

Dexter

Making search web 2.0 ready!

Submitted as a

Final Year Project

For the partial completion of

B. Tech in Computer Science and Engineering

By

Vaibhav Bajpai

Rahul Burman

Nupur Dixit

Yadavendra

Under the guidance of

Mr. Sanjeev Pippal

Associate Professor

Department of Computer Science and Engineering

Galgotias College of Engineering and Technology

Dexter

Making search web 2.0 ready!

CERTIFICATE OF ORIGINALITY

This to certify that the project entitled 'Dexter' has been completed successfully in Galgotias College of Engineering Technology by:-

Rahul Burman (0509710079)

Vaibhav Bajpai (0509713060)

Nupur Dixit (0509710063)

Yadavendra (0509710121)

Guide	Head of Department(Computer Science)
Mr. Sanjeev Pippal	Prof. Parmanand Astya

Dexter

Making search web 2.0 ready!

ACKNOWLEDGEMENT

We began a journey last year not knowing what perils lay in our path but only with a determination to be able to create a product, off which ourselves would be proud. Now when we see Dexter our heart gets filled with joy and eyes with tears as we remember all the effort, sacrifices and toils we underwent to that made this day possible.

In this moment of happiness we would like to thank Prof. Parmanand (HOD-CS) and our guide Mr., Sanjeev Pippal. If it was not for their guidance and encouragement this project would have remained a dream. I would also like to thank Mr. Vimal Gupta and all other faculty members who constantly supported us during the development of the project.

Dexter

Making search web 2.0 ready!

ABSTRACT

Objective – to blend social networking into the Google search engine and encapsulate it into a Web 2.0 browsing environment.

Inadequacies in Google search engine –

- The Google search results are still currently “only” ordered based on a page ranking algorithm completely isolating the ability of the user to provide a feedback to the search results.
- The search engine does not still provide the ability to bookmark the search results in the profile, resting the functionality to be provided by the browser (that stores bookmarks in local storage) or third party web solutions to keep them in the “cloud”.
- The Google homepage rests waiting for the user to provide the keyword before some “information content” could be made available.
- Google does not provide possibility of collaborating a search with a friend!

Solution –

- Dexter would enable users to vote the search results and comment on them, still preserving the current page rank algorithm
- Dexter would provide a full-fledged social bookmarking using tags on top of Google search results and save ‘em in the cloud.
- Dexter would also include another interface “Dexter Surf”, that would “push” the top voted content in a specific category or a web service(like YouTube) for a

Dexter

Making search web 2.0 ready!

desired timeframe in the homepage itself, providing “information content” as soon as one logs in.

- Dexter would maintain profile pages and provide ability to make friends whose activities could be followed, thereby increasing collaboration

Dexter

Making search web 2.0 ready!

TABLE OF CONTENTS

Certificate of originality	2
Acknowledgement	3
Abstract	4
Feasibility Study	8
Project Plan	13
Software Requirements Specification	15
◆ <i>Introduction</i>	15
◆ Purpose	15
◆ Document Conventions	15
◆ Intended Audience and Reading Suggestions	16
◆ Product Scope	16
◆ References	17
◆ <i>Overall Description</i>	18
◆ Product Perspective	18
◆ Product Functions	18
◆ User Classes and Characteristics	19
◆ Operating Environment	20
Design and Implementation Constraints	21
◆ User Documentation	22
◆ Assumptions and Dependencies	22
◆ <i>External Interface Requirements</i>	23
◆ User Interfaces	23
◆ Hardware Interfaces	28
◆ Software Interfaces	28
◆ Communications Interfaces	28
◆ <i>System Features</i>	29
◆ Actor Inheritance	29
◆ Member Actions (Feature 1)	30
◆ Moderator Actions (Feature 2)	30
◆ Administrator Actions (Feature 3)	31
◆ View Profile (Feature 4)	31
◆ Member Search (Feature 5)	32
◆ View Friends (Feature 6)	33
◆ Settings (Feature 7)	33
◆ Search (Feature 8)	34

Dexter

Making search web 2.0 ready!

◆ Guest Actions (Feature 9)	34
◆ Authentication (Feature 10)	35
◆ Filter (Feature 11)	35
◆ <i>Other Nonfunctional Requirements</i>	36
◆ Performance Requirements	36
◆ Safety Requirements	36
◆ Security Requirements	36
◆ Software Quality Attributes	36
Entity Relationship Diagram	37
<i>User Module</i>	37
<i>Admin Module</i>	38
<i>URL Module</i>	39
Class Diagram	40
<i>Action Package</i>	40
<i>Beans Package</i>	49
<i>Filters Package</i>	59
<i>Listener Package</i>	60
<i>Utilities package</i>	62
Implementation	64
<i>Code Snippet</i>	64
Future Possibilities	86
Conclusion	87
Appendix A: Glossary	88
Appendix B: Bibliography	90

Dexter

Making search web 2.0 ready!

FEASIBILITY STUDY

Objective *TO BLEND SOCIAL NETWORKING INTO A SEARCH ENGINE AND
ENCAPSULATE IT TO A WEB 2.0 BROWSING ENVIRONMENTS.*

Implementation Alternatives

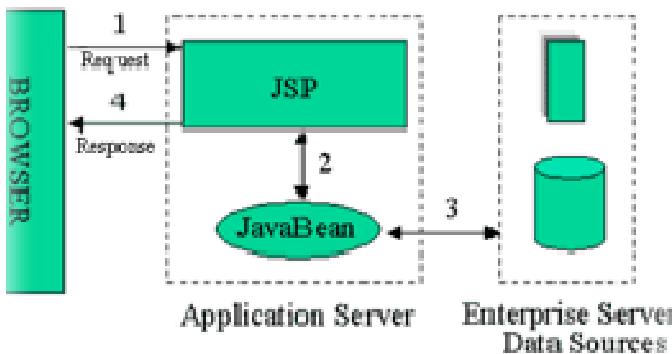


FIGURE 2: MODEL 1 ARCHITECTURE

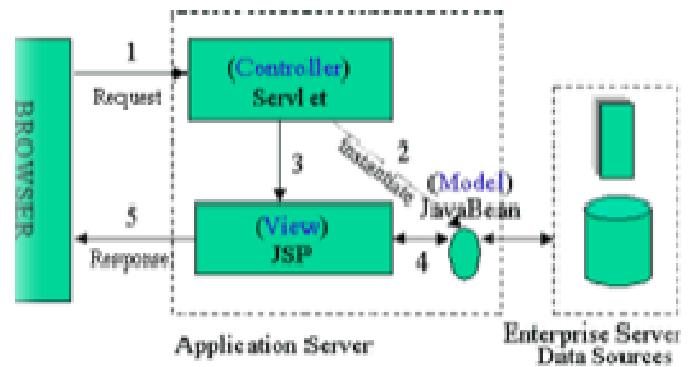


FIGURE 1: MODEL 2 ARCHITECTURE

Architectural Alternative

The Java EE Framework builds upon one of the two aforementioned web component architectures

Dexter

Making search web 2.0 ready!

Dexter is proposed to use the MVC Design Pattern (Model 2) with the view of completely isolating the presentation layer from the business logic to ensure higher degree of maintainability, reusability, and security of the product. This would eventually guarantee that the project would be able to overlay its Web 2.0 capabilities over any search engine whatsoever with minimum effort.

Moreover the inculcation of a full fledged ORM framework for back-end processing demands a design pattern as sophisticated as MVC in itself.

Economic Feasibility

The expected economic cost of the project is as follows

- Cost of Software Packages
 - There was a conscious decision to use Free and Libre' Open Source Software (Database, Package, IDE, SDK's, Application Servers), so the economic liability to their usage is ruled out.
- Cost of Hardware.
 - We would require 4 machines having at least 512MB ram and Pentium IV architecture. To develop this project
- Software Cost Estimation Through COCOMO
 - Dexter is an intermediate (in size and complexity) software projects in which team member have mixed experience levels in the technologies being used we categorize 'Dexter' as an intermediate project.

Intermediate COCOMO computes software development effort as function of program size and a set of "cost drivers" that include subjective assessment of product, hardware, personnel and project attributes. This extension considers a set of four "cost drivers", each with a number of subsidiary attributes. Given next is how Dexter evaluates to these attributes.

Drivers	Ratings	EAF
Product attributes		
Required software reliability	Normal	1.00
Size of application database	High	1.08
Complexity of the product	Very High	1.30
Hardware attributes		
Run-time performance constraints	High	1.11
Memory constraints	Normal	1.00

Dexter

Making search web 2.0 ready!

Volatility of the virtual machine environment	Low	0.87
Required turnaround time	Normal	1.00
Personnel attributes		
Analyst capability	Low	1.19
Applications experience	Normal	1.00
Software engineer capability	Low	1.17
Virtual machine experience	Normal	1.00
Programming language experience	Normal	1.00
Project attributes		
Use of software tools	Very High	0.82
Application of software engineering methods	Very High	0.83
Required development schedule	High	1.04
Total effort adjustment factor		1.34

The Intermediate COCOMO formula is of the form:

$$E = a_i(KloC)^{b_i} \cdot EAF$$

Where **E** is the effort applied in person-months, **KloC** is the estimated number of thousands of delivered lines of code for the project, and **EAF** is the factor calculated above. The coefficient $a_i = 3.0$ and exponent $b_i = 1.12$ for a semi detached project.

Dexter is expected to have a delivered line of code of around 10 Kloc

$$E = 3(10)^{1.12} \cdot 1.34$$

$$E = 53$$

Average staffing = (53 Person-Months) / (11 Months) = 4.81 people:

Technical Feasibility

Programming Language

A number of languages can be used for developing a web-app today. Most of them are well equipped with web framework that uses the MVC architecture. Some of the

Dexter

Making search web 2.0 ready!

most prominent languages used in web-application development and their corresponding MVC framework are given below.

- **Java**
 - Struts
 - Java Server Faces
 - Spring
- **Python**
 - Django
 - Pylons
- **Ruby**
 - Ruby on Rails
 - Nitro
- **Perl**
 - Catalyst
- **.Net Languages**
 - ASP .net MVC Framework
 - Monorail

Dexter Team has prior experience of coding in Java using Struts, thus it was chosen as the web framework.

Operational Feasibility

- DESIGN GOALS OF DEXTER INCLUDE
- CONSISTENT LOOK AND FEEL ACROSS THE WEB-APPLICATION
- SIMPLE USER INTERFACE
- FEEDBACK AFTER EVERY ACTION
- INFORMATION TO BE DISPLAYED IN A STRUCTURED MANNER BY GROUPING SIMILAR INFORMATION
- TOLERANCE TOWARDS USER ERROR (IGNORING WRONG INPUT, PROVIDING EASY RECOVERY OR GRACEFUL FALL)

Dexter is a web-application which is being designed keeping in mind the least common denominator in terms of the user's technological capability which amounts a user with only a basic understanding of computers enabling any person who uses internet to become its prospective user.

Dexter

Making search web 2.0 ready!

Political Feasibility

As we use open source or/and free technology we don't expect to face any legal or political difficulty.

Dexter

Making search web 2.0 ready!

PROJECT PLAN

Dexter has been divided into four modules.

MODULE NAME	DESIGNATED TO
DATABASE HANDLING USING HIBERNATES	Rahul Burman
MVC USING STRUTS	Vaibhav Bajpai
PRESENTATION LAYER	Nupur Dixit
ACCESSORIES	Yadavendra Yadav

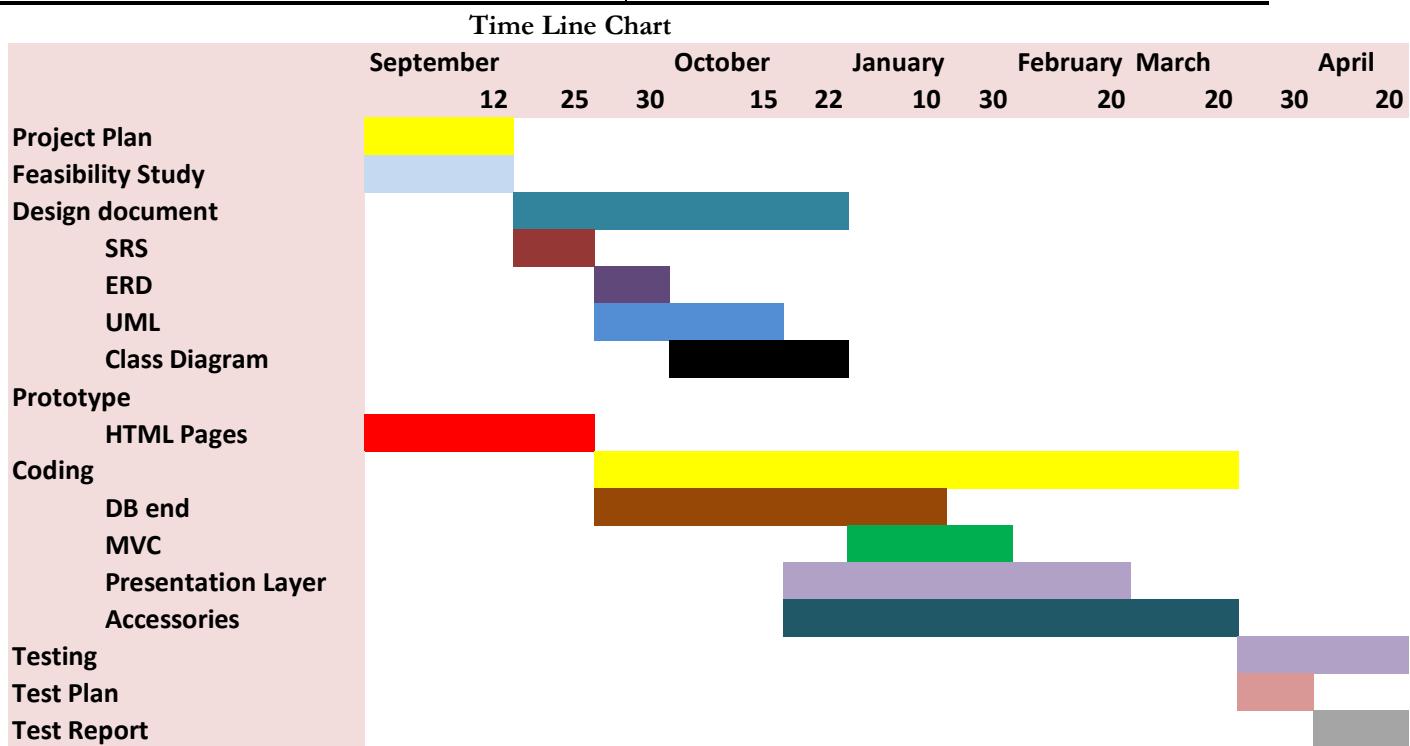
Schedule decided for Dexter is as follows

PHASE	DATE OF COMPLETION
PROJECT PLAN	12 th September
FEASIBILITY STUDY	12 th September
DESIGN DOCUMENT	
<i>SRS</i>	25 th September
<i>ERD</i>	30 th September
<i>UML</i>	15 th October
<i>Class Diagram</i>	22 nd October
PROTOTYPE	
<i>HTML Pages</i>	25 th September
CODING	
<i>Database and Access Layer</i>	10 th January
<i>MVC</i>	30 th January

Dexter

Making search web 2.0 ready!

