Discovery Agent of Vijjana, a Pragmatic Model for Collaborative, Self-organizing, Domain Centric Knowledge Networks

Rajesh Makineni

Problem Report submitted to the
College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements for the degree of
Master of Science in Electrical Engineering

Approved by
Dr. Yenumula. V. Reddy, Ph.D., Chair
Dr. James D Mooney, Ph.D.
Dr. Sumitra Reddy, Ph.D.
Lane Department of Computer Science and Electrical Engineering Morgantown, West Virginia 2010

Keywords: Vijjana, Semantic Web, Knowledge Network, User Interface, Firefox Extension, Hibernate
ABSTRACT

Discovery Agent of Vijjana, a Pragmatic Model for Collaborative, Self-organizing, Domain Centric Knowledge Networks

Rajesh Makineni

With the evolution of the advanced web technologies, the World Wide Web is flooded with millions of giga bytes of information every day. In such a prodigious field of web along with its explosive growth, even the most successful search engines would fail to give the relevant results to the user based on the keyword based crawling algorithms they use. Even the next giant step in the web history which is adding the semantics to the web documents could not help the user in getting his desired results from the web. When users are given the chance to get closer to the people of their kind over the web through the so called social networking websites, they are lost in a vast collection of unorganized links without any domain specific taxonomy. In such a scenario, Vijjana [1] is designed to represent a collective knowledge network based on classification and analysis. Vijjana provides a framework designed for building collaborative knowledge networks that are domain specific and well bound in search categories.

In this report, we explain the Discovery Agent of Vijjana that creates a User Interface to the Vijjana network. The User Interface is designed in the form of a Mozilla Firefox extension which, when installed on the user's browser becomes a toolbar with various functionalities that access several agents of the Vijjana architecture. Through this interface, user is given the facility to directly transfer an interesting link, while browsing the web, to the Vijjana network for better organization. While doing this, the bookmarking service extracts the meta-data from the web page and stores it in vijjana database along with the saved URL. Another service namely the Search agent performs the search process in the Vijjana network and brings the URLs saved in different domains of Vijjana based on the input keyword given by the user. We also define a conceptual way of exporting the bookmarks stored in the Firefox browser to the Vijjana database on a single click through a well developed user interface exporting service using the Object Relational Mapping provided by the Hibernate framework.
Acknowledgements

I would like to thank Dr. Yenumula V. Reddy for his constant support and invaluable guidance in many aspects of my research. His encouragement and enthusiasm motivated me to work with the Vijjana team and share my innovative thoughts to implement them for the development of the Vijjana architecture.

I would like to extend my thanks to Dr. James Mooney and Dr. Sumitra Reddy for serving on the committee and supporting the project with their suggestions and encouragement.

I would like to thank the Vijjana team and all my friends especially Luyi Wang, Biswajith Yarlagadda and Vijay Sankar Yalamanchi for their immense support in various stages of my research project.

Finally, I am deeply thankful to my beloved parents and sister for their encouragement and support in all the things I do.
# Table of Contents

Acknowledgements ........................................................................................................ iii  
List of Figures .................................................................................................................... vi  

## CHAPTER 1: INTRODUCTION .................................................................................. 1  
1.1 Background .............................................................................................................. 1  
1.2 Problem Statement .................................................................................................... 2  

## CHAPTER 2: LITERATURE OVERVIEW .............................................................. 4  
2.1 Information Filtering Agent ...................................................................................... 4  
2.2 Search Together ........................................................................................................ 4  
2.3 WebTagger ................................................................................................................ 5  
2.4 Deepa Mehta Desktop .............................................................................................. 5  
2.5 Social Bookmarking .................................................................................................. 6  

## CHAPTER 3: VIJJANA MODEL ............................................................................. 7  
3.1 Model Specification ................................................................................................. 7  
3.2 VIJJANA AGENTS ................................................................................................. 8  
  3.2.1 Taxonomy (T) and Domain Specific Relations (R) .......................................... 8 
  3.2.2 The Discovery agent (dA) ................................................................................ 8 
  3.2.3 The Organizing Agent (oA) ............................................................................. 9  
  3.2.4 The Consistency/Completeness Agent (cA) .................................................... 9  
  3.2.5 The Visualization Agent (vA) .......................................................................... 10  
  3.2.6 The Search agent (sA) .................................................................................. 11  
  3.2.7 The Rating Agent (rA) .................................................................................. 11  
  3.2.8. The Collaborative Filtering Agent (fA) ........................................................ 12  

## CHAPTER 4: DISCOVERY AGENT & its USER INTERFACE ............................... 13  
4.1 Discovery Agent ....................................................................................................... 13  
4.2 User Interface .......................................................................................................... 14  
4.3 User Interface Architecture ..................................................................................... 15  
4.4 Vijjana Toolbar Extension ....................................................................................... 16  
  4.4.1 INSTALL.RDF ............................................................................................... 17  
  4.4.2 CHROME.MANIFEST .................................................................................... 18  
  4.4.3 OVERLAY.XUL ............................................................................................. 18  
4.5 Discovery Agent Services ....................................................................................... 20  
  4.5.1 Bookmarking service ....................................................................................... 20  
  4.5.2 Exporting Bookmarks ...................................................................................... 22
List of Figures

Figure 1: Radial Graph View ................................................................. 10
Figure 2: Proposed View of the Rating Agent ........................................... 11
Figure 3: Vijjana Toolbar view ............................................................... 14
Figure 4: User Interface Architecture ....................................................... 15
Figure 5: INSTALL.RDF file ............................................................... 17
Figure 6: CHROME.MANIFEST file ......................................................... 18
Figure 7: OVERLAY.XUL file ............................................................... 19
Figure 8: Vijjana Bookmarks Form ......................................................... 21
Figure 9: Structure of Vijjana Database ................................................. 27
Figure 10: Structured view of the OODBMS ........................................... 32
Figure 11: Impedance Mismatch Problem ............................................... 34
Figure 12: Model View of the Data Access Layer ................................... 35
Figure 13: Model View of the Data Access Layer ................................... 36
Figure 14: Data Access using EJB & JDBC ........................................... 38
Figure 15: Hibernate Architecture ......................................................... 39
Figure 16: Data Access using Hibernate framework ................................ 40
Figure 17: Hibernate Configuration File ............................................... 42
Figure 18: Hibernate Mapping File ......................................................... 43
Figure 19: Workflow Diagram of Vijjana Discovery Agent Implementation .... 46
CHAPTER 1: INTRODUCTION

1.1 Background

The enormous growth of information on World Wide Web has made the search engines to perform a gigantic or a trivial task in order to provide relevant information to the user based on the keyword based queries. As everyone knows, almost all the search engines use the web crawler (sometimes called as web spider) to navigate through all the hypertext links present in the World Wide Web and it stores all the web pages in a database giving an index to it. When the user enters a keyword in the search engine, it fetches the most relevant or matched web pages indexed in the database and displays it. Every search engine employs their best techniques of crawling, indexing and searching to keep pace with the changing technology of web contents but finally the success of all these search engines depends on how much relevant information has been given to the user. Since there are millions of web pages being added every day, can the search engines used today are efficient enough to assure the user to give the exact set of results he is looking for? Even though the search engines succeed in giving lots of links by taking a single keyword as a query, the major task of selecting the relevant links from the resultant set is still a herculean task for the user especially when the user is completely new to the context he is searching for.

When Tim Berners – Lee, director of the World Wide Web consortium (W3C) expressed his vision of changing the non semantic web (Web 1.0, Web 2.0) into a semantic web (Web 3.0), various web technologies and innovative concepts have been developed to link the data on the web. The term “Semantic Web” refers to W3C’s vision of the Web of linked data. [2] Semantic Web technologies enable people to create data stores on the Web, build vocabularies, and write rules for handling data. To what extent is the idea of semantic
web has been advanced is still an incredible question to be answered in this modern world where the World Wide Web is flooded with exabytes of information. Therefore every individual is looking for a better solution to access and retrieve information from the web that makes his job easy and comfortable.

