace, and Nathan Bos
Research done at : School of Education, University of Michigan
Resource link : http://www.oise.utoronto.ca/cscl/papers/kupperman.pdf
Author contact jkupp@umich.edu,ravenmw@umich.edu,serp@umich.edu
Viewed on: 4th July 2006

[02]. Structured Cooperative Authoring on the World Wide Web
Authors: Dominique.Decouchant, Vincent.Quint, Manuel.Romero.Salcedo
Resource link : http://www.w3.org/Conferences/WWW4/Papers/91/
Author contact: Dominique.Decouchant@imag.fr, Vincent.Quint@inria.fr, Manuel.Romero@imag.fr
Viewed on 5th July 2006

[03]. Interoperability of Peer-To-Peer File Sharing Protocols
Authors: Siu Man Lui and Sai Ho Kwok
Published in: ACM SIGecom Exchanges, Vol. 3, No. 3, August 2002,- page 27
Resource link: http://www.acm.org/sigs/sigecom/exchanges/volume_3/3.3-Lui.pdf
Viewed on: 2nd August 2006

[04]. rel-tag-Draft Specification 2005-01-10
Author: Tantek Çelik
Resource link: http://microformats.org/wiki/rel-tag
Viewed on: 2nd August 2006

[05] HTML
Resource: http://www.w3.org/MarkUp/
Viewed on: 3th August 2006

[06] Robust Annotation Positioning in Digital Documents
Author: A.J. Bernheim Brush,David Bargeron, Anoop Gupta, and JJ Cadiz
Collaboration and Multimedia Systems Group,Microsoft Research,One Microsoft Way,Redmond, WA 98052
Resource: Conference on Human Factors in Computing Systems. Proceedings of the SIGCHI conference on Human factors in computing systems.Seattle, Washington, United States ,Pages: 285 – 292, Year of Publication: 2001, ISBN:1-58113-327-8
Author contact: ajb@cs.washington.edu, {davemb, anoop, jjcadiz}@microsoft.com
Viewed on: 2th August 2006

[07] Annotations
Author: Robert Wilensky,
Resource link:
http://www.cs.berkeley.edu/~wilensky/CS294/lectures/Annotations.pdf
Viewed on: 2th August 2006

[08] Toward an ecology of hypertext annotation
Author: Catherine C.Marshall
Publisher: ACM Press, New York, NY, USA
Resource: Conference on Hypertext and Hypermedia Proceedings of the ninth ACM conference on Hypertext and hypermedia : links, objects, time and space---structure in hypermedia systems: links, objects, time and space---structure in hypermedia systems Pittsburgh, Pennsylvania, United States ,Pages: 40 - 49 ,Year of Publication: 1998 ,ISBN:0-89791-972-6
Author contact : marshall@parc.xerox.com
Viewed on: 2th August 2006

[09] Browser Statistics Month by Month
Resource link: http://www.w3schools.com/browsers/browsers_stats.asp
Viewed on: 20th September 2006

[10] Understanding Metadata
Authors : Rebecca Guenther and Jacqueline Radebaugh
Publisher :NISO Press,National Information Standards Organization,4733 Bethesda Avenue, Suite 300,Bethesda, MD 20814 USA,ISBN 1-880124-62-9
Resource link:
http://www.niso.org/standards/resources/UnderstandingMetadata.pdf
Viewed on: 2th August 2006

[11] XHTML Metainformation Module
Section 13.1.1. meta and search engines
Resource link:
http://www.w3.org/TR/2003/WD-xhtml2-20030506/mod-meta.html
Viewed on: 6th July 2006

[12] User Guidelines for Dublin Core Creation
Version 1.0 1997-05-29
Resource link:
http://www.sics.se/~preben/DC/DC_guide.html
Viewed on: 7th July 2006

[13] Dublin Core Element Set, Version 1
Resource link:
http://dublincore.org/documents/1998/09/dces/#
Viewed on: 5th July 2006

[14] W3C Document Object Model
Resource link:
http://www.w3.org/DOM/
Viewed on: 5th October 2006

[15] The XMLHttpRequest Object
Resource link:
http://www.w3.org/DOM/
Viewed on: 5th October 2006

[16] The DOM Inspector
Resource link:
http://www.mozilla.org/projects/inspector/
Viewed on: 8th October 2006

[17] A Server-mediated Peer-to-peer System
Author: Kwok, S. H, Chan, K. Y.and Cheung, Y. M.
Publisher: ACM Press, New York, NY, USA
Resource: ACM SIGecom Exchanges, Vol. 5, No. 3, April 2005.
Resource link: http://www.acm.org/sigs/sigecom/exchanges/volume_5/5.3-Kwok.pdf
Viewed on: 10th October 2006