Presentation Layer		
ACCESSORIES		20 th February
TESTING		20 th March
<i>Test Plan</i>		30 th March



Dexter

Making search web 2.0 ready!

SOFTWARE REQUIREMENTS SPECIFICATION

□ INTRODUCTION

□ PURPOSE

The purpose of the document is to describe the system requirements of the project ‘Dexter’. The document contains sufficient detail in the functional system requirements so that a full-fledged design solution can be devised. The functional level requirements are specified using the Use Case Model approach.

The document would place borders around the problem, solidify ideas, and help break down the problem into its component parts in an orderly fashion.

The document would also serve as the parent document for testing and validation strategies that will be applied to the requirements for verification.

Dexter is a product which allows user to search the web. It differs from existing product in the market like Google, yahoo by taking an alternative methodology to search by taking in account the human opinion.

□ DOCUMENT CONVENTIONS

The terms included in single quotation mark signify terms having special meaning in context of ‘Dexter’ and their corresponding meaning have been explained in Glossary.

Dexter

Making search web 2.0 ready!

□ INTENDED AUDIENCE AND READING SUGGESTIONS

The intended audience of this document is people interested in understanding the basic requirement and functionality of ‘Dexter’.

□ PRODUCT SCOPE

Objective – to blend social networking into the Google search engine and encapsulate it into a Web 2.0 browsing environment.

Goals –

- The search results should be ordered in a way that they give preference to reader’s opinion over algorithmic or machine based approaches.
- The non submitted search results should show up in the same order as that by the page rank algorithm
- The search interface should be as simplistic as the Google search engine homepage.
- The user should be able to separately ‘submit’ a URL to the Dexter database.
- Any URL ever ‘voted up’ or submitted should be logged in for future reference by the user.
- The user should be able to ‘favorite’ the voted results, and be able to tag them.
- User should be able to add ‘friends’.
- The user should be able to follow the activity of his ‘friends’ and be able to ‘shout’/email them of an interesting content.
- The privacy of the user profile should be maintained.
- User should be able to choose one of the two interfaces as default home (‘Surf’ /‘Search’)
- Mechanism should exist to weed out ‘spam’ and ‘inappropriate’ url and comment

Dexter

Making search web 2.0 ready!

Benefits –

- Dexter implements the social search paradigm where relevance of search results is determined by considering the interactions or contributions of users, in contrast to established algorithmic or machine-based approaches where relevance is determined by analyzing the text of each document or the link structure of the documents
- Dexter would enable users to vote the search results and comment on them, still preserving the current page rank algorithm
- Dexter would provide a full-fledged social bookmarking using tags on top of Google search results and save ‘em in the cloud.
- Dexter would also include another interface “Dexter Surf”, that would “push” the top voted content in a specific category or a web service(like YouTube) for a desired timeframe in the homepage itself, providing “information content” as soon as one logs in.
- Dexter would maintain profile pages and provide ability to make friends whose activities could be followed, thereby increasing collaboration
- Reduces impact of link spam on search result by relying less on link structure of web pages
- Increased relevance because each result has been selected by users
- leverage a network of trusted individuals by providing an indication of whether they thought a particular result was good or bad
- The introduction of ‘human judgment’ suggests that each web page has been viewed and endorsed by one or more people, and they have concluded it is relevant and worthy of being shared with others using human techniques that go beyond the computer’s current ability to analyze a web page.
- Web pages are considered to be relevant from the reader’s perspective, rather than the author who desires their content to be viewed, or the web master as they create links.
- As ‘Dexter’ would be constantly getting feedback it is potentially able to display results that are more current or in context with changing information

□ REFERENCES

Software Requirements Specifications – [Wikipedia](#)

Dexter

Making search web 2.0 ready!

□ OVERALL DESCRIPTION

□ PRODUCT PERSPECTIVE

Dexter is assumed to be a replacement for current conventional Web 1.0 search engines, which are based upon an algorithm that decides the value/order of a search result URL based upon a fixed criteria (*link recommendation as used in Google or keyword based search as used in Yahoo!*).

These search engines do not involve the human element in deciding the effectiveness of a search result URL which seems inadequate in today's Web 2.0 scenario.

The philosophy behind "Dexter" is that humans are the best judge of a page's relevance and it's their opinion which is given a higher priority over algorithmic approaches.

□ PRODUCT FUNCTIONS

- The user should be able to search the web without having to leave 'Dexter'
- The user should be able to search without being logged in, though should not be allowed to vote or comment.
- The user should be able to separately submit a URL to the Dexter database.
- The user should be able to bookmark the voted results, and be able to tag them.
- The user should be able to see its recent activity as in comments and votes.
- The user should be able to follow the activity of its friends and be able to shout/email them of an interesting content.
- User should be able to choose one of the two interfaces as default home ('Surf' / 'Classic')
- User should be able to add Dexter friends from Google contacts.
- User can view the article that have been endorsed maximum no of times in various 'category', 'channel' or 'timeframe'
- Categories and channel can be added/removed dynamically.
- Any url/comment can be reported by the user after the user has 'voted it down' and moderators would take decision on the reported URL and comment.

Dexter

Making search web 2.0 ready!

□ USER CLASSES AND CHARACTERISTICS

Users

Dexter treats every “user” as equal invariant of one’s technical expertise, educational level, or experience to reflect transparency in the sanctity of the votes.

Moderators

Will have the privilege to

- delete a submitted url based on profanity of content
- delete a submitted comment based on profanity of content
- Change the category of submitted URL based on reports received.

Administrators

- . Have privilege to add/remove categories and order them as would appear on the web-page.
- . Have privilege to add/remove channel and order them as would appear on the web-page.
- . have privilege to ban user
- . have privilege to add/remove IM service name offered to user to supply their IM details
- . have privilege to add/remove a visibility control level
- . Assign moderator privilege to users.

Dexter

Making search web 2.0 ready!

□ OPERATING ENVIRONMENT

Client-Side Requirements

Hardware Platform –

Dexter, being a web-application only restricts the hardware requirements for the client to the capability of a machine being able to connect to the internet or the LAN in which the web application is deployed to

Operating System –

Dexter, being a web-application would allow the client side interface to run on all Operating Systems that have an inbuilt network stack incorporated into the kernel.

Software Platform –

Browser – IE 6+, Firefox 2+, Safari, Opera, Chrome

Server-Side Requirements

Hardware Platform –

- at-least 256 MB RAM
- 10 GB HDD space

Operating System –

- 32-bit Linux/Windows Operating System
- 2 GB Page File/Swap

Software Platforms –

- Java SE 6
- Java EE 5 compatible Application Server (Glass fish)
- Apache Web Server
- Struts and Hibernate libraries.

Dexter

Making search web 2.0 ready!

DESIGN AND IMPLEMENTATION CONSTRAINTS

Paradigm Constraints

- As users can directly add results to a social search engine there is a risk that some users could insert search spam directly into the search engine. Elimination or prevention of this spam would require the ability to detect the validity of a users' contribution, such as whether it agrees with other trusted users.
- There are so many unique searches conducted that most searches, while valid, are performed very infrequently. A search engine that relied on users filling in all the searches would be at a disadvantage to one that used machines to crawl and index the entire web.
- ① The application would not be much useful to users until it acquires a reasonable user base.

Hardware/Software Limitations

- Oracle 10XE used in the project is not available for Mac OSX
- Java SE 6 is not available for 32-bit hardware running Mac OSX

Software Constraints

- Explicit usage of Free and/or Open Source Software
- Explicit Avoidance of Bloated Packages (Netbeans)

Design Constraints:

- No Explicit Style code in HTML
- No Flash Content
- No Frames
- No Tables for Layout Design

Coding Constraints:

- No Business Logic in Presentation Layer
- Separation of Form Beans and Data Beans
- Security Layer in Authentication

Dexter

Making search web 2.0 ready!

- Resolving the Object-Relational Mismatch
- Explicit HTML Coding using HTML Editor than using RAD tools

□ USER DOCUMENTATION

- Java Docs for the source.
- About/FAQ in the Web-application website.
- Video Screen-cast explaining the features/usage.

□ ASSUMPTIONS AND DEPENDENCIES

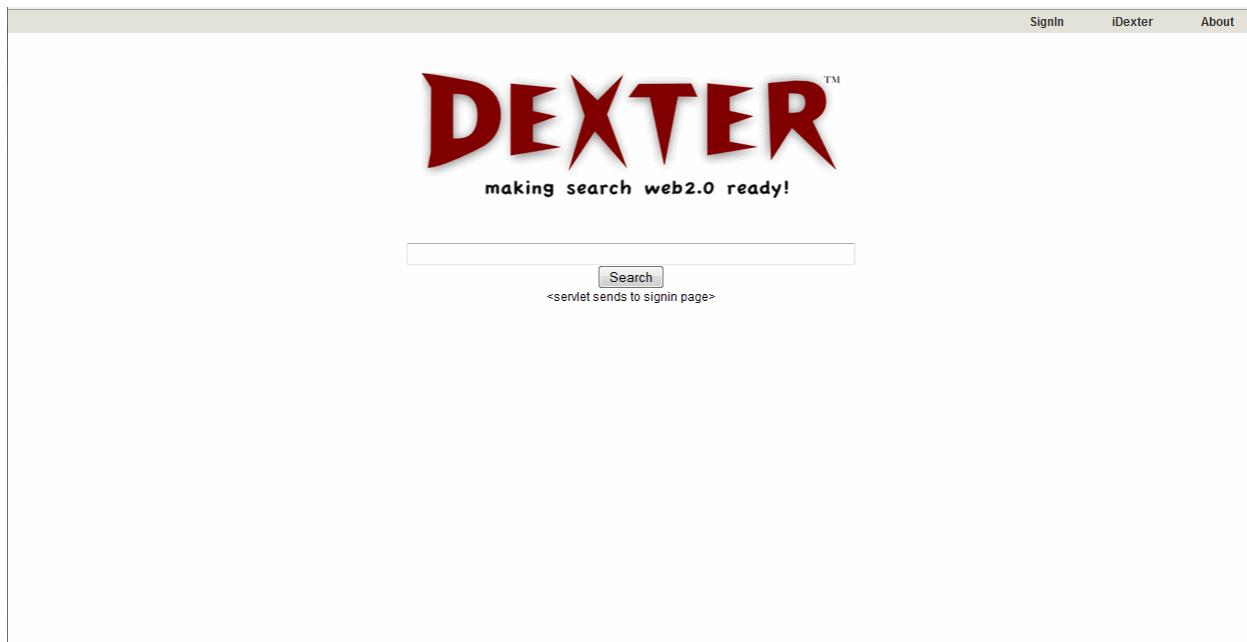
- Dexter being an overlay on top of the search engine, assumes the 24x7 functioning of the search engine in itself.
- Dexter uses Google AJAX Web Search API to retrieve the search results pushed against a query, and assumes the API would remain open to public.
- Dexter uses Google Account Authentication API to retrieve contact details of a user, and assumes the API would remain open to public.

Dexter

Making search web 2.0 ready!

□ EXTERNAL INTERFACE REQUIREMENTS

□ USER INTERFACES



Dexter Search Page

A screenshot of the Dexter search results page. The page has a dark header with various navigation links like 'Technology', 'Business', 'Science', etc. Below the header is a search bar. The main content area displays two search results. Each result is preceded by a small orange thumbs-up icon. The first result is 'URL1(20) <curl directs to actual page>' followed by 'comment, favorite, submitter id<takes to profile page>, time submitted'. The second result is 'URL2(10) <curl directs to actual page>' followed by the same comment and submission details. At the bottom of the page, there are 'Previous' and 'Next' navigation links.

Search Results Page – Higher voted results on top.

Dexter

Making search web 2.0 ready!