In such a context, to extend the features of semantic web and refine the search process over web, our framework named Vijjana is designed. It is a pragmatic model for collaborative, self-organizing and domain centric knowledge networks. To achieve several goals of this project, various agents are designed in which Discovery agent is the most important one that makes this project accessible to the user on the web. This consists of an user interface designed in the form of a browser plug-in, which provides the access to the various agents of Vijjana architecture in the Business logic layer before mapping the objects to the Database tables in the Data Access Layer through the Object Relational Mapping provided by the hibernate framework.

1.2 Problem Statement

The motto behind designing Vijjana, a semantic network is to build a collaborative and useful knowledge domain in which every single user can bring individually identified links called as jans from a narrow specific domain to a useful semantic knowledge network that is useful to many others using the network. Such mechanism is done through the Discovery Agent of Vijjana, which is the most important and the initial step for any user in the Vijjana Network.

One way of discovering the jan is through browsing the web. Once the user gets a web page he wants to store in Vijjana, the Discovery agent (also termed as Markup agent) designed here, helps the user to transfer the web page details from the web browser to the Vijjana database. To accomplish this task, a Mozilla Firefox extension is developed
extensively for Vijjana project which includes a toolbar with several functions such as Bookmarking in Vijjana, Vijjana Home page, searching in Vijjana network, etc. Extension is created following the w3 standards of Mozilla 3.0.* browser specifications.

Bookmarking service designed in this process keeps track of the web page details such as the location of the web page, title of webpage and the description of the web pages along with keywords that are suggested by the source of the webpage and user’s own keywords until it saves in the Vijjana database for further organization of jans.

If the above described objective is one way of storing a jan in the Vijjana, one of the many other ways of populating the Vijjana database is to export the bookmarks stored in the browser of the local computer to the Vijjana network. To achieve this task we tried to access the bookmarks backup file stored by the Firefox browser and read all the details stored in the file before transferring it to the Vijjana database.
CHAPTER 2: LITERATURE OVERVIEW

2.1 Information Filtering Agent

Kwang Mong [3] brought up an enhanced way of filtering the URLs given by the search engines that assists users to pick from a selected set of useful links. The filtering agent is designed in such a way that it removes the out of context URLs given by the search engine according to the user’s query of keywords or phrases. Such type of filtering agents determine the relevance of the web pages through heuristic approaches like counting the nearness of the keywords in the web page, detecting the evidence phrases in the document, etc. Although the agent is successful in eliminating the irrelevant links given by the search engines based on the context, the technique lacks the domain centric ontology which is considered to be the most important specification of today’s search results.

2.2 Search Together

Meredith et al [4] came up with a new solution of collaborative searching by creating an interface for a group of users who want to perform searching on the web synchronously or asynchronously. Design of such an interface came with many advantages such as persistence, group searching, etc. It provides an integrated support to evaluate various search engine results used by several users and create an interactive environment to hold discussions between the users. According to the authors, this is designed specifically to support awareness, division of labor and persistence. The web links browsed through the “Search Together” are all given specific user ratings which are identified by several users to mark the relevance and the eccentricity of context of the search. Such a prototype is very useful in context of collaborative searching but it could be expanded in terms of taxonomy and ontology. This is specifically designed on the sole purpose of sharing the searched results but it lacks organization of knowledge gained through the web.
2.3 WebTagger

Richard. M. Keller et al [5] came up with a new bookmarking service for organizing and sharing URLs over the World Wide Web. The WebTagger is designed as an interface, which gives shared memory called as repository to the individuals to organize their URLs in various categories present in the system. Users are benefited by accessing their bookmarks anywhere in the world through this system. Unlike the bookmarking facility given by the browsers where there would be a predefined structure to save the bookmarks, WebTagger provides complete freedom to the user to create several categorical folders in which the bookmarks can be saved. All the created folders can be organized manually or they are automatically organized following a predefined strategy of ranking and augmented indexing. When it comes to the retrieval of the bookmarks saved in the repository, the user is given the choice of browsing all the folders in the repository in order to obtain his desired URL. Also the user can search based on the indexed categories that include various rating mechanisms, relevant feedbacks and other such inclusions. The proposed strategy comes as an advantage to the users who want to access their bookmarks anywhere on the internet and also provided with an organized hierarchical way to store one’s own links on a public shared memory over the internet. But the term public shared memory sounds odd in this modern world where multiple social networking websites are emerging day by day, that provide private spaces for every individual to organize their bookmarks in a more effective manner.

2.4 Deepa Mehta Desktop

In the year 2000, Jorg Richter and his team were successful in developing an open source semantic desktop application [6] which integrates all the work space applications into an integrated single knowledge network based on relational maps termed as topic maps. This application makes the user work with a collection of applications instead of working
separately with each and every application. This is made possible by connecting all the applications in a semantic look alike interface full of topic maps where each application is given an ID to be represented in the map. Every instance of usage of an application is considered to be performing in a different interface and the user has the freedom to switch between different applications. This is an older way of representation of the personal knowledge management where it is constrained to only one individual.

2.5 Social Bookmarking

Social Bookmarking is the term used to share, manage, organize and search the links or bookmarks or other resources over the web. After all these attempts were made to bring a semantic revolution in the World Wide Web, these days Social Bookmarking is one term that is constantly getting popular among the Web fanatics. Websites such as delicious.com [7], digg.com, diigo.com, twine.com[8] have already made their mark in this Social Bookmarking over the web. All these websites succeeded in storing the user links but failed to provide certain taxonomy, so that the user could have specific knowledge of the domain he is looking for. Hence to create a domain centric semantic web network, where a new user could search for the context irrespective of the knowledge he has on the domain, Vijjana is designed which organizes the web links stored by the user in a well defined domain specific manner.
CHAPTER 3: VIJJANA MODEL

3.1 Model Specification

The Vijjana model is an integration of agents and we define Vijjana as follows

Vijjana-X = \{J, T, R, dA, oA, cA, vA, sA, rA\}

Where

X = the domain name

J= the collection of JAN’s in the Vijjana-X

T = the Taxonomy used for classification of JAN’s

R= the domain specific relations

dA = the discovery agent which find relevant JAN’s

oA = the organizing agent which interlinks the JAN’s based on R

cA = the consistency/completeness agent

vA = the visualization agent

sA = the search agent

rA = the rating agent

fA = the collaborative filtering agent

Each agent is very important for the effective functioning of the Vijjana model and
more information about these agents is given in the remainder of this chapter.
3.2 VIJJANA AGENTS

3.2.1 Taxonomy (T) and Domain Specific Relations (R)

The initial step in developing any knowledge base is the classification of the Jans into various groups and getting a relationship between them to form a semantic net. We start with defining a taxonomy that suits the domain for which; the Vijjana knowledge base is developed. The word Taxonomy refers to “the practice and science of classification”. A wide variety of taxonomies are used to organize the knowledge. The Vijjana model is designed to work with any taxonomy and associated relational semantics by treating the taxonomies as parameters which can be modified with further development of the knowledge base, thus making Vijjana completely portable. We can also import and export any knowledge base having a taxonomy and relational semantics similar to that of Vijjana’s.