[18] Search and replication in unstructured peer-to-peer networks.
Author: Lv, Q., Cao, P., Cohen, E., Li, K. and Shenker, S
Publisher: In Conference Proceedings of the 2002 International Conference on SUPERCOMPUTING. ACM. 2002, (2002), 84-95.
Resource link:
http://coblitz.codeen.org:3125/citeseer.ist.psu.edu/cache/papers/cs/29679/http:zSzzSzwww.cs.princeton.eduzSz~qlvzSzdownloadzSzsearchp2p_ics02.pdf/lv01search.pdf
Viewed on: 10th October 2006

[19] Marginalia Web Annotation
Author: Geof Glass.
Resource link:
http://www.geof.net/code/annotation
Viewed on: 10th October 2006

[20] E-Marked
Author: HUYREKA Solution Inc.
Resource link:
http://www.e-marked.com
Viewed on: 10th October 2006

[21] iMarkUp-PlugInUserGuide
Author: iMarkup Solutions, Inc.
Resource link:
http://www.imarkup.com/docs/PlugInUserGuide.pdf
Viewed on: 12th October 2006

[22] Web Annotator
Author: Dale Reed , Sam John
Publisher:
ACM Press New York, NY, USA
Technical Symposium on Computer Science Education
Proceedings of the 34th SIGCSE technical symposium on Computer science education
Reno, Navada, USA
SESSION: Using the web , Pages: 386 - 390 ,Year of Publication: 2003 , ISBN:1-58113-648-X
Resource link:
http://logos.cs.uic.edu/Annotator/p70reed.pdf
Viewed on: 12th October 2006

[23] Moodle - A Free, Open CMS
Author: moodle.org
Resource link:
http://moodle.org/
Viewed on: 15th October 2006

[24] Annotation for Moodle
Author: Geoffrey Glass dated 30 Jan 2006
Resource link:
http://molokai.ol.mala.bc.ca/moodle_doc/documents/annotation_tool_report_2006-01-30.pdf
Viewed on: 15th October 2006

[25] Cornell University Website
Resource link:
www.cornell.edu/
Viewed on: 15th October 2006

[26] About Hypernews
Resource link:
http://www.hypernews.org/HyperNews/get/hypernews.html
Viewed on: 6th October 2006

[27] Tools for collaborative learning
Resource link:
http://www.cs.cornell.edu/~dph/annotation%5Cannotations.html
Viewed on: 5th October 2006

[28] Domain Adaptive Information Extraction From Text
Author: Robert Arens
Resource link:
http://www.cs.uiowa.edu/~rarens/pubs/CompsPaperFinal.pdf
Viewed on: 5th October 2006

[29] Amilcare – Adaptive IE tool
Author: Fabio Ciravegna, Department of Computer Science, University of Sheffield
Resource link:
http://nlp.shef.ac.uk/amilcare/
Viewed on: 11th October 2006

[30] Melita- Overview
Author: University of Sheffield, Department of Computer Science
Resource link:
http://nlp.shef.ac.uk/melita/
Viewed on: 11th October 2006

[31] Annotator Instructions
Author: Wendy Seltzer
Resource link:
http://cyber.law.harvard.edu/cite/instructions.html

Viewed on: 5th October 2006

[32] Connotea Website
Resource link:
http://www.connotea.org/
Viewed on: 1st October 2006

[33] Ruby Annotations
Resource link:
http://www.w3.org/TR/ruby/
Viewed on: 1st September 2006

[34] Vibrant Media Website
Resource link:
http://www.vibrantmedia.com/
Viewed on: 1st September 2006

[35] Annotea project homepage
Resource link:
http://www.w3.org/2001/Annotea/
Viewed on: 11th September 2006

[36] Annozilla project homepage
Resource link:
http://annozilla.mozdev.org/
Viewed on: 12th September 2006

[37] Resource Description Format - RDF
Resource link:
http://www.w3.org/RDF/
Viewed on: 13th September 2006

[38] XML Pointer Language (XPointer) Version 1.0
Resource link:
http://www.w3.org/TR/WD-xptr
Viewed on: 13th September 2006

[39] Interface Reference - nsIXMLHttpRequest
Resource link:
http://www.xulplanet.com/references/xpcomref/ifaces/nsIXMLHttpRequest.html
Viewed on: 14th September 2006