The screenshot shows the Dexter homepage. At the top is a navigation bar with links for Signout, Classic, My Profile, My Friends, Submit, Settings, and About. Below the navigation bar is a main menu with categories: Technology, Business, Science, Gaming, Lifestyle, Entertainment, Offbeat, and Sports. Under the Lifestyle category, there are sub-links for Art, Educational, Food, Health, and Travel. Below the main menu is a section for Images, Youtube, Wikipedia, Orkut, Blogs, and News. A Timeframe dropdown shows options for Today, Week, Month, and Year. The main content area displays two items, each with a thumbs-up icon, a URL link, a description, and a timestamp.

URL1(20) <url directs to actual page>
comment, favorite, submitter id<takes to profile page>, time submitted

URL2(10) <url directs to actual page>
comment, favorite, submitter id<takes to profile page>, time submitted

Previous Next

Dexter Surf (URLs belonging to a chosen category are listed)

The screenshot shows the Dexter Surf feature. It lists URLs under the Lifestyle category, each with a thumbs-up icon, a URL link, a description, and a timestamp. Below the list is a comment section with buttons for Comments, Shout, Share, Report, and Friend's Activity. The comment section includes a dropdown for sorting (Oldest First) and a note about highlighting friend-voted comments. It also shows a reply section for Comment1(30), Comment Replies (15), Comment2(20), and Comment3(10).

URL1(20) <url directs to actual page>
favorite, submitter id<takes to profile page>, time submitted

expand all | only mine | only friends Oldest First settings
<will use color scheme to highlight friend voted comments>

Comment1(30) reply

Comment Replies (15) reply <1 level only> reply <collapsible replies>

Comment2(20) reply

Comment3(10) reply

Action specific to a particular URL

Dexter

Making search web 2.0 ready!

Favourites

URL's you have recently added as your favourite appear here
[more...](#)

About Me

Vaibhav Bajpai
Joined Dexter in Feb 10, 2005



Instant Message

 Google Talk  Yahoo
 MSN  AIM

Social Networking Profiles

 Digg	 Orkut
 Facebook	 Twitter
 Last.fm	 LinkedIn
 YouTube	 StumbleUpon

Shouts

User Profile Page

Friend Requests

Rahul Burman has sent you a friend request.
Do you want to approve? [yes](#) [no](#)

Linus Torvalds has sent you a friend request.
Do you want to approve? [yes](#) [no](#)

Siddharth Singh has sent you a friend request.
Do you want to approve? [yes](#) [no](#)
[more...](#)

Friends

 Vaibhav Bajpai	 Garima Bajpai
 Amit Mishra	 Gaurav Bajpal

[more...](#)

Add Friends

Enter Dexter ID [Add](#)

Click [Here](#) to search for Dexter Friends...

User's Friends Page

Dexter

Making search web 2.0 ready!

Edit Profile My Settings

Preferences

Comment Sort	Oldest First
Expanded Comments	10 or more
Comments per Page	25
Comments Initially Collapsed	True

Privacy Settings

Privacy	Anyone	Friends	Only Me
FullName	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
Age	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
Gender	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
Location	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
Email Address	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
Website	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>

Change Password

Current Password	<input type="text"/>
New Password	<input type="text"/>
Confirm New Password	<input type="text"/>

Apply

Settings Page

Home About

DEXTER™
making search web2.0 ready!

Signin

Username	<input type="text"/>
Password	<input type="password"/>
Signin	

Don't have a Dexter account?
[Create an account now](#)

Lost Credentials?

Enter Your Email Address	<input type="text"/>
Retrieve Password	

Sign In Page

Dexter

Making search web 2.0 ready!

The screenshot shows the Dexter Admin Page interface. At the top, there is a navigation bar with links: Home, Signout, iDexter, My Profile, My Friends, Submit, Settings, and About. Below the navigation bar, there is a "Timeframe" dropdown menu with options: Today, Week, Month, and Year.

The main content area contains several modules:

- Add Category**: A form with a "Category Name" input field and an "Add" button.
- Remove Category**: A form with a "Category Name" dropdown menu set to "Technology" and a "Remove" button.
- Position Category**: An empty module.
- Category News**: A module displaying news items for the "Technology" category. It shows:
 - 1001 articles submitted in 'Entertainment' in last 1 day
 - 400 comments in 'Entertainment' in last 1 day

To the right, there is a **Dashboard** sidebar with the following menu items:

- Admin Home
- Channel
- Category** (highlighted with a blue arrow)
- User
- Moderator
- Instant Messaging
- Visibility Control

Admin Page

Dexter

Making search web 2.0 ready!

□ HARDWARE INTERFACES

Communication Protocols

- HTTP 1.1
- SSL for Login Authentication.

Minimum Number of Machines

- one or more - Database Storage
- one - Application Server Deployment (Server)
- one or more - Client

□ SOFTWARE INTERFACES

Database will be distributed across machines, the recordset would be retrieved by the application server deployed at another machine, where the result sets would map to EJB modules inside the EJB container which will be pushed to the Web Container for the servlets to work upon and build presentation level java beans for the JSP to invoke and display the results as standard HTML page, which would be sent to the client upon a GET/POST request.

□ COMMUNICATIONS INTERFACES

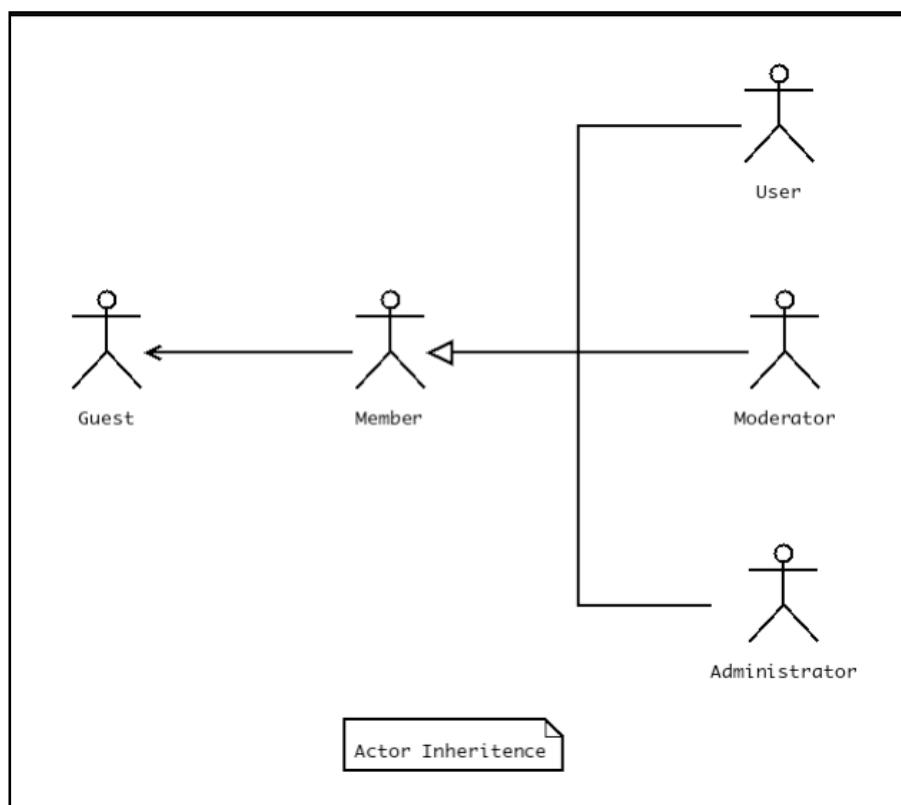
- Web-browser – the web-app would be accessed by the client using a web browser
- Protocols – HTTP 1.1, SMTP for sending email to friends
- Security – SSL for login authentication
- Data Transfer – the server response would be gzipped for efficiency.

Dexter

Making search web 2.0 ready!

□ SYSTEM FEATURES

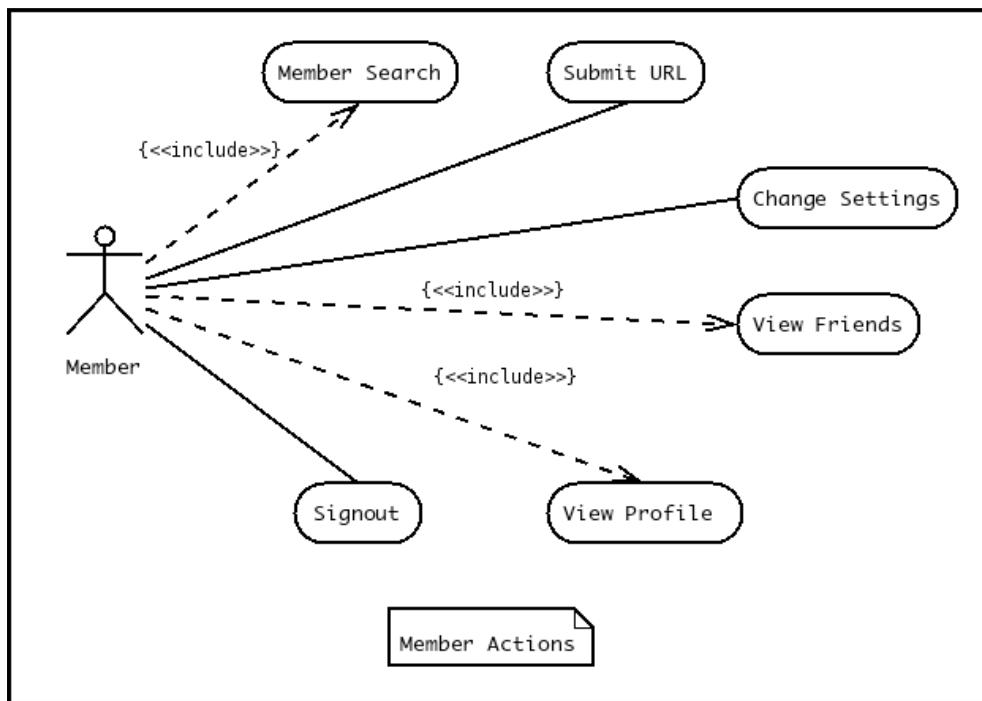
□ ACTOR INHERITANCE



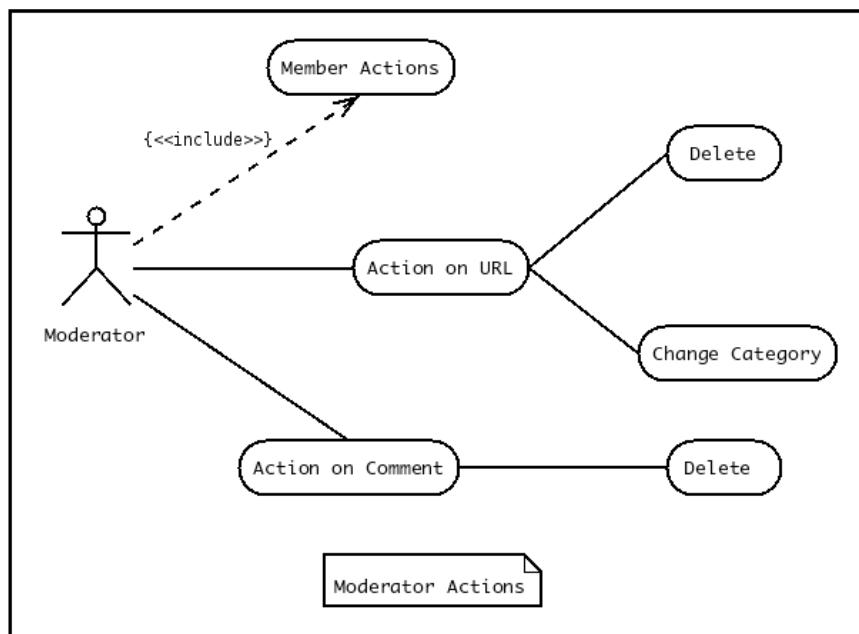
Dexter

Making search web 2.0 ready!