3.2.2 The Discovery agent (dA)

The Discovery Agent designed in the Vijjana model keeps track of the web page details such as the location of the web page, title of webpage and the description of the web pages along with keywords that are suggested by the bookmarking algorithm and user’s own keywords until it saves in the Vijjana database for further organization of jans(term synonymous to links). Discovery Agent is responsible for populating the database in various methods such as bookmarking a single page from the user browser, adding a single jan to the network manually, exporting the bookmarks from the Firefox browser, and automatically getting the jan information from RSS feeds. More details about the Discovery Agent could be discussed in the following chapters of this report.
3.2.3 The Organizing Agent (oA)

A Jan obtained through any one of the possible ways, either a RSS feed or Email or by clicking on the “bookmark” button installed in the browser by the user or through the Vijjana client interface, is sent to the organizing agent. The organizing agent first ensures that the Jan represents a genuine link, that is, the link is not broken or it is not submitted by any unreliable source. Then the bookmark information of the Jan that is used for the classification and interlinking is examined. Once the link is stored in the Vijjana database, it is meant to be organized in its respective domain following a pre-defined taxonomy using the Organizing Agent built in the Vijjana model. For example, if the user adds a link pertaining to the National Football League, then it is automatically stored with domain related paths such as Sports → Football → NFL followed by the URL.

3.2.4 The Consistency/Completeness Agent (cA)

The main function of the Consistency/Completeness Agent is to maintain the integrity of the Vijjana knowledge network. The cA does that by periodically visiting all the URL’s of the Jans once in a while to make sure that they are still “alive”. The consistency agent receives a HTTP status code 200(OK) which tells it that the URL is still “alive”. If it receives a HTTP status code 404(Not found), then it indicates that the URL is no more available on the server. The cA will also find out the relational links that are incomplete and which can be filled by the user and displays them in attractive colors. This helps in getting the attention of the user who can take necessary steps to fill out the incomplete links and correct the problem.
3.2.5 The Visualization Agent (vA)

The Visualization Agent (vA) is one of the major phases in the development of the Vijjana model. The main function of the Visualization Agent is to display the Vijjana Knowledge network in various forms depending on the preferences of the user. The Vijjana model uses three different views to visualize the data; they are Tree view visualization, Radial graph visualization and Hyper graph visualization. In all the three types of visualizations, the user is given the freedom of traversing through various nodes representing the domains of the Vijjana Ontological model.

Figure 1: Radial Graph View
3.2.6 The Search agent (sA)

The search agent helps the user by implementing various search operations based on the keyword input provided by the user. This feature is included in the model to cope up with various knowledge networks which the user is already familiar with but needs results quickly without browsing. For example, if the user inputs a keyword “cricket” in the search bar of the user interface, then the sA will return all the Jans that satisfy the stated criteria which may be the links saved by him or other users in the database.

3.2.7 The Rating Agent (rA)

The Rating Agent (rA) is another important feature of Vijjana, which grabs the attention of the users towards high value Jans. High value Jans are determined through a rating mechanism which is a combination of user specified values, relevance, number of hits etc. The user is given a user interface page to rate every Jan he saves into the database and he could also rate the jans saved by other users in the Vijjana network. Every user’s inputs regarding this rating is considered before giving the final automated rating to the jan in the database.

Figure 2: Proposed View of the Rating Agent
3.2.8. The Collaborative Filtering Agent (fA)

The Collaborative Filtering agent is a new agent added to the Vijjana network, which gives an edge over all other Social Networking websites. This is intended to suggest the user base on his choices of domains and his interest in various fields in the Vijjana network. Suppose a user saves a URL related to the Apple IPOD, the collaborative agent traverses the user’s history of adding jans in that specific field and comes with the suggestions of links from the user’s categories of elated domains, user’s friends domains based on the rating given to the links stored in the Vijjana database. In this way the user could come to know about other links in the Vijjana that are really useful for his IPOD and also the accessories which are best suited for his IPOD. Additional to that the collaborative filtering agent is intended to provide some more functionality which are still in the designing stage and need to be finalized before integrating to the final Vijjana Architecture.
CHAPTER 4: DISCOVERY AGENT & its USER INTERFACE

4.1 Discovery Agent

The Vijjana model is designed in an ideal way where every service and operation is built upon an agent paradigm sequence. The Discovery Agent in the Vijjana model is the most important and the initial step for the construction of Vijjana. This agent helps the user in adding the URLs to the Vijjana database. According to this agent, if the user finds any interesting URL while browsing the web and wants to store it in Vijjana, the Discovery Agent designed here provides an interface to add that link along with the pre processed meta-data of the web page to the Vijjana database. Once the link is stored in the Vijjana database, it is meant to be organized in its respective domain following a pre defined taxonomy using the Organizing Agent built in the Vijjana model. For Example if the user adds a link pertaining to the National Football League, then it is automatically stored with domain related paths such as Sports → Football → NFL followed by the URL.

Therefore to build such type of domain centric knowledge sharing networks, a web interface is designed for the Discovery Agent that is discussed in the next section of this report. It gives the user an easy way to add the links to the Vijjana database while browsing the web. Not only while browsing the web, but the user can have the privilege to export the bookmarks which he saved in the Mozilla Firefox Browser to our Vijjana database. This gives an added advantage to the user to get access to his bookmarks anywhere in the world. Therefore the user carries his bookmarks wherever he goes even without his personal computer. Also the discovery agent contains a searching service that takes the keyword as input from the user and runs an algorithm to perform the searching in the database for the URLs related to that keyword and presents it to the user in the client application.
4.2 User Interface

Our intention of creating a Firefox extension for Vijjana is to create an interface between the Web browsing sessions and the Vijjana’s collaborative knowledge sharing analysis. The extension designed here removes the human effort for the user to store every interesting link in Vijjana personally. The extension gives an easy method of accessing the Vijjana agents such as the Discovery Agent and the Search Agent. The extension is designed to install a new toolbar in the user’s Mozilla Firefox browser consisting of certain buttons like Vijjana, Search and Export bookmarks. All these Firefox buttons connect to the Vijjana server and henceforth act as an interface between the users browsing the web and the Vijjana network. Apart from the Vijjana toolbar, the extension also creates an option in the right click menu of the Firefox browser features list by the name “Bookmark This Page in Vijjana” which helps the user in moving a link from the browser to the Vijjana database. The extension when installed in the Firefox browser is as shown in figure 1.

![Vijjana Toolbar view](image)

Figure 3: Vijjana Toolbar view
4.3 User Interface Architecture

Any extension is designed intentionally to perform minimal tasks at the client side, whereas the major task of performance and the design goals of the extensions are made to be executed on the server side [9]. Similarly our extension is also designed to do the task of extracting the browser’s details along with the meta-data information of the current web page being browsed by the browser and transfer it to the server side where there are several applications designed for each specific task. User Interface architecture can be viewed in figure 2 below which represents the layered structure of the Vijjana user interface where the Vijjana toolbar represents the Presentation layer, Vijjana Architecture containing the Discovery Agent applications symbolizes the Business Logic Layer, whereas the Hibernate framework used to access the Vijjana database becomes the Data Access layer of the project.

![User Interface Architecture Diagram]

Figure 4: User Interface Architecture
4.4 Vijjana Toolbar Extension

Extensions are the added features for the client applications that could be integrated to the user’s web browser. Mozilla Firefox [10] web browser allows the users to install several client applications as long as there is no security breach in the configurations and settings of the normal functionalities of the browser. Hence Vijjana also took an initiative to give an extension package to the user which he could install in his Mozilla Firefox browser for the access of the Vijjana network. This extension will be an added attraction to the user who could feel the comfort in accessing the various agents of the Vijjana architecture from his browser during his browsing sessions.

Creating an extension that could work in all models of the Firefox browser requires the setting up an environment for the design of an extension according to the configuration settings built by the Mozilla Corporation [11]. In order to give a good design to the extension, the following structure need to be followed before starting to write the core code for the extension.