[40] XML Path Language (XPATH)
Resource link:
http://www.w3.org/TR/xpath
Viewed on: 14th September 2006

[41] Yet Another Web Annotation System (YAWAS)
Resource link:
http://www.fxpal.com/people/denoue/yawas/
Viewed on: 14th September 2006

[42] Internet Explorer Toolbar (Deskband) Tutorial
Resource link:http://www.codeproject.com/atl/ietoolbartutorial.asp
Viewed on: 14th September 2006

[43] Annotation Schema
Resource link: http://www.w3.org/2000/10/annotation-ns#schemaProperties
Viewed on: 25th October 2006

[44] A Survey of Web Annotation Systems.
Resource link
http://www.math.grin.edu/~rebelsky/Blazers/Annotations/Summer1999/Papers/survey_paper.html
Viewed on: 25th October 2006

Appendix A – List of terms used.

Selected text

This is the text that the user selects in the browser environment and in association to which he will insert his annotation text. This is also called as the base text or the anchor text. The annotation text will appear in the vicinity or in association with this text.

Anchor text / Base text

This has the same meaning as the selected text.

Annotation

The user comments, opinions, views or some text that the user associates with the selected text along with all its properties is called the annotation (The properties are the display properties or the storage properties etc)

Annotation text

The text that the user actually enters in the annotation and associates it with the selected text is called annotation text.

Annotation engine

The set of codes that work together to allow the user to insert, delete, modify, view, store and retrieve annotations is called an annotation engine. With context to the implementation of the thesis it is the toolbar along with the server functionality

Toolbar

This is the part of the annotation engine which is installed in the firefox browser and which acts as the client part of the annotation engine.

Private annotation

This is a type of annotation which is only accessible to the owner of the annotation. This is the default type of annotation.

Shared annotation

This is a type of annotation which can be accessed by all the users of the network. It is not a default and it has to be set explicitly by the user as shared so that the other users of the network can access it.

Adoption of an annotation

This is the process by which the user of the network will adopt or copy a shared annotation. In case of an adoption the annotation is set into the account of the user and is accessible to him on demand. However the owner of the annotation is still the original user that inserted the annotation for the first time.

DOM

The DOM means the document object model. It is referred to in the thesis with context to addressing of the annotation.

Content / web content

This is the actual content that is loaded in the web browser. It could be a web page or could be even a text file.

Popup

This is the a type of display of the annotation. By a popup it is meant that when the user rolls his mouse pointer over the marked text he will be able to see a popup or a display that will hold the contents of the annotation that was inserted before.

Annotationid

This is an identifier to the annotation by which it will be referred in the annotation engine and the whole doodle network

Server

This is the part of the annotation engine which resides on the a location other than that of the user. It takes care of storage and retrieval of user annotations on demand by the client or the toolbar.

Client

This is the part of the annotation engine which is part of the browser in the case of the thesis or could be any other piece of software that uses the apis of the server to access the server to insert, delete, modify, display the annotations.

User

This is a registered user of the doodle network who has access to save his inserted annotations to the server.

Online account

By an online account it is meant the area of the user on the server where he can access and manage his annotations and do some other activities like sending requests to other friends to join the network.

Orphan annotation

Orphan annotation is an type of an annotation whose anchor text cannot be found at that instant of time either due to change of the original content or some other reason

Loaded annotation

This is a type of the annotation which has been successfully loaded in the browser environment in association to the anchor text.

Anchor text color

The color that the anchor text will be displayed in to indicate that the user has inserted some annotation at that anchor text.

Login name and password

These refer to the username and password of the doodle network.

Donor

In case of an adoption of a shared annotation the donor refers to the person or user to whom the shared annotation belongs to.

Doodle Network

This is the network that is formed by the connection of users of the registered toolbar users on the common server (in this case on doodle.walimbe.com)

Appendix B – Credits and acknowledgements

I would like to thank the following people for their help and contributions in my thesis work

Prof Thomas Schmidt

For guiding my thesis and giving me proper suggestions at required times.

Mr. Ravishankar Bhande

For designing the “Doodle” logo

Mr. Anup Patil

For help on Linux based issues.

Mrs. Nilambari Patil

For editing the logo for the toolbar

Declaration

I declare within the meaning of section 25(4) of the Examination and Study Regulations of the International Degree Course Information Engineering that : this Master report has been completed by myself independently without outside help and only the defined sources and study aids were used. Sections that reflect the thoughts or work of others are made known through the definition of sources

Hamburg 30th of October 2006 _____