□ MEMBER ACTIONS (FEATURE 1)



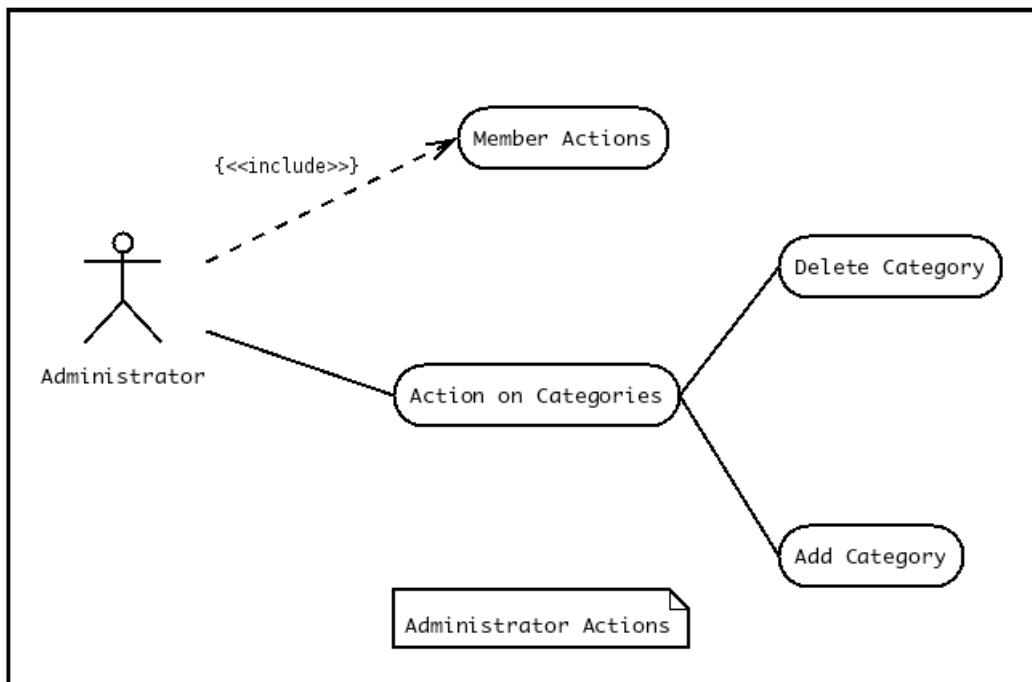
□ MODERATOR ACTIONS (FEATURE 2)



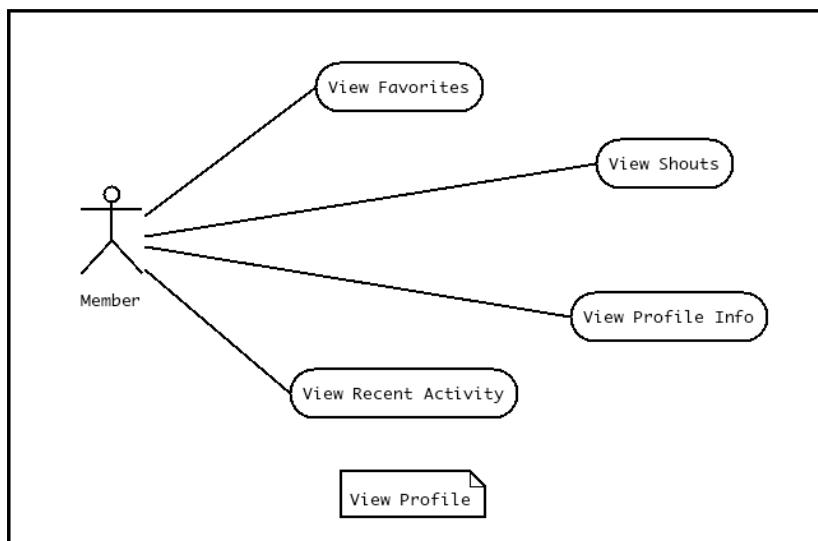
Dexter

Making search web 2.0 ready!

□ ADMINISTRATOR ACTIONS (FEATURE 3)



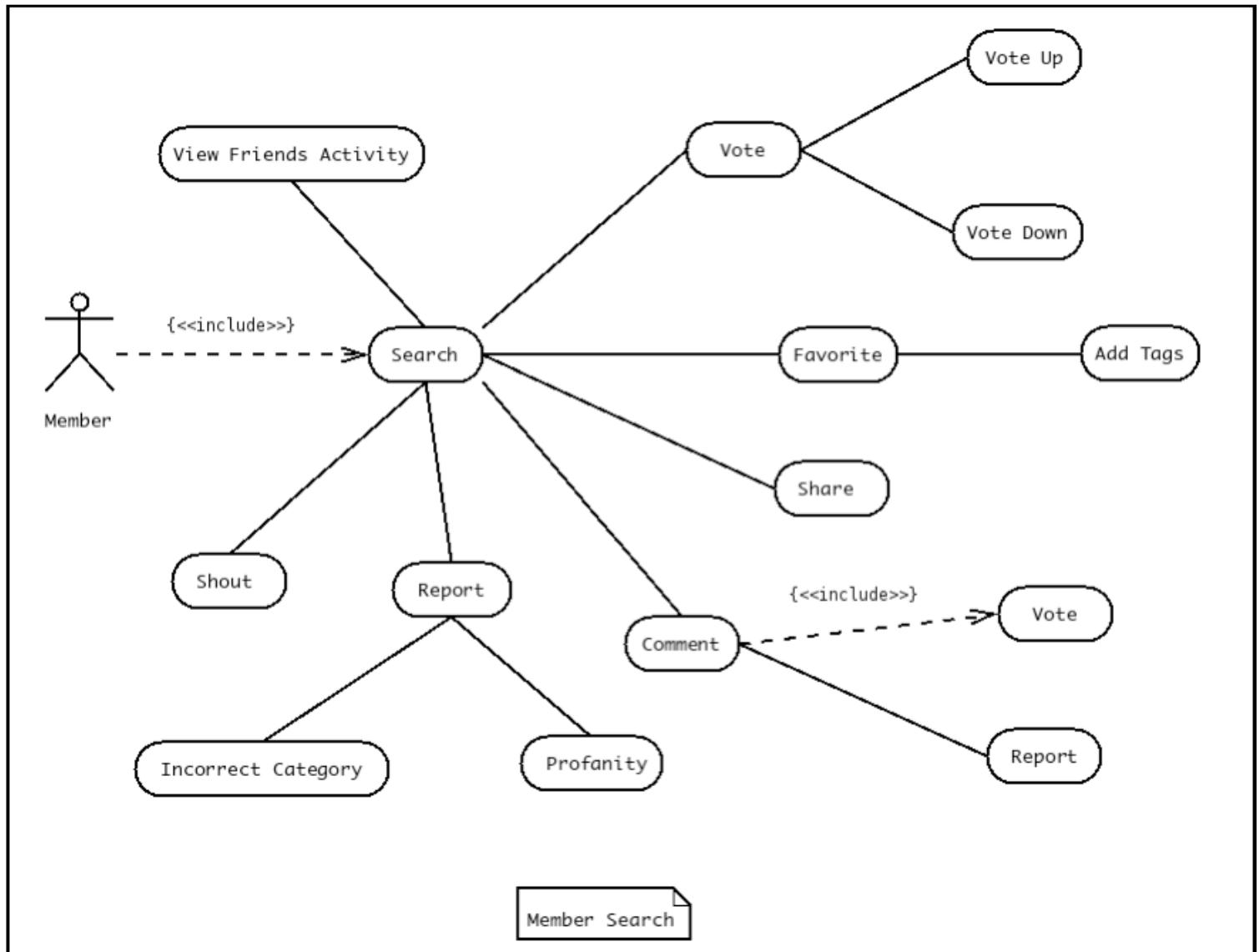
□ VIEW PROFILE (FEATURE 4)



Dexter

Making search web 2.0 ready!

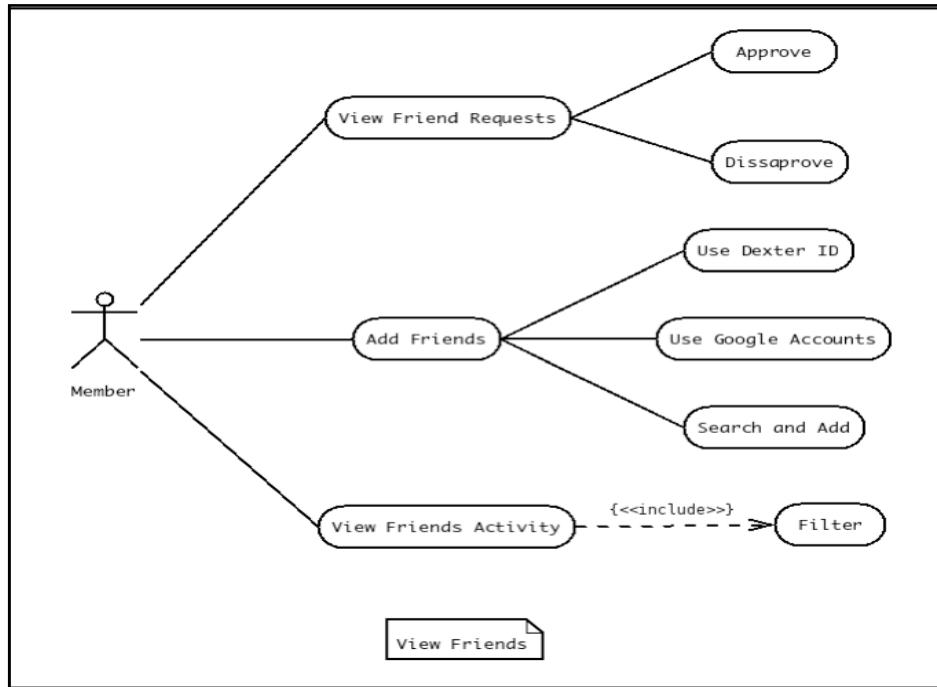
□ MEMBER SEARCH (FEATURE 5)



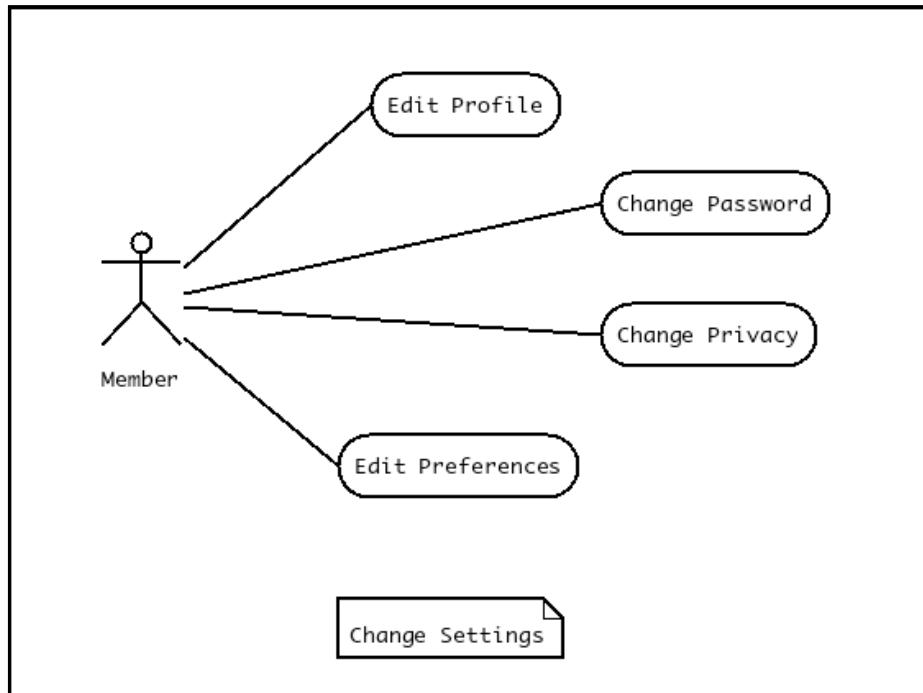
Dexter

Making search web 2.0 ready!

□ VIEW FRIENDS (FEATURE 6)



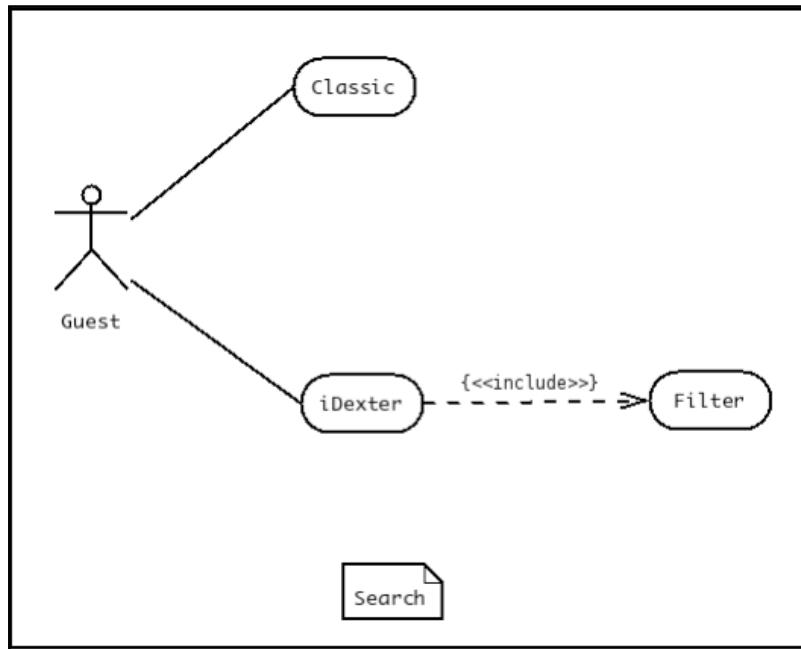
□ SETTINGS (FEATURE 7)



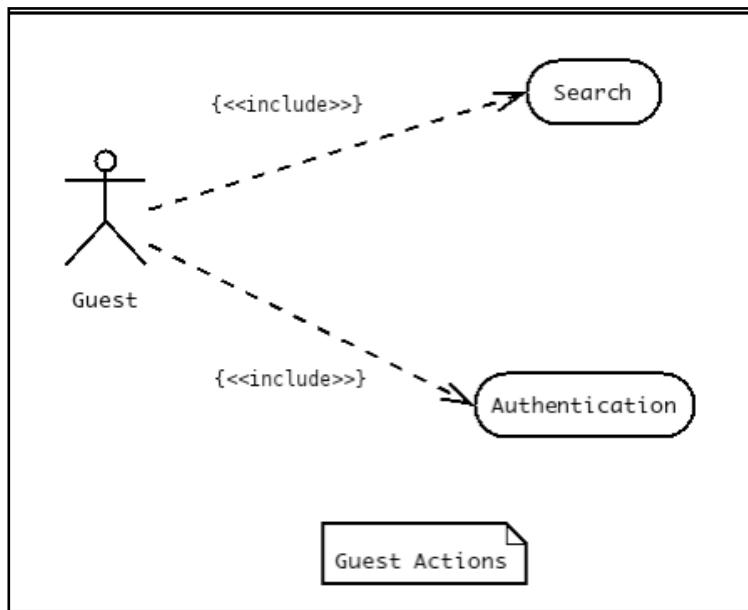
Dexter

Making search web 2.0 ready!