![File Structure]

The above structure of files is the base model for the design of any extension but nevertheless every extension needs all the files in the above model. Some of the important files written for the creation of this extension are discussed here.
4.4.1 INSTALL.RDF

This file contains the intimidating code of the Vijjana toolbar extension, which consists of all the meta-data information about the extension created here such as the versions this extension could support and other assorted information. The install.rdf file for the Vijjana toolbar extension looks like the one shown below.

```xml
<?xml version="1.0"?>
<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  <Description about="urn:mozilla:install-manifest">
    <em:id>SampleWebApplicationExtension@raj.com</em:id>
    <em:name>Vijjana Extension</em:name>
    <em:version>0.1</em:version>
    <em:description>Creates a toolbar for Vijjana</em:description>
    <em:creator>Makineni</em:creator>

    <em:targetApplication>
      <Description>
        <em:id>{ec8030f7-c20a-464f-9b0e-13a3a9e97384}</em:id>
        <em:minVersion>0.7</em:minVersion>
        <em:maxVersion>9.9</em:maxVersion>
      </Description>
    </em:targetApplication>
  </Description>
</RDF>
```

**Figure 5: INSTALL.RDF file**

As shown above the install.rdf file gives the details of the minimum and maximum versions of the browser that supports the browser, other details about the description of the extension and the intended purpose of creating this extension.
4.4.2 CHROME.MANIFEST

The scope of any Firefox application is within its content window such as the web browser toolbar, menus and the status bar. The chrome.manifest file created here gives the details about how the created extension will be added to the Firefox browser, and how it will work. This CHROME.MANIFEST file looks like the figure shown below.

```plaintext
content    vijjana    chrome/content/
content    vijjana    chrome/content/
contentaccessible=yes
overlay
chrome://browser/content/browser.xul
chrome://vijjana/content/overlay.xul
```

Figure 6: CHROME.MANIFEST file

As shown above the manifest file gives the path to the Firefox browser where it could find the content files created by the user and also the details of the file which the extension uses to override web browser elements, and add items to the toolbar, menu and status bar.

4.4.3 OVERLAY.XUL

XUL stands for the XML User Interface Language which is developed by the Mozilla Corporation for the developing of Extensions in Firefox, Thunderbird, etc. This is the file used by the designer to override some of the default look of the web browser, i.e. add a button to the toolbar, an item to the Tools menu and a status bar icon. Vijjana created this file to add an extra tool bar in the Firefox browser as part of the Vijjana User Interface. The file also contains the XML coding along with the JavaScript functions to access the vijjana’s agents from the browser.
The OVERLAY.XUL file created to form the extra tool bar in the Firefox browser can be shown in the figure below.

```
<toolbox id="navigator-toolbox">
  <toolbar id="VijjanaToolbar" toolbarname="Vijjana Toolbar" >
    <label value="Vijjana Toolbar: "/>
    <toolbarbutton id="Vijjana" label="Vijjana" oncommand="vijjanaClick()" />
    <textbox id="keyword" size="18" />
    <toolbarbutton id="Vijjana" label="search" oncommand="searchclick(event)" />
    <toolbarbutton id="Export" label="Export Bookmarks" oncommand="exportclick()" />
  </toolbar>
</toolbox>

<!-- This is for the right click menu. -->
<popup id="contentAreaContextMenu">
  <menuitem id="Vijjana Second Try" label="Bookmark this in Vijjana" accesskey="H" insertafter="context-stop" oncommand="rightclickprint();"/>
</popup>

<!-- This is for the Tools menu. -->
<menupopup id="menu_ToolsPopup">
  <menuitem insertafter="devToolsSeparator" label="Bookmark this in Vijjana" accesskey="H" oncommand="rightclickprint();"/>
</menupopup>
```

**Figure 7: OVERLAY.XUL file**

This file in Vijjana creates a new tool bar named Vijjana in the Firefox browser which looks like the figure 3. Each and every button created here is connected to a specific service designed in the Discovery agent that resides in the business logic layer and are hosted by the web server. All those service are discussed in the following sections.
4.5 Discovery Agent Services

From figure 4, the user interface architecture is divided into three layers i.e. the presentation layer, the business logic layer and the data access layer. Any extension is designed intentionally to perform minimal tasks at the client side, whereas the major task of performance and the design goals of the extensions are made to be executed on the server side. The business logic layer contains the backend architecture of the user interface which consists of various services designed to do specific task on the storing of links from the user interface to the vijjana database.

4.5.1 Bookmarking service

The bookmarking service is the first service designed in the Discovery agent configuration which starts with the processing of the meta-data information of the web page. Every web page is constructed using the HTML or the XHTML elements which describe the structure of the web page information. These elements are presented in the form of tags that could be read from the source of the web page. When the extension installed in the user’s browser picks the URL from the browser and gives to the extraction algorithm in the discovery agent, it makes an URL connection to the specified link to read the source code from the page. Once the HTML elements are read from the source code then all the tags are processed and a form is displayed to the user containing the extracted information such as the URL address of the page, title of the page and input fields for the user description and set of keywords that the user is intended to save along with the link in the Vijjana database. During the extraction of the meta-data, the extracting algorithm gives a set of keywords read from the source code of the web page, which are generally inserted in the source code by the web page creators which is a good practice of maintaining a website.
The Vijjana bookmarks form once installed on the right click menu of the Firefox browser looks like the one shown below.

![Vijjana Bookmarks Form](image)

**Figure 8: Vijjana Bookmarks Form**

As shown above, the user is intended to fill the empty text boxes in the form with the description of the website and the keywords column which he could fill from the suggested set of keywords or his own list of keywords. After the user submits the form, the details are transferred from the Firefox browser to the Bookmark application in the Discovery agent of the Business logic layer through a secured session bean connection. Once all the form data is processed and is ready to be stored in the database, the data access layer creates a mapping strategy to map the objects in the bookmarking service with the tables in the Vijjana database through the Object relational mapping provided by the hibernate framework which is discussed in the next chapter of this report.
4.5.2 Exporting Bookmarks

Another most important service being operated in the Discovery Agent is the Exporting Bookmarks service that functions as an extended interface between the Vijjana network and the Mozilla Firefox browser. Every user saves his bookmarks in the folder provided by the Firefox browser for his local reference. When he needs to access such locally stored bookmarks outside his home without his local computer, there comes the need of storing such bookmarks in a public space such as social networking websites or knowledge sharing domain networks like Vijjana. Hence we came with a solution to give the user the chance of exporting his bookmarks from the local computer to the Vijjana network on a single click of a button in the extension created as part of the User Interface. This makes the user feel comfortable in exporting any number of bookmarks on a single click without really storing the links individually in the Vijjana database.

This exporting service in the discovery agent brings the latest bookmark backup file stored by the Firefox browser for the extraction of the bookmark information. All the latest versions of Firefox browsers store the bookmarks in a specified format known as the JSON format which is termed as JavaScript Object Notation, a light weight data interchange format [12]. JSON is the subset of the JavaScript programming language and is extensively used for the transfer of data between various web sources. These days JSON is widely used for the communication between the client web page and the applications in the hosted server. JavaScript objects cannot be carried to the JSP page hosted on the server due to the isolation of the client side scripting and the server side scripting for a web application. JSON format introduces the new bridge between the client side and the server side, so that the objects in client side could be transferred literally to the server side programming through secured interchanging pattern.
According to the JSON format, all the bookmarks saved by the user in Firefox browser are organized as a collection of name/value pairs that follow common conventions used by all the Object Oriented Programming languages. The JSON format in which the bookmarks backup file is written by the Firefox browser looks like the set of statements shown below

```
```

Such a back up file written in JSON format contains the type of the bookmarks folder in which the bookmarks are stored and the meta-data information of the bookmarks such as the URL of the web page, title of the page, description and keywords explaining the page characteristics and also the dates of addition and modification to the respective bookmarks. When the user clicks on the exporting service in the Vijjana toolbar extension, the user is displayed the form to upload the latest bookmark backup file stored in the local computer. In future if the Vijjana team wants to integrate a local space for the links of vijjana in the user’s computer, then this service will be utilized to upload the local file of bookmarks upon the user’s request of exporting all the saved bookmarks to the Vijjana network.
As the exporting service takes the JSON file from the Firefox browser, it is processed to give a JavaScript object to the client side application. From the JavaScript object various fields such as the web address, web title, description and keywords are extracted to a collection/associated array. Then the extracted bookmarks are displayed to the user in a web page where check boxes are kept against each of the bookmark extracted. Now user is given the chance to select the bookmarks from the list provided on the web page that are to be transferred to the Vijjana database. User doesn’t require transferring all the bookmarks to the vijjana database but before transferring it, the exporting service leaves that decision to the user itself. Once the user selects the bookmarks in the list and clicks the submit button, the selected bookmarks are transferred to the exporting service in the business logic layer of the Vijjana architecture which stores the collected bookmarks in the Vijjana database using the Object Relational mapping provided by the Hibernate framework.