□ SEARCH (FEATURE 8)



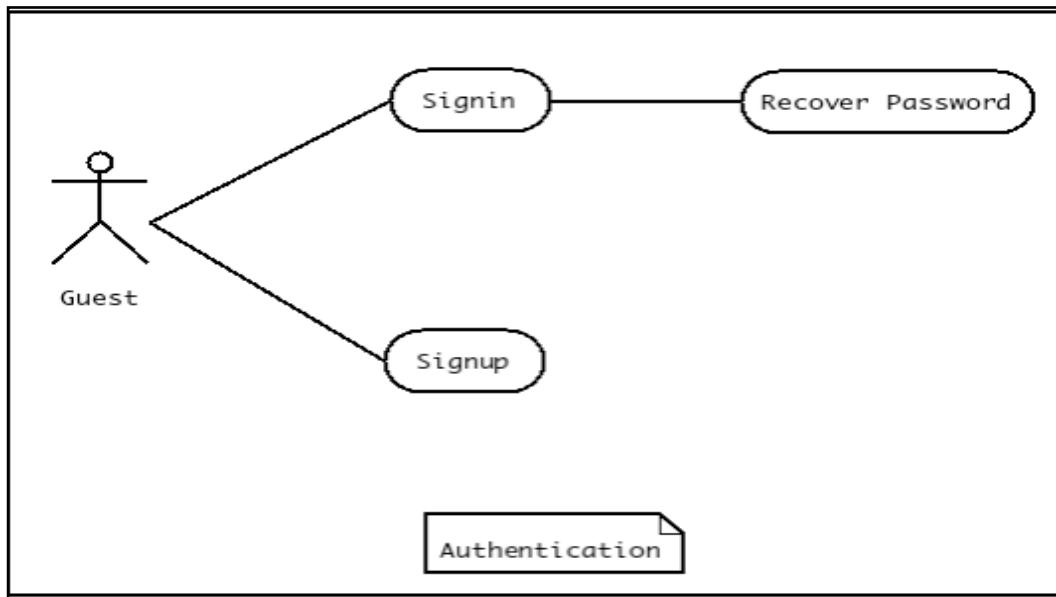
□ GUEST ACTIONS (FEATURE 9)



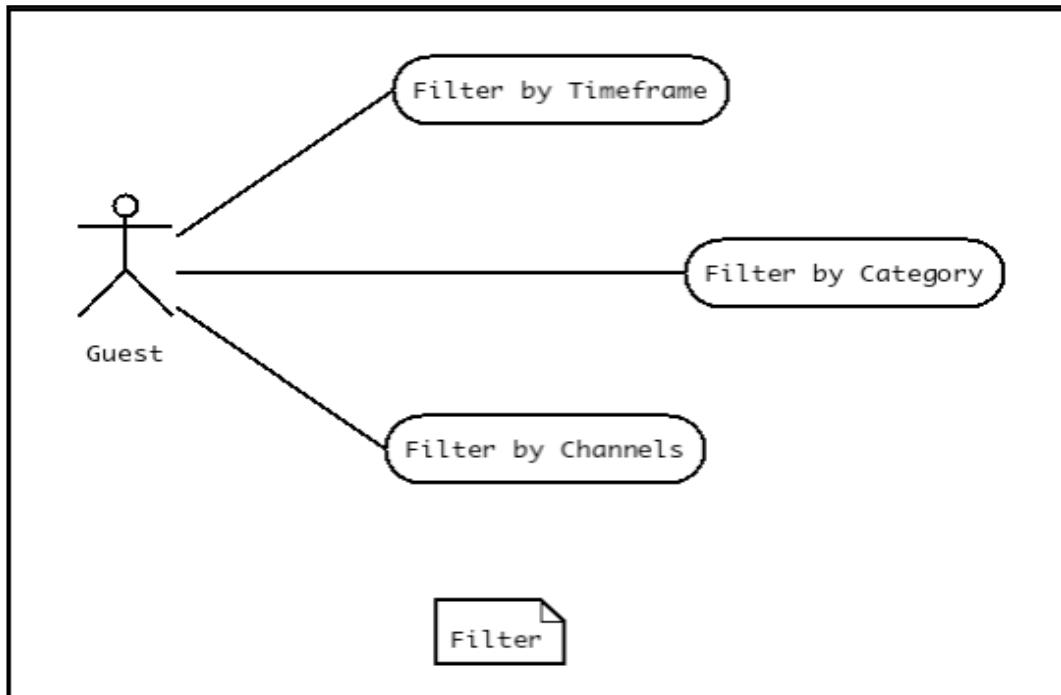
Dexter

Making search web 2.0 ready!

□ AUTHENTICATION (FEATURE 10)



□ FILTER (FEATURE 11)



Dexter

Making search web 2.0 ready!

□ OTHER NONFUNCTIONAL REQUIREMENTS

□ PERFORMANCE REQUIREMENTS

- The Google search result retrieval and display should be as fast as possible.

□ SAFETY REQUIREMENTS

- The search pattern as being logged by the web-application would reveal the user-characteristics and one's activity over the web.

□ SECURITY REQUIREMENTS

- The registration information should traverse the cloud in an encrypted form to prevent eavesdropping (need to use SSL)
- User needs to have options to set the privacy level for registration information and its activities in the form of comments or votes.

□ SOFTWARE QUALITY ATTRIBUTES

Adaptability – as Dexter directly uses the Google search result API, it would automatically adapt to the changing order of search results which have not yet been submitted.

Flexibility – the search results are flexible as they are user ordered.

Maintainability – Moderators/Administrators have the work to maintain the content in the Dexter database

Reusability – Dexter being open-source could be used by other projects who would wish to add to its functionality.

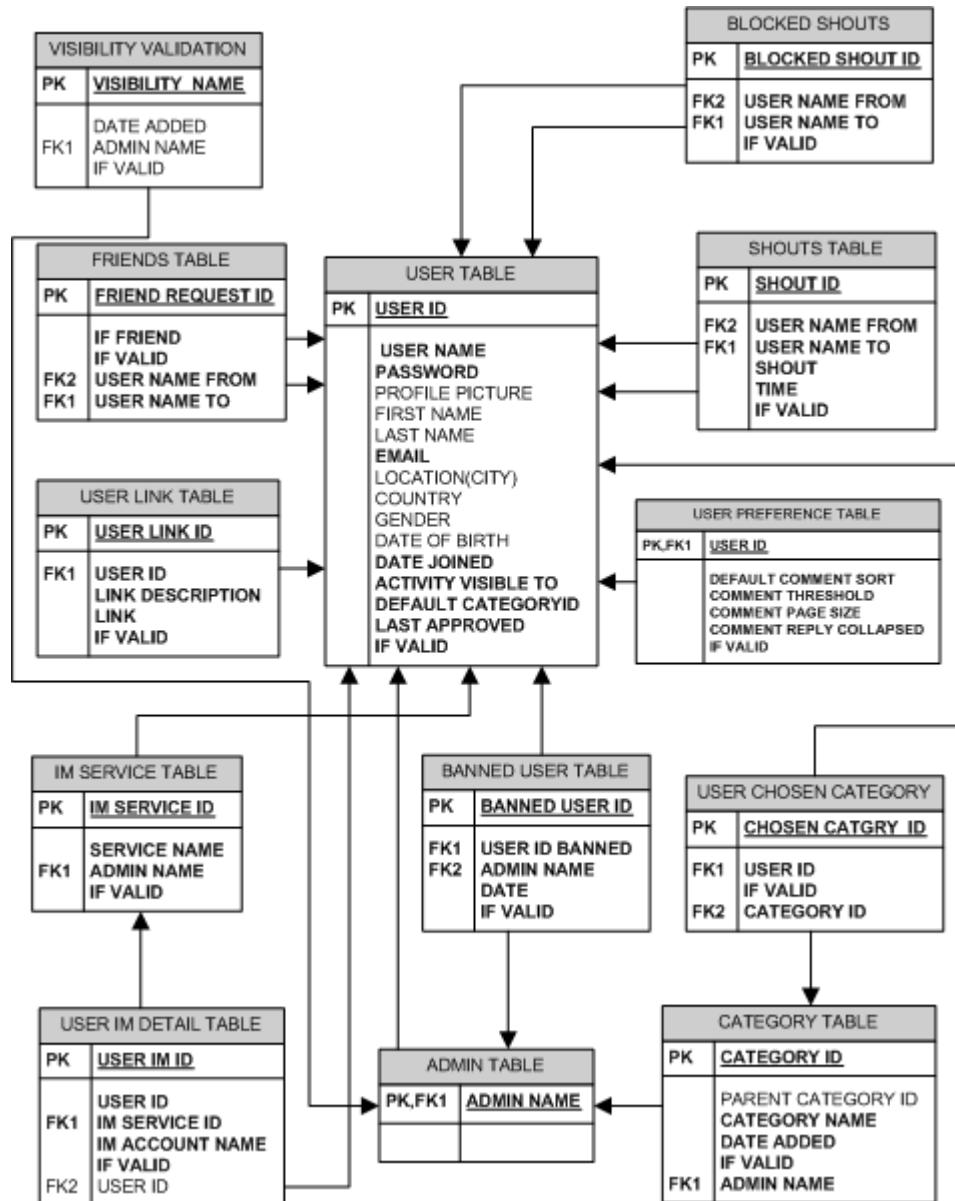
Usability – It's even usable in cases when user does not know what to search for, and can use the iDexter interface to get started.

Dexter

Making search web 2.0 ready!

ENTITY RELATIONSHIP DIAGRAM

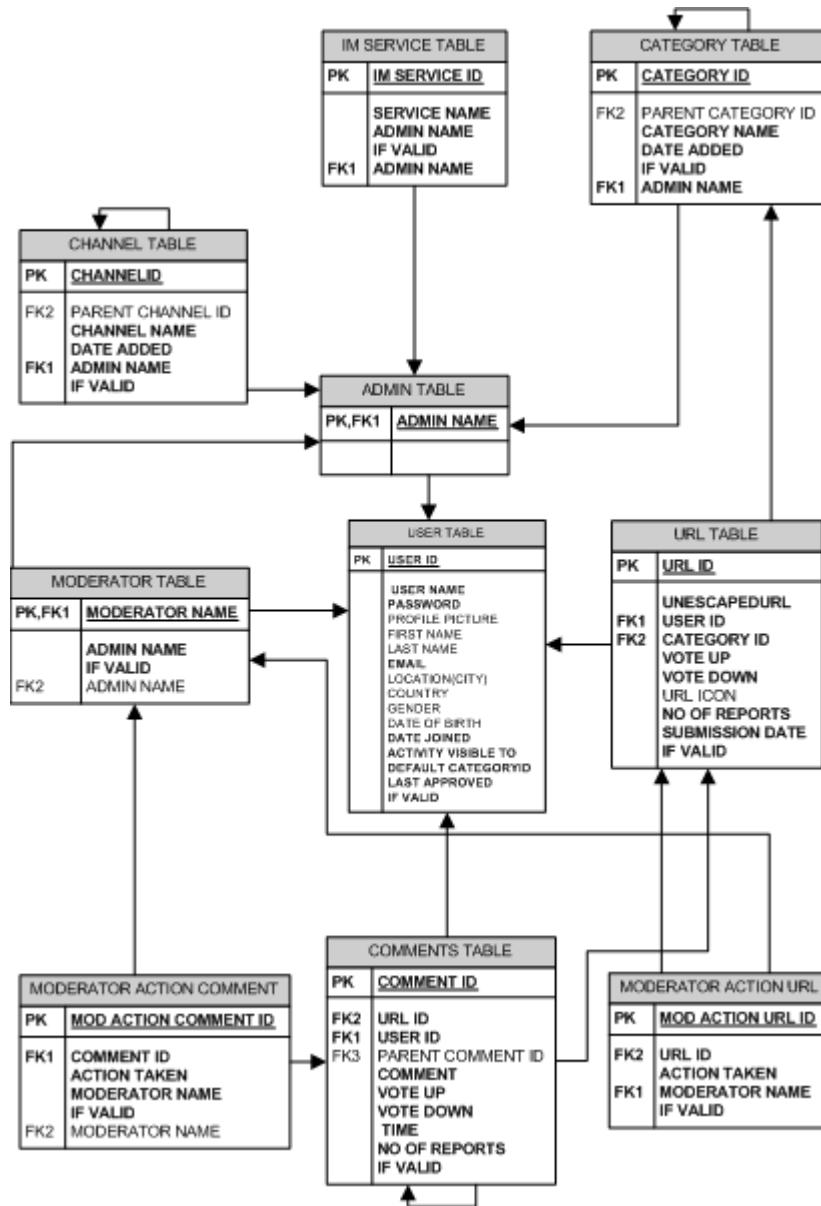
USER MODULE



Dexter

Making search web 2.0 ready!

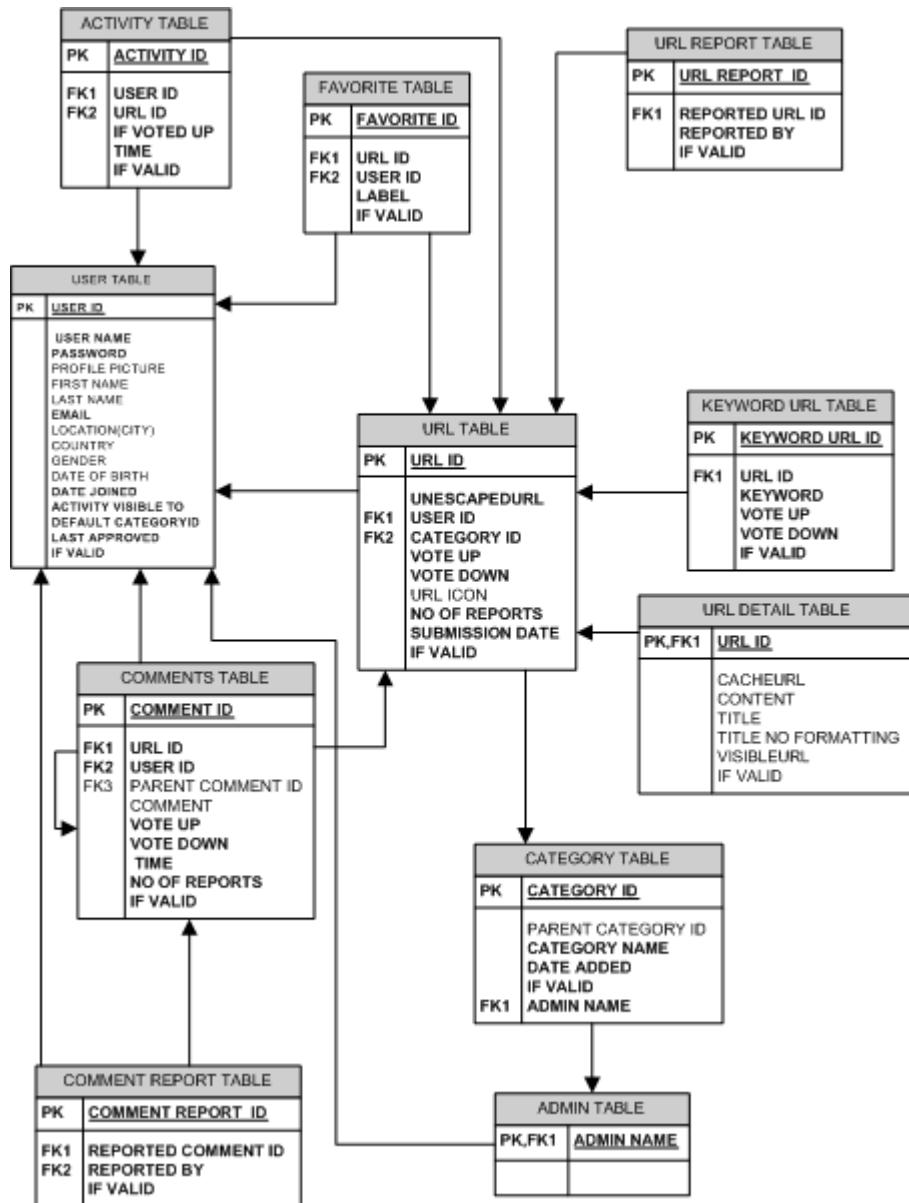
ADMIN MODULE



Dexter

Making search web 2.0 ready!

URL MODULE



Dexter

Making search web 2.0 ready!

CLASS DIAGRAM

ACTION PACKAGE

AddFriendAction	
Attributes	
private String YES = "yes"	
private String NO = "no"	
private String SUCCESS = "return"	
private FriendsFacadeRemote friendsFacadeRemote	
private UserFacadeRemote userFacadeRemote	
private User user	
private Friends friend	
Operations	
public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response)	
private void actionOnRequest(String ifapproved)	
private void addNewFriend(HttpServletResponse response)	

AllFriendsAction	
Attributes	
private String SUCCESS = "return"	
private FriendsFacadeRemote friendsRemote	
private HttpSession session	
private User user	
Operations	
public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response)	
private void saveFriendContext()	

DeleteUserAction	
Attributes	
private String SUCCESS = "return"	
private UserFacadeRemote userFacadeRemote	
Operations	
public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response)	

DisableUserAction	
Attributes	
private String SUCCESS = "return"	
private UserFacadeRemote userFacadeRemote	
Operations	
public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response)	

Dexter

Making search web 2.0 ready!

DisplayFavoriteAction	
<i>Attributes</i>	
private String SUCCESS = "return"	
private HttpSession session	
private User user	
private FavoriteFacadeRemote favoriteFacadeRemote	
private URLDetailFacadeRemote uRLDetailFacadeRemote	
private ServletContext servletContext	
<i>Operations</i>	
public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response)	
private void saveFavoriteContext()	

EditProfileAction	
<i>Attributes</i>	
private String RETURN = "return"	
<i>Operations</i>	
public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response)	

FavoriteAction	
<i>Attributes</i>	
private String page	
private String search	
private String category	
private String visibleURL	
private String timeframe	
private String urlid	
private String redirectURL	
private HttpSession session	
private URL url	
private URLFacadeRemote UrlRemote	
private User user	
private String tags	
private FavoriteFacadeRemote favoriteFacadeRemote	
<i>Operations</i>	
public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response)	
private String restructureURL()	

Dexter

Making search web 2.0 ready!

FriendsActivityAction	
<i>Attributes</i>	
private String SUCCESS = "return"	
private HttpSession session	
private User user	
private FriendsFacadeRemote friendsRemote	
private ActivityFacadeRemote activityFacadeRemote	
private Friends friendList[0..*]	
private Activity friendActivityList[0..*]	
private Friends friendRecord	
private URLDetailFacadeRemote uRLDetailFacadeRemote	
<i>Operations</i>	
public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response)	
private void saveFriendsRecentActivityContext()	

FriendsRequestAction	
<i>Attributes</i>	
private String SUCCESS = "return"	
private FriendsFacadeRemote friendsRemote	
private HttpSession session	
private User user	
<i>Operations</i>	
public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response)	
private void saveFriendRequestContext()	

IDexterAction	
<i>Attributes</i>	
private String KEY = "idexter"	
private String page	
private String timeframe	
private String category	
private String visibleURL	
private HttpSession session	
private ServletContext servletContext	
private User user	
<i>Operations</i>	
public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response)	
private void addRecord(URLDetail urlDetailRecord)	
private boolean checkCategory(URLDetail record)	
private boolean checkChannel(URLDetail record)	
private CategoryBean[0..*] checkParentCategory()	
private boolean checkTimeFrame(URLDetail record)	
private String getChannelName(URL url)	
private SearchResultBean[0..*] getSubmittedResults()	
private String getURLID(long id)	
private String getUserName(User user)	
private boolean isFilterConditionSatisfied(URLDetail record)	
private boolean isFilterON()	
private void populateResultBean(URLDetail record, SearchResultBean bean)	
private void sortSubmittedResults()	
private String getredirectFlag()	

Dexter

Making search web 2.0 ready!

LoginAction	
<i>Attributes</i>	
<pre>private String CLASSICSUCCESS = "classic" private String IDEXTERSUCCESS = "idexter" private String FAILURE = "failure" private String ADMINISTRATOR = "administrator" private String MODERATOR = "moderator" private String USER = "user" private User userRecord private UserLink userLinkRecord private UserFacadeRemote userFacadeRemote private ModeratorFacadeRemote moderatorFacadeRemote private UserLinkFacadeRemote userLinkFacadeRemote private BannedUserFacadeRemote bannedUserFacadeRemote package String redirectionURL</pre>	
<i>Operations</i>	
<pre>public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response) public String getUserRole(String username) public boolean mapBeanUserDB(SignupFormBean userbean) public boolean passwordValid(String password) public boolean usernameValid(String username) private boolean accountEnabled() private boolean checkRedirectionFailure() private boolean checkRedirectionSuccess() private void populatePreferenceBean(PreferenceFormBean bean, UserPreference record) private void populatePrivacyBean(PrivacyFormBean privacybean, UserPrivacy privacyRecord) private void saveUserContext(HttpSession session) private boolean userBanned()</pre>	

LogoutAction	
<i>Attributes</i>	
<pre>private String SUCCESS = "success"</pre>	
<i>Operations</i>	
<pre>public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response)</pre>	

Dexter

Making search web 2.0 ready!

MoreInfoAction	
<i>Attributes</i>	
private String SUCCESS = "return" private URLReportFacadeRemote uRLReportFacadeRemote private URLFacadeRemote uRLFacadeRemote private ActivityFacadeRemote activityFacadeRemote private HttpSession session private String urlid private URL url private User user private long longurlid private UserFacadeRemote userFacadeRemote private BlockedShoutsFacadeRemote blockedShoutsFacadeRemote private ShoutsFacadeRemote shoutFacadeRemote	
<i>Operations</i>	
public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response) private void createShout() package void saveWhoVotedItUp() package void reportedURLHandler()	

MyFriendAction	
<i>Attributes</i>	
private String SUCCESS = "return" private FriendsFacadeRemote friendsRemote private HttpSession session private User user	
<i>Operations</i>	
public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response) private void saveFriendContext() private void saveFriendRequestContext()	

MyProfileAction	
<i>Attributes</i>	
private String SUCCESS = "return" private HttpSession session private User user private ShoutsFacadeRemote shoutsRemote private FavoriteFacadeRemote favoriteFacadeRemote private URLDetailFacadeRemote uRLDetailFacadeRemote private ActivityFacadeRemote activityFacadeRemote	
<i>Operations</i>	
public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response) private void saveShoutContext() private void saveFavoriteContext() private void saveRecentActivityContext() private void setDateString(Shouts shoutsList[0..*])	

Dexter

Making search web 2.0 ready!

PreferencesAction	
Attributes	
private String SUCCESS = "return"	
package UserPreference recordSet[0..*]	
package User user	
private UserPreference userPreferenceRecord	
Operations	
public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response)	
private boolean recordExists()	
private boolean getCommentReplyCollapsed(int commentReplyCollapsed)	
private boolean getCommentSort(int commentSort)	

PrivacyAction	
Attributes	
private String SUCCESS = "return"	
private UserPrivacy recordSet[0..*]	
private User user	
private UserPrivacy userPrivacyRecord	
private VisibilityValidation visibilityrecordSet[0..*]	
private VisibilityValidation everyoneOBJ	
private VisibilityValidation friendsOBJ	
private VisibilityValidation meOBJ	
Operations	
public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response)	
private VisibilityValidation getVisibility(int privacy)	
private void getvisibilityObjects()	
private boolean recordExists()	

RecentActivityAction	
Attributes	
private String SUCCESS = "return"	
private HttpSession session	
private User user	
private ActivityFacadeRemote activityFacadeRemote	
private URLDetailFacadeRemote uRLDetailFacadeRemote	
private ServletContext servletContext	
Operations	
public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response)	
private void saveRecentActivityContext()	

Dexter

Making search web 2.0 ready!

SearchAction	
<i>Attributes</i>	
private String USERFAILURE = "userfailure"	
private String USERSUCCESS = "usersuccess"	
private String NOUSERFAILURE = "nouserfailure"	
private String NOUSERSUCCESS = "nousersuccess"	
private String GOOGLEKEY = "search"	
private String DEXTERKEY = "searchdexter"	
private HttpSession session	
private int page	
private String escapedstring	
private ServletContext context	
private User user	
private String category	
private String timeframe	
private String visibleURL	
<i>Operations</i>	
public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response)	
private void fetchDexterResults()	
private void sortSubmittedResults()	
private void addRecord(URLDetail urlDetailRecord, KeywordURL keywordRecord)	
private void populateResultBean(URLDetail record, KeywordURL keywordRecord, SearchResultBean bean)	
private String getChannelName(URL url)	
private String getURLID(long id)	
private String getUserName(User user)	
private void fetchGoogleResults(HttpServletRequest request)	
private boolean isFilterConditionSatisfied(URLDetail record)	
private boolean checkTimeFrame(URLDetail record)	
private boolean checkCategory(URLDetail record)	
private CategoryBean[0..*] checkParentCategory()	
private boolean checkChannel(URLDetail record)	
private boolean isFilterON()	
private String getForwardFlag()	
private boolean isSearchStringEmpty()	

Dexter

Making search web 2.0 ready!

ShoutAction	
<i>Attributes</i>	
private String SENDSHOUT = "sendshout" private String SEESHOUTS = "seeshouts" private String DELETE = "delete" private String BLOCK = "block" private String UNBLOCK = "unblock" private String message private String userName private ShoutsFacadeRemote shoutFacadeRemote private UserFacadeRemote userFacadeRemote private BlockedShoutsFacadeRemote blockedShoutsFacadeRemote private Shouts shout private User from private User to private String id private HttpSession session private String editshout private User user private String urlid private String urlString	
<i>Operations</i>	
public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response) private void createShout() private void editShout() private void saveShoutContext() private void setDateString(Shouts shoutList[0..*])	

SignupAction	
<i>Attributes</i>	
private String SUCCESS = "success" private String FAILURE = "failure" private String USER = "user" private VisibilityValidation visibilityrecordSet[0..*] private UserPrivacy recordSet[0..*] private VisibilityValidation everyoneOBJ private VisibilityValidation friendsOBJ private VisibilityValidation meOBJ private HttpSession session private Date sqlDate private UserPreference userPreference private UserPrivacy userPrivacy private User userRecord	
<i>Operations</i>	
public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response) private void createDefaultPreferences() private void createDefaultPrivacy() private void getvisibilityObjects() private void populatePreferenceBean(PreferenceFormBean bean, UserPreference record) private void populatePrivacyBean(PrivacyFormBean bean, UserPrivacy record) private void saveUserContext(SignupFormBean formbean)	

Dexter

Making search web 2.0 ready!

UrlAction	
<i>Attributes</i>	
private String SUCCESS = "success"	
private URLFacadeRemote uRLFacadeRemote	
private URLDetailFacadeRemote uRLEDetailFacadeRemote	
private CommentFacadeRemote commentFacadeRemote	
private URL urlRecord	
private URLDetail uRLEDetail	
private Comment commentList[0..*]	
<i>Operations</i>	
public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response)	

Dexter

Making search web 2.0 ready!