### 4.5.3 Searching Bookmarks

This is the service written in the Discovery agent, which resembles that of the Search agent in the Vijjana model. The user interface designed as part of the Discovery agent consists of a search field in the Vijjana toolbar extension. When a user enters a keyword in this search field, the extension connects to this Searching bookmarks service which searches the Vijjana database for the bookmarks containing the input keyword through the HQL search strategy. Then the agent displays the bookmarks saved by the user and also the bookmarks saved by other users in the Vijjana network containing the given keyword. This service performs the minimal task of searching all the bookmarks as part of the user’s extension in the Firefox browser and the real search agent being developed in the Vijjana model is intended to perform high scale prolific search mechanisms pertaining to the domain strategies implemented in the Vijjana architecture.
CHAPTER 5: HIBERNATE & OBJECT RELATIONAL MAPPING

5.1 Introduction

The Vijjana database is aimed to store large chunks of information links in a well organized manner and all the URLs stored in the database need to be linked in a semantic way that could enhance the search domain strategy being developed in the Vijjana project. Hence relational database became our predominant choice of storing the information and retrieving the records using SQL queries. But the general concern for every programmer working with different agents in Vijjana is that they prefer to work with persistent data instead of using SQL queries to access the data in the relational database. Many of the Vijjana’s working agents use the Object Oriented Programming Structures in their design and while connecting to the database it is rather a solitary strategy to use the Structured Queried Language. Hence we looked upon a mapping strategy that would bridge the gap between the programmer’s objects and the relational database structures. We found Hibernate as an efficient framework that provides the Object Relational mapping between the database tables and the application objects.

5.2 Database

Databases are an integrated collection of records that are designed to offer an organized mechanism of storing, managing and retrieving information. Vijjana is set to maintain a huge database in this aspect to collect a large domain of objects especially from the User point of view. Vijjana requires a database that could accommodate various types of data including the text, videos, pictures, audio and mixed media. To maintain specific domains and keep all the information in an organized manner, the whole Vijjana database is constituted into 3 types of data namely the raw data, User Data and the Website data. At this
moment the raw data for this project is taken from the Open Directory Project [13] that provides millions of records classified into various categories following its own classification analysis. The Open Directory Project is also an open source framework which is available to the users over the web to populate with various data sources varied across several domains that gives an edge of developing the Vijjana project from the initiated step of classification and implement our own model of organizing the data in specific domain centric categories.

From the user’s point of view, the Vijjana database consists of the information of the user and his history of activities in Vijjana. It consists of the user details such as his login information, user security details, etc. Along with the user information, the Vijjana database is designed to store the information of the URL (we term it as Jan in Vijjana) saved by the User in his respective account. Every user is given an account in the Vijjana database where his share of uploaded jans are stored. As stated earlier, Vijjana is developed to create an organized knowledge domain of information; all the saved URLs are organized following a pre defined taxonomy for which the database is divided to update all those URLs under respective domain categories.

The Vijjana database presents the schema of its organized structure as shown in the figure below where various tables are intended for specific purposes.

1. The tables General, Identifier and Classification are related to the Collection of the JAN’s from the Open Directory Project.
2. The integral Properties of the JAN are given by the tables Educational, Technical, Rights.
3. The Status of the Jan is given by the Table Life Cycle.
4. The User Information is given in the table Membership
5. The relation between the user and JAN is given by the tables Comment, CommentRating, JanRating, Keyword
Figure 9: Structure of Vijjana Database
5.3 Database Management System

Once the Vijjana database is implemented, it is the responsibility of the data access layer and the database management system to maintain the database and control the range of operations performed on the database. The selection of DBMS is considered to be the important task in the Discovery Agent of Vijjana that looks after the insertion, updating, and the deletion of records in the Vijjana database. There are multiple ways to deal with the databases in the form of relational database management system, object oriented database management system, hierarchical model and the network model. It is the part of the DBMS to connect the agents in the Business Logic Layer to the tables in the database. Hence we look for an efficient model that could enhance the settings of the database used in the Vijjana and be superior in accessing the tuples in the database. Following are the two types of Database Management systems that are extensively used for the web applications dealing with large databases.

5.3.1 Relational Database Management System

The Relational Database model is developed by Edgar F Codd at IBM to give a more defined structure to the ordinary databases used in the 1960s. It is a set of operations performed on the database tables to create relationships between the attributes of the tables in the database. The RDBMS could update the tables in the database by modifying, inserting and deleting depending upon the commands from the Business Logic layer. This type of management system is extensively used to maintain large databases for the web applications. Some of the popular Relational database models used are Microsoft SQL server, ORACLE, etc. All these systems generally use the Structured Query Language to access the tables in the database. Programmers feel the SQL queries to be easier than the regular programming code to pick the records from the database.
The main advantage of using the relational database management systems is its concept of following the ACID properties (Atomicity Consistency Isolation Durability) which gives an edge over other hierarchical models and the network models. According to the ACID properties, the RDBMS allows safe sharing of data and keep the database tables intact and secure from different kinds of requests. The Atomicity property of the RDBMS ensure that a complete update known as transaction is performed on the database avoiding any partially completed transactions. According to this property, there will be either a completed transaction or nothing is modified on a table in the database. The second property namely the Consistency ensures that any changes in the values of an instance would be consistent across all the instances of that object. Isolation is the third property by which the RDBMS is preferred well ahead of all other database management systems. This property avoids concurrent transactions being performed on the same records of a database. Whenever there are multiple requests for a same instance of the database table, the system sees that it is available to the requests in a serial procedure such that the transaction is completely committed before being allocated to the next request. The fourth property in the ACID is the Durability which ensures the recovery of the committed transactions and the updates performed on the database.

Vijjana team used this relational database management system to connect to the database for modifying and performing all other stuff from various agents in the Business Logic layer. People working on various agents such as the Visualization agent, Consistency agent used the JDBC connections and other means of connecting the database before performing the insertions, deletions and updates on the database tuples through the RDBMS. Since this model is being extensively used by many of the popular websites and organizations successfully for 3 decades, we have followed to use this previously.
Advantages of RDBMS

➢ It is simple, flexible and productive.

➢ It is a dominant model and extremely popular in the field of website databases and other server databases.

➢ Ease in extending the database by adding several categories without the original structure being changed.

➢ This model is successful in the past 30 years and we do have large quantities of data stored in this format already by various agents.

➢ Programmers feel comfortable in accessing the database through the SQL queries instead of the regular programming codes.

Disadvantages of RDBMS

Even though the RDBMS model is widely used and successful in maintaining a database, it had its own share of disadvantages which could be listed as follows

➢ Very hard to support saving of images, videos, audios and other media.

➢ Very slow in searching of large databases even though new methods to optimize the speed factor have come in the market.

➢ Inefficient in handling Object oriented languages such as Java, C# for the communication between objects and database.