VoteAction	
<i>Attributes</i>	
private String KEY = "search" private String page private String search private String mod private String category private String visibleURL private String timeframe private String resultID private Category categoryObject private String redirectURL private HttpSession session private ServletContext servletContext private URL url private URLFacadeRemote urlRemote private ActivityFacadeRemote activityFacadeRemote private User user private String keywords private KeywordURLFacadeRemote keywordURLRemote private KeywordActivityFacadeRemote keywordActivityRemote private String keywordID	
<i>Operations</i>	
public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response) private void createGoogleSearchResult(SearchResultBean resultBean) private void createUserVote(URL urlRecord, Activity activityRecord) private void editDexterSearchResult() private void editiDexterResult() private KeywordURL findKeywordURL(KeywordURL keywordurl) private Category getCategory() private Channel getChannelObject(String visibleUrl) private SearchResultBean getResultBean() private URL getURLEJB() private long getVoteDown() private long getVoteUp() private boolean isiDexter() private void modifyUserKeywordVote(KeywordURL keywordurlRecord, KeywordActivity activityRecord) private void modifyUserVote(URL urlRecord, Activity activityRecord) private void modifyUserVoteWithKeyword(URL urlRecord, Activity activityRecord) private void populateEJB(SearchResultBean resultBean) private void persistActivity(User user, URL url, boolean ifVotedUp) private void persistKeywordURL(KeywordURL keywordurl) private URL findURL(URL url) private void persistURL(URL url) private void populateURLTables(SearchResultBean resultBean) private void pushToDatabase(SearchResultBean resultBean) private String restructureURL()	

BEANS PACKAGE

Dexter

Making search web 2.0 ready!

CategoryBean	
<i>Attributes</i>	
<code>private String categoryname</code>	
<i>Operations</i>	
<code>public CategoryBean[0..*] getChild()</code>	
<code>public void setChild(CategoryBean child[0..*])</code>	
<code>public String getCategoryname()</code>	
<code>public void setCategoryname(String categoryname)</code>	

CommentFormBean	
<i>Attributes</i>	
<code>private URL urlRecord</code>	
<code>private URLDetail urlDetail</code>	
<code>private Comment commentList[0..*]</code>	
<code>private String comment</code>	
<code>private String success</code>	
<code>private String error</code>	
<i>Operations</i>	
<code>public String getError()</code>	
<code>public void setError(String error)</code>	
<code>public String getSuccess()</code>	
<code>public void setSuccess(String success)</code>	
<code>public String getComment()</code>	
<code>public void setComment(String comment)</code>	
<code>public Comment[0..*] getCommentList()</code>	
<code>public void setCommentList(Comment commentList[0..*])</code>	
<code>public URLDetail getUrlDetail()</code>	
<code>public void setUrlDetail(URLDetail uRLDetail)</code>	
<code>public URL getUrlRecord()</code>	
<code>public void setUrlRecord(URL urlRecord)</code>	

Dexter

Making search web 2.0 ready!

FriendRequestBean	
<i>Attributes</i>	
private User requestFrom	
private User requestTo	
private boolean ifFriend	
<i>Operations</i>	
public boolean isIfFriend()	
public void setIfFriend(boolean ifFriend)	
public User getRequestFrom()	
public void setRequestFrom(User requestFrom)	
public User getRequestTo()	
public void setRequestTo(User requestTo)	

FriendsFormBean	
<i>Attributes</i>	
private String username	
private String success	
private String error	
<i>Operations</i>	
public String getUsername()	
public void setUsername(String username)	
public String getSuccess()	
public void setSuccess(String success)	
public String getError()	
public void setError(String error)	

Dexter

Making search web 2.0 ready!

LoginFormBean	
	<i>Attributes</i>
private String username private String password private String error private String searchstring private String searchpage private String idexterpage private boolean enable	<i>Operations</i> public boolean isEnabled() public void setEnabled(boolean enable) public String getSearchpage() public void setSearchpage(String searchpage) public String getIdexterpage() public void setIdexterpage(String idexterpage) public String getSearchstring() public void setSearchstring(String searchstring) public String getUsername() public void setUsername(String username) public String getPassword() public void setPassword(String password) public void setError(String error) public String getError()

Dexter

Making search web 2.0 ready!

MoreInfoFormBean	
	<i>Attributes</i>
private String report private String success private String error private String shoutSuccess private String shoutError private String userName private String message	<i>Operations</i> public String getMessage() public void setMessage(String message) public String getShoutError() public void setShoutError(String shoutError) Unnamed public String getShoutSuccess() public void setShoutSuccess(String shoutSuccess) public String getUserName() public void setUserName(String userName) public String getError() public void setError(String error) public String getReport() public void setReport(String report) public String getSuccess() public void setSuccess(String success)

Dexter

Making search web 2.0 ready!

PasswordFormBean	
<i>Attributes</i>	
private String currentpwd	
private String newpwd	
private String confirmnewpwd	
private String error	
private String success	
<i>Operations</i>	
public String getSuccess()	
public void setSuccess(String success)	
public String getError()	
public void setError(String error)	
public String getConfirmnewpwd()	
public void setConfirmnewpwd(String confirmnewpwd)	
public String getNewpwd()	
public void setNewpwd(String newpwd)	
public String getCurrentpwd()	
public void setCurrentpwd(String currentpwd)	

PreferenceFormBean	
<i>Attributes</i>	
private int commentPageSize	
private int commentReplyCollapsed	
private int commentThreshold	
private int commentSort	
private String success	
<i>Operations</i>	
public String getSuccess()	
public void setSuccess(String success)	
public int getCommentSort()	
public void setCommentSort(int commentSort)	
public int getCommentThreshold()	
public void setCommentThreshold(int commentThreshold)	
public int getCommentReplyCollapsed()	
public void setCommentReplyCollapsed(int commentReplyCollapsed)	
public int getCommentPageSize()	
public void setCommentPageSize(int commentPageSize)	

Dexter

Making search web 2.0 ready!

PrivacyFormBean	
<i>Attributes</i>	
private int age	
private int email	
private int gender	
private int location	
private int name	
private int profilelinks	
private int shouts	
private int userid	
private String success	
<i>Operations</i>	
public String getSuccess()	
public void setSuccess(String success)	
public int getUserId()	
public void setUserid(int userid)	
public int getShouts()	
public void setShouts(int shouts)	
public int getProfilelinks()	
public void setProfilelinks(int profilelinks)	
public int getName()	
public void setName(int name)	
public int getLocation()	
public void setLocation(int location)	
public int getGender()	
public void setGender(int gender)	
public int getEmail()	
public void setEmail(int email)	
public int getAge()	
public void setAge(int age)	

Dexter

Making search web 2.0 ready!

SearchResultBean	
<i>Attributes</i>	
private String cacheUrl	
private String content	
private String title	
private String titleNoFormatting	
private String unescapedUrl	
private String url	
protected String visibleUrl	
private String ID	
private String category	
private String channel	
private long numberofreports	
private String submissiondate	
private String userFullName	
private long voteUp	
private long voteDown	
private boolean ifmodded	
private boolean ifvotedup	
private long keywordurlid	
private String tags	
<i>Operations</i>	
public String getTags()	
public void setTags(String tags)	
public long getKeywordurlid()	
public void setKeywordurlid(long keywordurlid)	
public boolean isIfvotedup()	
public void setIfvotedup(boolean ifvotedup)	
public boolean isIfmodded()	
public void setIfmodded(boolean ifmodded)	
public long getVoteDown()	
public void setVoteDown(long voteDown)	
public long getVoteUp()	
public void setVoteUp(long voteUp)	
public String getChannel()	
public void setChannel(String channel)	
public String getUserFullName()	
public void setUserFullName(String userFullName)	
public String getSubmissiondate()	
public void setSubmissiondate(String submissiondate)	
public long getNumberofreports()	
public void setNumberofreports(long numberofreports)	
public String getCategory()	
public void setCategory(String category)	
public String getID()	
public void setID(String ID)	
public String getVisibleUrl()	
public void setVisibleUrl(String visibleUrl)	
public String getUrl()	
public void setUrl(String url)	
public String getUnescapedUrl()	
public void setUnescapedUrl(String unescapedUrl)	
public String getTitleNoFormatting()	
public void setTitleNoFormatting(String titleNoFormatting)	
public String getTitle()	
public void setTitle(String title)	
public String getContent()	
public void setContent(String content)	
public String getCacheUrl()	
public void setCacheUrl(String cacheurl)	
public int compareTo(Object o)	

Dexter

Making search web 2.0 ready!

ShoutFormBean	
<i>Attributes</i>	
private String	userName
private String	message
private String	success
private String	error
<i>Operations</i>	
public String	getError()
public void	setError(String error)
public String	getMessage()
public void	setMessage(String message)
public String	getSuccess()
public void	setSuccess(String success)
public String	getUserName()
public void	setUserName(String userName)

Dexter

Making search web 2.0 ready!

SignupFormBean	
<i>Attributes</i>	
private String country private String username private String password private String confirmpassword private String firstname private String lastname private String email private String city private String website private String gender private String day private String month private String year private String file private String dayjoined private String monthjoined private String yearjoined private String error private String success private String landingpage private String activityprivacy private String CLASSIC = "classic" private String IDEXTER = "idexter" private String EVERYONE = "everyone" private String FRIENDS = "friends" private String ME = "me"	
<i>Operations</i>	
public String getYearjoined() public void setYearjoined(String yearjoined) public String getMonthjoined() public void setMonthjoined(String monthjoined) public String getDayjoined() public void setDayjoined(String dayjoined) public SignupFormBean() public String getActivityprivacy() public void setActivityprivacy(String activityprivacy) public String getConfirmpassword() public void setConfirmpassword(String confirmpassword) public String getLandingpage() public void setLandingpage(String landingpage) public String getSuccess() public void setSuccess(String success) public String getError() public void setError(String error) public String getFile() public void setFile(String file) public String getYear() public void setYear(String year) public String getMonth() public void setMonth(String month) public String getDay() public void setDay(String day) public String getGender() public void setGender(String gender) public String getWebsite() public void setWebsite(String website) public String getCity() public void setCity(String city) public String getEmail() public void setEmail(String email) public String getPassword() public void setPassword(String password) public String getFirstname() public void setFirstname(String firstname) public String getLastname() public void setLastname(String lastname) public String getUsername() public void setUsername(String username) public String getCountry() public void setCountry(String country)	

Dexter

Making search web 2.0 ready!

FILTERS PACKAGE

CheckAdmin	
<i>Attributes</i>	
private String ADMINISTRATOR = "administrator"	
<i>Operations</i>	
public void init(FilterConfig filterConfig)	
public void destroy()	
private void doBeforeProcessing(ServletRequest request, ServletResponse response)	
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)	

CheckMod	
<i>Attributes</i>	
private String USER = "user"	
<i>Operations</i>	
public void init(FilterConfig filterConfig)	
public void destroy()	
private void doBeforeProcessing(ServletRequest request, ServletResponse response)	
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)	

CheckUser	
<i>Attributes</i>	
<i>Operations</i>	
public void init(FilterConfig filterConfig)	
public void destroy()	
private void doBeforeProcessing(ServletRequest request, ServletResponse response)	
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)	

Dexter

Making search web 2.0 ready!

ForceClear	
<i>Attributes</i>	
package ServletContext context package String sslport	
	<i>Operations</i>
public ForceClear() public void init(FilterConfig filterConfig) public void destroy() public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) private void doBeforeProcessing(HttpServletRequest request, HttpServletResponse response) private String httpsURL(String origURL) private String portURL(String sslURL)	

ForceSSL	
<i>Attributes</i>	
package ServletContext context package String sslport	
	<i>Operations</i>
public ForceSSL() public void init(FilterConfig filterConfig) public void destroy() public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) private void doBeforeProcessing(HttpServletRequest request, HttpServletResponse response) private String httpsURL(String origURL) private String portURL(String sslURL, int port)	

LISTENER PACKAGE

ContextAttributeListener	
<i>Attributes</i>	
	<i>Operations</i>
public void attributeAdded(ServletContextAttributeEvent scab) public void attributeRemoved(ServletContextAttributeEvent scab) public void attributeReplaced(ServletContextAttributeEvent scab)	

Dexter

Making search web 2.0 ready!

ContextListener	
<i>Attributes</i>	
private String BBC = "news.bbc.co.uk"	
private String FACEBOOK = "www.facebook.com"	
private String EBAY = "cgi.ebay.com"	
private String PIRATEBAY = "thepiratebay.org"	
private String RAPIDSHARE = "rapidshare.de"	
private String WIKIPEDIA = "en.wikipedia.org"	
private String YOUTUBE = "www.youtube.com"	
private String EVERYONE = "everyone"	
private String FRIENDS = "friends"	
private String ME = "me"	
private VisibilityValidationFacadeRemote vvfr	
private CategoryFacadeRemote categoryRemote	
private VisibilityValidation everyoneOBJ	
private VisibilityValidation friendsOBJ	
private VisibilityValidation meOBJ	
private VisibilityValidation visibilityrecordSet[0..*]	
<i>Operations</i>	
public void contextInitialized(ServletContextEvent sce)	
public void contextDestroyed(ServletContextEvent sce)	
private void addAdmin()	
private void addChannels()	
private void createAdmin(User user)	
private void createDefaultPreferences(User userRecord)	
private void createDefaultPrivacy(User userRecord)	
private void addVisValValues()	
private void addParentCategories(Date sqlDate)	
private void addCategories()	
private void getvisibilityObjects()	

SessionAttributeListener	
<i>Attributes</i>	
<i>Operations</i>	
public void attributeAdded(HttpSessionBindingEvent se)	
public void attributeRemoved(HttpSessionBindingEvent se)	
public void attributeReplaced(HttpSessionBindingEvent se)	

Dexter

Making search web 2.0 ready!