➢ There is no specific way to access the database. Every agent operates in its own way to access the RDBMS which lacks integrity of the system.
5.3.2 Object Oriented Database Management System

The relational database management system used by the Vijjana developers is efficient enough to store the information in a simple and flexible format into the database and makes easily accessible to the programmers, but lacks object oriented structure followed by all the developers in developing various agents of Vijjana. When we try to map the objects we create in our programming language with the tables in the database, we come across the impedance mismatch problem which could be solved by seeking an alternative to the RDBMS. Such an alternative to the RDBMS is the Object Oriented Database Management System (OODBMS) which is a combination of the Object Oriented Programming concepts and the Database Management properties. All the concepts we use in the OOPS strategy such as the polymorphism, encapsulation and inheritance are integrated with the DBMS principles like the ACID (Atomicity Consistency Isolation Durability) properties.

The OODBMS is best known for its persistence nature of the object in the database. It accesses the objects in the database in a very transparent manner such that there is no need of an interface or a bridge between the accessing layer and the database. One of the important features of the OODBMS is that there is no need of any Structure queried language to control the objects in the database. According to this concept, the data is stored in the form of objects into the database which are more or less similar to the objects we create in our object oriented programming models. This feature of storing the data in object form gives an added advantage of using this system to store even the images, videos and other media information. When there is a need of indexing the database, the OODBMS provides a structural indexing where the objects and its instances follow the inheritance strategy of the classes and supports the hierarchy of objects stored in the database.
Figure below shows the structured view of the Object Oriented Database Management system, which features the database concepts connected with the Object oriented programming principles. It should be noted that there need not be any relationship model such as the entity mapping between the programming objects and the database objects since the programming objects are just the instances of the database model. Such type of a model is successful in integrating various types of objects in the database, but it needs to be shared by the large community for its growth since users still retain with the RDBMS in the web market.
Advantages of OODBMS

- Avoids impedance mismatching problem since there is no need of matching the objects to the database entities.
- Supports several complex types of data such as the images, audio, video and other media information.
- Can isolate the internal parts of the objet and access it separately performing all the operations internally without actually presenting it to the client application maintaining a cache layer.
- Follows the hierarchical model similar to the inheritance concept of the Object oriented programming which makes it easier in organizing the objects of the database.
- Allows the agent programs to search the database in multiple dimensions optimizing the speed of access and retrieval.

Disadvantages of OODBMS

- Lacks sufficient quality programmers in this field and occupies a meager share in the database market.
- The database objects are tied to the designed language at the type of creation and works only with that specific language concepts in the future use of the agents. There is no generic type of language to access the information from the database.
- Structure of the data in the database model is fixed and could not be modified such as adding new attributes in a class, deleting the original attributes in the class, etc.

The above mentioned disadvantages cost a lot to some programmers which made them stick to the RDBMS overseeing all the above advantages.
### 5.4 Object Relational Mapping

When we find it very hard to choose between the relational database management system and the object oriented database management system, we came with a new solution of mapping the objects in the business logic layer with the regular relational database tables through the Object Relational Mapping. As stated earlier Vijjana team works with different object oriented languages for the design of various agents of Vijjana architecture and when they all need to access the tables in the database, Object Relational Mapping could provide a common mapping strategy between the programming objects and the tuples in the database. Object/Relational Mapping (ORM) provides a methodology and mechanism for object-oriented systems to hold their long-term data safely in a database [14] which could be called as Persistence. This also avoids the famous impedance mismatch problem that arises when we use the regular SQL queries to access the database from the Object classes in the business logic layer. When we look for the persistence of the objects in the database, there always occurs a mismatch between the object model and the relational model of the database. Figure below shows the impedance mismatch problem when several instances of the programming object try to access the same attribute in the relational database.

![Impedance Mismatch Problem](image)

**Figure 11: Impedance Mismatch Problem**
The Object Relational Mapping bridges such types of mismatches through the OR mappings or the entity model mapping methods. Vijjana architecture supports two types of modeling at two different ends of the data access layer. The programming end of the data access layer follows the object model with all the types of relationships such as one to one, one to many and many to one mapping. On the other end of the data access layer there is the relational model consisting of the different types of tables in the database which support collective joins and mutative mapping relationships. It is the part of the data access layer to connect the object model and the relational model in order to access the database for all the applications in the business logic layer. Figure below shows the two types of models in the Vijjana architecture which needs to be mapped against each other. It should be noted that whenever we come with a new mapping strategy between these two models, the schemas of both the models need to be configured.

![Diagram of Model View of the Data Access Layer]

**Figure 12: Model View of the Data Access Layer**
The best strategy we got to map the object model with the relational data model is the Object Relational mapping [15] which acts as an intermediary between the object oriented code base and a relational database. While there are several styles of implementing the object relational mapping, we took the strategy of introducing the entity model in the middle of the two models we have, which overcomes the deficiencies of the relational databases such as providing the hierarchy between the tables, etc. The implementation of the entity model and its stand between the programmer’s object and the relational database could be seen in the figure below.

![Diagram of Object-Relational Mapping](image)

**Figure 13: Model View of the Data Access Layer**

Designing such object relation mapping along with the entity models encourages the programmers to concentrate on their object oriented programming since the model automatically looks after the operations that are to be performed on the database such as insertions, deletions and other modifications. Out of a wide variety of ORM tools [16], we have chosen the hibernate framework for the implementation of the OR mapping which we discuss in the later sections of this report.
5.5 Enterprise JavaBeans

In a web centric application such as Vijjana, the Business logic layer containing various agents of vijjana architecture needs to be highly scalable and consistent with the mapping relations to the database. Since most of the programming is done in Java, Enterprise JavaBeans is the best suited architecture for the development and deployment of large, robust and highly scalable business applications. EJB is successful in maintaining the connections to the database and controls all the operations performed between the business logic layer and the data access layer. Therefore the EJBs act as the middleware between the client applications and the database access to perform the transactions on the relational database tables. EJBs have an added advantage of generating its own business logic in its container using the reliable sources such as the J2EE connector, session bean, and JDBC driver connection. The EJBs need not be on the same machine where the database resides or the client applications are designed. EJBs can offer services to the remote clients as well by providing remote interfaces between the application programs and the container with all the entity beans that maintain the connections to the database.

As the Business Logic layer of the Vijjana architecture consists of various applications being written in several object oriented programming languages such as Java, several entity beans and session beans are used to deal with the client interface in the presentation layer and communicate with the data access layer. The proposed view of the Discovery agent of Vijjana using the EJB architecture to connect to the database is shown in the figure below which gives an estimated model of the EJB container acting as the middleware between the Discovery Agent’s various services and the Vijjana database.
From Figure 3, the EJB container maintains a pool of connections to the database through the JDBC driver. When an enterprise bean from the Discovery Agent requests a connection, the EJB container fetches from the pool and assigns to the bean for a single context. Once the database task for that bean is completed the container retains the connection and keeps it in the pool to share for other beans. Maintaining such a pool of connections is of course expensive and time consuming. Also when we use a JDBC driver connection to the database, we need to change every instance of the object to the relational format of the database. Hence every time the database is changed, the code for maintaining the JDBC connection need to be changed along with the SQL queries used to communicate with the database.
5.6 HIBERNATE

Hibernate is a powerful, high performance object/relational persistence and query service, which lets us develop persistent classes following object-oriented idiom - including association, inheritance, polymorphism, composition, and collections [17]. It comes up with a new query language HQL which looks similar to the SQL, to interact with the Hibernate persistent objects but nevertheless stops the user in using his regular SQL queries in accessing the database. Figure below shows the high level view of the Hibernate architecture, which shows the Hibernate giving persistent data object services to the built applications and providing mapping in XML between the programming objects and the relational database tables.

![Hibernate Architecture Diagram](image)

Figure 15: Hibernate Architecture
The user interface designed in this project requires a lot of communication between the Business Logic layer and the Data Access Layer where the final database resides. Generally we use Java in most parts of the Business Logic layer that works with a large set of classes and objects. Hibernate framework provides a solution to map the database tables to the Java classes so that the programmer could concentrate working on various agents and their business logic leaving the database tasks such as fetching and updating to the Hibernate framework. Hibernate proves to be the better solution than the usual entity beans or the direct Java Database Connection applet used to connect to the database due to its Object Relational Mapping and the Transparent Persistence. The data access layer between the various services of the Discovery agent and the Vijjana database when implemented through the Hibernate framework looks like the figure shown below.

![Figure 16: Data Access using Hibernate framework](image-url)
As figure shows, Hibernate emerged as a powerful alternative to the direct JDBC access by its entity relationship mapping strategy and the transparent persistence of the data storage. It replaced the pool of connections to the database with a single mapping strategy where each and every attribute in the database is mapped with respective variables in the persistent classes developed in the data access layer. It starts with the creation of the persistent data classes along with their entity mapping to the programming objects of the applications in the Business logic layer. Hibernate generates an automatic XML configuration file for the mapping of the database tables with the Java classes. Before mapping the tables directly, Hibernate framework provides a cache mechanism in two levels – first level by session object and second level by session factory object, that creates image of the database tables on which the insertion and retrieval is to be done. Therefore the HQL performs all the requested operations with the already loaded data in the cache and hits the original database only when it is required to be updated. This is done to optimize the performance of the Hibernate Query Language which reduces the number of hits on the database and increases the performance of the data access layer. Hibernate automatically provides the mapping strategy for each and every table in the database with the Java objects (POJO) which is termed as Transparent Persistence. This reduces the developer’s effort of writing a code to explicitly map database tables with the objects in the business logic layer. Apart from these the Hibernate framework could handle multiple requests from the business logic layer with the two levels of cache it keeps from the database and the entity models it organizes between the database tables and the Java objects. The process of configuring the database along with the driver connections, creation of the XML mapping are all discussed in the following sub sections.
5.6.1 Configuring the Database Connection

To set the required environment for the Hibernate to make a connection pool to the database, hibernate provides a configuration file which is written in the form of XML mapping structures. These days Netbeans and Eclipse IDEs are coming with an integrated framework of Hibernate 3.0 which provides a design template to automatically feed the details of the connection drivers and other configuration setup information. The file named hibernate.cfg.xml when filled with the details of Vijjana database server connections would like the figure shown below.

![Hibernate Configuration File](image)

**Figure 17: Hibernate Configuration File**

Figure shows the screen shot of the configuration file template created by the Netbeans IDE 6.5.1. Apart from the JDBC properties and the Data source properties shown above, the configuration file also contains other configuration properties such as the dialect property which tells hibernate the type of database it is going to connect from the hibernate framework.
5.6.2 Mapping Objects to Database Tables

Once hibernate makes connection with the database, it automatically searches the database and displays all the eligible relational tables which had atleast one primary key. The programmer could select the tables that need to be mapped with his objects in the business application. When the required tables are selected hibernate framework could automatically generate a mapping file reading all the attributes in the tables and creating mapping objects with the same name as the table columns. It should be noted that the mapping file can be created manually if the programmer feels to map only a certain number of attributes in the table instead of mapping the entire set of attributes. The mapping file generated by the hibernate framework for the Vijjana bookmarks table is shown below.

```xml
<hibernate-mapping>
  <class name="comorg.vijjana.hibernate2.Vijbookmarks" table="vijbookmarks"
catalog="vijjanatest">
    <id name="bmarkId" type="java.lang.Integer">
      <column name="bmark_id" />
      <generator class="identity" />
    </id>
    <property name="webAddresss" type="string">
      <column name="web_addresss" />
    </property>
    <property name="webTitle" type="string">
      <column name="web_title" />
    </property>
    <property name="description" type="string">
      <column name="description" />
    </property>
    <property name="lname" type="string">
      <column name="lname" />
    </property>
  </class>
</hibernate-mapping>
```

**Figure 18: Hibernate Mapping File**
5.6.3 Hibernate Query Language

Once Hibernate provides complete setup for the object relational mapping between the programmer’s objects and the database, there is a need of a language that could work well with the object oriented languages and retrieve the records from the database. Hibernate provides a strong querying language namely the Hibernate Query Language [18] which looks similar to the Structured Query language, which we have used for years to access the records in the database. Unlike SQL, Hibernate Query Language is object oriented and supports several features of the object oriented programming concepts like inheritance, polymorphism and association. This make easy access of the records in the database from the programmer’s point of view since they feel comfortable working with the objects.

Advantages of Hibernate

- Improves the productivity in case of implementing the data access layer by eliminating the tedious JDBC code.
- Simple and flexible with different types of objects in the application layer.
- Number of lines in the code for the access of objects in the database id decreased.
- Very easy to implement the modifications in either the database or the code in the data access layer.
- Provides the cache mechanism which decreases the number of hits to the database and increases its efficiency in both performance and speed.
CHAPTER 5: IMPLEMENTATION

The discovery agent of Vijjana is implemented in various stages following the Layered architecture of Vijjana. It consists of the Presentation layer which contains the Vijjana toolbar installed in the Firefox browser, the Business Logic layer which contains the Discovery Agent services and the Data Access Layer which contains the Object Relational Mapping to the Vijjana database.

First the User Interface to the Discovery Agent is designed in the form of a Firefox extension which when installed in the user’s browser creates a toolbar that consists of various buttons to access several services of the Discovery Agent.

Then the Bookmarking service in the Discovery Agent is designed to fetch the URL from the user’s browsing session and extract all the HTML meta-data processing the source code of the web page and display it to the user in a form. The user is given the chance to feed his details along with the extracted information before saving it to the Vijjana database.

Then the Exporting service is designed as part of the Discovery Agent to export the bookmarks saved by the user in his local computer. This service prompts the user to upload the latest bookmark backup file which is read in JSON and converted to JavaScript object to extract all the saved bookmarks in that file. The bookmarks are displayed to the user and asked to select a set that need to be exported to the Vijjana database.

The Searching service of Discovery Agent is designed to give the user, all the bookmarks saved in the Vijjana database based on the input keyword. In this service, user is asked to input a keyword and based on the input keyword, a HQL query is made to search the Vijjana database to display all the bookmarks resulted from the search query. All these services are connected to the Vijjana database using the Hibernate Object Relational Mapping.
Figure 19: Workflow Diagram of Vijjana Discovery Agent Implementation
CHAPTER 6: FUTURE WORK & CONCLUSIONS

We have proposed a designed User Interface that works as an extension in the Mozilla Firefox browser and acts as a Discovery Agent to Vijjana in providing URLs and related information to the Vijjana database. The interface installs a Vijjana toolbar in the Firefox browser which contains several features connected to the different services in the Discovery agent hosted at the Vijjana server. The bookmarking service provided the user to bookmark a page in Vijjana while browsing the web, where the backend of the bookmarking service extracts and refines the meta-data of the web page by making an URL connection from the server to the proposed web page. One of the other services designed here include the searching service that is incorporated in the Vijjana toolbar. This service is designed to take the user’s keyword as input and provide the search results from the Vijjana network. The user interface designed here succeeded in exporting the user’s bookmarks stored in the Firefox browser to the Vijjana network, which is domain centric knowledge sharing and self organizing network. The Discovery agent used the Object Relational Mapping provided by the Hibernate framework that came with an effective solution in bridging the gap between the presentation layer and the data access layer of our application.

In future we are looking for an enhanced way of creating a local Vijjana database in the user’s computer apart from the public space which every user gets in the server database, so that the user could save the interesting URL’s in his local space and can export it to the public space upon his interest. This requires a folder structure to be created in the Firefox Browser and a procedure to store the bookmarks in a compatible format such as JSON format that can support the regular Firefox bookmarks. Further work need to be done regarding the effective Batch processing strategy in the Hibernate framework and its locking methods.
The search service implemented here takes a single keyword at a time for searching the database. This service can be extended to incorporate high scalable searching algorithms, which could take multiple keywords or a sentence as input from the user and search the database for a more clear view of the knowledge domain network. Optimal algorithms to increase the speed of the search strategy can be designed to increase the performance of the search agent implemented here.

The bookmarking service is designed to add a URL along with its meta-data to the Vijjana database. Future work can be done on this service to add files, images and other types of media to the vijjana database from the web browser. This requires creation of APIs that connects the media files uploaded on various websites and the applications that process the media files to store them in the Vijjana database.
REFERENCES


13. Corporation, Netscape Communications. About the Open Directory Project. [dmoz.](http://www.dmoz.org/about.html)


Outline

- Vijjana Architecture
- User Interface
- Discovery Agent Services
- Hibernate & Object Relational Mapping
**Vijjana Architecture**

**Discovery Agent**

- Initial step in constructing the Vijjana model
- Helps the user to add URLs to the Vijjana database
- Provides an interface to the user browsing the web to access the various agents of Vijjana
- Deals more with the meta-data information of the URLs being browsed by the user
Creating Firefox extensions

- Firefox extension is designed to create a toolbar and a feature in the right click menu of the browser.
- Designing a folder structure to implement the code for a extension

- Presented the complete extension in the form of a XPI file that needs to be installed in the user’s browser.
Vijjana Toolbar View

Discovery Agent Services

Forms the business logic layer of the Vijjana architecture hosted in the Vijjana web server
- Bookmarking Service
- Searching Service
- Exporting Bookmarks service
Bookmarking Service

- User browsing the web initiates the service from the right click menu of the Firefox browser
- Takes the URL as input from the extension created in the browser.
- Makes a URL connection from the server to the given link and reads the source code from the web page
- Information from the HTML tags in the source code is read and meta-data is constructed for the URL

Vijjana Bookmarks Form
Searching service

- Searches the bookmarks stored in the vijjana database based on the input keyword given by the user
- Performs HQL based searching on the database
- Returns the URLs saved by the user with the given keyword and also the URLs saved by other users in the network
- This is just an instance of the basic search agent which is intended to perform high scalable search mechanisms.

<table>
<thead>
<tr>
<th>Bookmark ID</th>
<th>Web Address</th>
<th>Website Name</th>
<th>Description</th>
<th>Saved by</th>
</tr>
</thead>
<tbody>
<tr>
<td>167</td>
<td><a href="http://www.schoolbooks.com/">http://www.schoolbooks.com/</a></td>
<td>W/vSchools Online Web Tutorials</td>
<td>W/ VU/MX EMAIL ACCOUNTS: rajesh</td>
<td></td>
</tr>
</tbody>
</table>

Vijjana has other users' results for the searched keyword. List:

<table>
<thead>
<tr>
<th>Bookmark ID</th>
<th>Web Address</th>
<th>Website Name</th>
<th>Description</th>
<th>Saved by</th>
</tr>
</thead>
<tbody>
<tr>
<td>231</td>
<td><a href="http://www.schoolbooks.com/">http://www.schoolbooks.com/</a></td>
<td>W/vSchools</td>
<td>All kinds of web tutorials</td>
<td>homba</td>
</tr>
<tr>
<td>232</td>
<td><a href="http://www.schoolbooks.com/">http://www.schoolbooks.com/</a></td>
<td>vSchools</td>
<td>All kinds of web tutorials</td>
<td>rajip</td>
</tr>
<tr>
<td>244</td>
<td><a href="http://www.schoolbooks.com/">http://www.schoolbooks.com/</a></td>
<td>HTML Tutorial</td>
<td>HTML tutorials at schoolbooks</td>
<td>sreotha</td>
</tr>
<tr>
<td>248</td>
<td><a href="http://www.tiny.com/htmlT">http://www.tiny.com/htmlT</a></td>
<td>HTML Tutorial - Introduction</td>
<td>web tutorials at tiny</td>
<td>sreotha</td>
</tr>
<tr>
<td>252</td>
<td><a href="http://www.htmlgeodes.com/">http://www.htmlgeodes.com/</a></td>
<td>HTML geodes</td>
<td>web tutorials</td>
<td>sreotha</td>
</tr>
<tr>
<td>258</td>
<td><a href="http://www.schoolbooks.com/">http://www.schoolbooks.com/</a></td>
<td>vSchools</td>
<td>All kinds of web tutorials</td>
<td>shgdo</td>
</tr>
<tr>
<td>261</td>
<td><a href="http://www.schoolbooks.com/">http://www.schoolbooks.com/</a></td>
<td>W/ Schoolbooks Online Web Tutorials</td>
<td>all kinds of web tutorials</td>
<td>shgdo</td>
</tr>
<tr>
<td>307</td>
<td><a href="http://www.schoolbooks.com/">http://www.schoolbooks.com/</a></td>
<td>W/ Schoolbooks Online Web Tutorials</td>
<td>all tutorials</td>
<td>shgdo</td>
</tr>
</tbody>
</table>
Exporting Bookmarks Service

- Every user saves his bookmarks in the Firefox browser for local reference
- Such bookmarks could be exported to the Vijjana network on a single click of a button through this exporting service
- User is asked to upload the latest bookmark backup file stored by the Firefox browser

JSON

- JSON – JavaScript Object Notation, a subset of the JavaScript programming language
- Collection of name/value pairs

```json
[{
"name":"bookmarkProperties\description","root":"toolbar folder","children": [{
  "index":1,
  "title":"mix",
  "uri":"http://mix.wvu.edu/"
},
```
JSON (contd...)

Bookmarks file written as a JSON string is refined and converted to a JavaScript object

```
Var myJSONObject=eval("Given JSON String");
```

Bookmarks will be extracted from the client side scripting through the JavaScript object

```
webadd[i]=myJSONObject.children[1].children[i].uri;
webtitle[i]=myJSONObject.children[1].children[i].title;
```
Database

Integrated collection of various types of data such as the user data, raw data, knowledge domain categories.

- User data consists of the login details, user’s history and user’s network in Vijjana
- Raw data consists of all the meta-data added to the Vijjana database by all the users in Vijjana
- Domain categories presents several fields of data that are already organized in the Vijjana database.

Database Management System

- Performs all the operations on the database and is responsible in connecting the objects in the business logic layer to the database tables.
- Several types of Database Management systems
  - Hierarchical Model
  - Network Model
  - Relational Database Management System
  - Object Oriented Database Management System
Relational Database Management System

- Developed by E. F. Codd to give a more defined structure to the databases.
- Uses SQL queries to access the database tables.
- First of its kind to satisfy the ACID properties (Atomicity, Consistency, Isolation, Durability).
- Dominant model in managing databases and occupies a large share in the web market.

Object Oriented Database Management System

- OODBMS – combination of object oriented programming concepts and the database management properties.
- Best known for creating a persistent database.
- No need of any Structured query language for the access of database.
- Data is stored in the format of objects in the database.
Structured View of OODBMS

Object Relational Mapping

- Proposes a common mapping strategy between the programmer’s objects and the tuples in the database.

- Bridges the mismatches between the object model and the relational model in the data access layer.
Model View of the Data Access Layer

Impedance Mismatch Problem
Mapping through the Entity Model

Data Access using EJB & JDBC
Hibernate Architecture

Data Access using Hibernate Framework
Configuring database connection

Hibernate Mapping
Hibernate Query Language

- HQL is used by Hibernate framework to access the mapped tables in the database
- Unlike SQL, HQL supports OOPS concepts such as inheritance, encapsulation and polymorphism.
- Improves the productivity in case of implementing the data access layer by eliminating the tedious JDBC code.

Conclusion & Future Work

- User Interface in the form of a Firefox extension is designed to bookmark an URL, access the searching service of Vijjana and export the bookmarks from the browser to the Vijjana Network
- Requires a local database to be installed in the user's computer.
- Search service needs to be extended to provide high scalable mechanisms
- An application has to be designed to add images, videos, etc to the Vijjana database.