 SessionListener
<i>Attributes</i>
<i>Operations</i>

```
public void sessionCreated( HttpSessionEvent event )
public void sessionDestroyed( HttpSessionEvent arg0 )
private CategoryBean[0..*] fetchCategory( )
private Channel[0..*] fetchChannels( )
```

UTILITIES PACKAGE

 DateUtility
<i>Attributes</i>
<i>Operations</i>

```
public String getDaysPassed( Date submissionDate )
private int calculateDifference( Date a, Date b )
```

 EJBUtility
<i>Attributes</i>
<i>Operations</i>

```
public Object lookup( String EJBReference )
```

 SearchGoogle
<i>Attributes</i>
<i>Operations</i>

```
public SearchGoogle( )
public ArrayList getArrayList( String searchString, int pageNumber, StringBuffer referrer )
public SearchResultBean[0..*] getArrayList( String searchString, int pageNumber, String visibleURL, StringBuffer referrer )
private URLConnection createConnection( String searchurl, StringBuffer referrer )
private String getSearchUrl( int start, String searchString )
private StringBuilder streamToBuilder( InputStream stream )
private SearchResultBean jsontoBean( JSONObject result )
```

Dexter

Making search web 2.0 ready!

 UserUtility	
<i>Attributes</i>	
<i>Operations</i>	
public UserPreference userPreferenceExists(User userRecord) public boolean userExists(HttpSession session) public String getUserRole(HttpSession session) public String getMonth(Integer month) public UserPrivacy userPrivacyExists(User userRecord)	

Dexter

Making search web 2.0 ready!

IMPLEMENTATION

CODE SNIPPET

SearchGoogle.java

```
package utilities;
```

```
import beans.SearchResultBean;  
import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStream;  
import java.io.InputStreamReader;  
import java.net.MalformedURLException;  
import java.net.URL;  
import java.netURLConnection;  
import java.util.ArrayList;  
import java.util.UUID;  
import org.json.JSONArray;  
import org.json.JSONException;  
import org.json.JSONObject;
```

```
public class SearchGoogle {
```

```
    ArrayList<SearchResultBean> resultBeanArray;
```

Dexter

Making search web 2.0 ready!

```
public SearchGoogle() {  
    resultBeanArray = new ArrayList();  
}  
  
public ArrayList getArrayList(String searchString, int pageNumber, StringBuffer referrer)  
throws IOException, JSONException {  
  
    /* initialize start */  
    //int pageNumber = Integer.parseInt(page);  
    int start = (pageNumber - 1) * 8;  
  
    /* Loop twice to get 2 JSON Objects  
     * each JSON Object contains 4 search results  
     * each page would return 8 search results  
     */  
    for (int i = 1; i <= 2; i++, start += 4) {  
  
        /* initialize search url */  
        String searchurl = getSearchUrl(start, searchString);  
  
        /* create a socket */  
        URLConnection connection = createConnection(searchurl, referrer);  
  
        /* get the response */  
        InputStream stream = connection.getInputStream();
```

Dexter

Making search web 2.0 ready!

```
StringBuilder builder = streamToBuilder(stream);

/* Wrap the response in JSONObject */
JSONObject json = new JSONObject(builder.toString());

/* Get StatusCode sent by Google */
int statusCode = json.getInt("responseStatus");

if (statusCode == 200) { // status(200) is OK

    /* Fetch responsedata */
    JSONObject responseData = json.getJSONObject("responseData");

    /* Ferch results */
    JSONArray results = responseData.getJSONArray("results");

    /* Iterate the results JSONArray */
    for (int j = 0; j < results.length(); j++) {

        /* Fetch a search result */
        JSONObject result = results.getJSONObject(j);

        /* Push search result details to searchresultbean */
        SearchResultBean resultbean = jsontoBean(result);

        /* Add the bean to ArrayList */
    }
}
```

Dexter

Making search web 2.0 ready!

```
        resultBeanArray.add(resultbean);

    }

}

return (resultBeanArray);

}

public ArrayList<SearchResultBean> getArrayList(String searchstring, int pagenumber,
String visibleURL, StringBuffer referrer) throws IOException, JSONException {

do {

/*initialize start*/
//int pagenumber = Integer.parseInt(page);
int start = (pagenumber - 1) * 8;

/* Loop twice to get 2 JSON Objects
 * each JSON Object contains 4 search results
 * each page would return 8 search results
*/
for (int I = 1; I <= 2; i++, start += 4) {

/*initialize search url*/
String searchurl = getSearchUrl(start, searchstring);

/* create a socket */

```

Dexter

Making search web 2.0 ready!

```
URLConnection connection = createConnection(searchurl,referrer);

/* get the response */

InputStream stream = connection.getInputStream();

StringBuilder builder = streamToBuilder(stream);

/* Wrap the response in JSONObject */

JSONObject json = new JSONObject(builder.toString());

/* Get StatusCode sent by Google */

int statusCode = json.getInt("responseStatus");

if (statusCode == 200) { // status(200) is OK

/* Fetch responsedata */

JSONObject responseData = json.getJSONObject("responseData");

/* Ferch results */

JSONArray results = responseData.getJSONArray("results");

/* Iterate the results JSONArray */

for (int j = 0; j < results.length(); j++) {

/* Fetch a search result */

JSONObject result = results.getJSONObject(j);
```

Dexter

Making search web 2.0 ready!

Dexter

Making search web 2.0 ready!

}

```
/*initialize search url*/  
private String getSearchUrl(int start, String searchstring) {  
  
    String startparam = "&start=" + start;  
    String searchparam = "&q=" + searchstring;  
    String googleapipath = "http://ajax.googleapis.com/ajax/services/search/web?v=1.0";
```

```
    String url = googleapipath + startparam + searchparam;  
    return (url);  
}
```

```
/* wrap stream to stringbuilder */  
private StringBuilder streamToBuilder(InputStream stream) throws IOException {
```

```
    BufferedReader reader = new BufferedReader(new InputStreamReader(stream));  
    StringBuilder builder = new StringBuilder();  
    String line = null;
```

```
    do { /* iterate until EOF */
```

```
        line = reader.readLine();
```

```
        if (line != null) {
```

```
            builder.append(line);
```

```
}
```

Dexter

Making search web 2.0 ready!

```
    } while (line != null);

    return (builder);
}

/* wrap jsonobject result to searchresult bean */
private SearchResultBean jsontoBean(JSONObject result) throws JSONException {

    /* Fetch details of search result */

    String cacheUrl = result.getString("cacheUrl");

    String content = result.getString("content");

    String title = result.getString("title");

    String titleNoFormatting = result.getString("titleNoFormatting");

    String unescapedUrl = result.getString("url");

    String url = result.getString("url");

    String visibleUrl = result.getString("visibleUrl");

    /* Generate random boolean */

    UUID randomUUID = UUID.randomUUID();

    String boolean = randomUUID.toString();

    /* Initialize search result bean */

    SearchResultBean resultbean = new SearchResultBean();

    /* Populate the bean */

    resultbean.setCacheUrl(cacheUrl);
```

Dexter

Making search web 2.0 ready!

```
resultbean.setContent(content);
resultbean.setTitle(title);
resultbean.setTitleNoFormatting(titleNoFormatting);
resultbean.setUnescapedUrl(unescapedUrl);
resultbean.setUrl(url);
resultbean.setVisibleUrl(visibleUrl);
resultbean.setID( oolean );
return (resultbean);
}
```

dexterAction.java

```
package actions;

import beans.CategoryBean;
import beans.SearchResultBean;
import ejb.ActivityFacadeRemote;
import ejb.URLDetailFacadeRemote;
import entity.Activity;
import entity.Category;
import entity.Channel;
import entity.URL;
import entity.URLDetail;
import entity.User;
```

Dexter

Making search web 2.0 ready!

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.Iterator;
import java.util.List;
import javax.naming.NamingException;
import javax.servlet.ServletContext;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.action.ActionForward;
import utilities.DateUtility;
import utilities.EJBUtility;
import utilities.UserUtility;

public class IdexterAction extends Action {

    private final static String KEY = "idexter";
    private String page;
    private String timeframe;
    private String category;
    private String visibleURL;
    private HttpSession session;
```

Dexter

Making search web 2.0 ready!

```
private ServletContext servletContext;  
  
ArrayList<SearchResultBean> resultList;  
  
private User user;  
  
  
@Override  
public ActionForward execute(ActionMapping mapping, ActionForm form,  
HttpServletRequest request, HttpServletResponse response)  
throws Exception {  
  
/* Get Servlet Context */  
servletContext = oolean .getServletContext();  
  
/* Get Request Parameters */  
page = request.getParameter("page");  
timeframe = request.getParameter("timeframe");  
visibleURL = request.getParameter("channel");  
category = request.getParameter("category");  
  
/* Get Current Session */  
session = request.getSession();  
  
/* Get Current User */  
user = (User) session.getAttribute("userRecord");  
  
/* Get ArrayList of Submitted ResultBeans */  
resultList = getSubmittedResults();
```

Dexter

Making search web 2.0 ready!

```
/* Sort ArrayList wrt. Votes */
sortSubmittedResults();

/* Save ArrayList in Servlet Context */
servletContext.setAttribute(KEY, resultList);

/* Get Redirect Flag */
String redirectFlag = getredirectFlag();

/* Return */
return mapping.findForward(redirectFlag);

}

private void addRecord(URLDetail urlDetailRecord) throws NamingException {

/* Create a Result Bean */
SearchResultBean resultBean = new SearchResult();

/* Populate Result Bean with Record */
populateResultBean(urlDetailRecord, resultBean);

/* Add Result Bean to ArrayList */
resultList.add(resultBean);

}
```

Dexter

Making search web 2.0 ready!

```
private boolean checkCategory(URLDetail record) {  
  
    /* Get Record Category Name */  
  
    Category recordCategory = record.getUrl().getCategory();  
  
    /* Check for Parent Category Filter */  
  
    ArrayList<CategoryBean> childCategoryList = checkParentCategory();  
  
    if (childCategoryList != null) {  
  
        if  
        (recordCategory.getParentCategory().getCategoryName().equalsIgnoreCase(category)) {  
  
            return true;  
        } else {  
  
            return false;  
        }  
  
    } else {  
  
        if (recordCategory.getCategoryName().equalsIgnoreCase(category)) {  
  
            return true;  
        } else {  
  
            return false;  
        }  
    }  
}
```

Dexter

Making search web 2.0 ready!

}

```
private boolean checkChannel(URLDetail record) {  
    if (record.getVisibleURL().equalsIgnoreCase(visibleURL)) {  
        return true;  
    } else {  
        return false;  
    }  
}
```

```
private ArrayList<CategoryBean> checkParentCategory() {
```

```
/* Get Category Bean */
```

```
ArrayList<CategoryBean> categoryList = (ArrayList<CategoryBean>)  
servletContext.getAttribute("category");
```

```
/* Get Iterator */
```

```
Iterator<CategoryBean> iterator = categoryList.iterator();
```

```
/* Iterate ArrayList */
```

```
while (iterator.hasNext()) {
```

```
    CategoryBean parentCategoryBean = iterator.next();
```

```
    if (parentCategoryBean.getCategoryname().equalsIgnoreCase(category)) {
```

```
        return (parentCategoryBean.getChild());
```

```
}
```

```
}
```

Dexter

Making search web 2.0 ready!

```
/* Category Filter Selected is Not Parent Category */

return (null);

}

private boolean checkTimeFrame(URLDetail record) {

    /* Get Days Passed Since Submission */

    String daysPassed = DateUtility.getDaysPassed(record.getUrl().getSubmissionDate());

    int days = Integer.parseInt(daysPassed);

    /* Check Timeframe */

    if (timeframe.equalsIgnoreCase("today")) {

        if (daysPassed.equalsIgnoreCase("0")) {

            return (true);

        } else {

            return (false);

        }

    }

    if (timeframe.equalsIgnoreCase("week")) {

        if (days <= 7) {

            return (true);

        } else {

            return (false);

        }

    }

}
```

Dexter

Making search web 2.0 ready!

```
}
```

```
if (timeframe.equalsIgnoreCase("month")) {
```

```
    if (days <= 30) {
```

```
        return (true);
```

```
    } else {
```

```
        return (false);
```

```
}
```

```
}
```

```
if (timeframe.equalsIgnoreCase("year")) {
```

```
    if (days <= 365) {
```

```
        return (true);
```

```
    } else {
```

```
        return (false);
```

```
}
```

```
}
```

```
    return false;
```

```
}
```

```
private String getChannelName(URL url) {
```

```
    String channelName;
```

```
    Channel channel = url.getChannel();
```

```
    if (channel != null) {
```

Dexter

Making search web 2.0 ready!

```
channelName = channel.getChannelName();  
}  
  
else {  
    channelName = null;  
}  
  
return channelName;  
}  
  
private ArrayList<SearchResultBean> getSubmittedResults() throws NamingException {  
  
/* Create ArrayList */  
resultList = new ArrayList<SearchResultBean>();  
  
/* Get Remote Object */  
URLDetailFacadeRemote urlDetailRemote = (URLDetailFacadeRemote)  
EJBUtility.lookup("URLDetailFacade");  
  
/* Get Recordset */  
List<URLDetail> urlDetailRecordset = urlDetailRemote.findAll();  
  
/* Iterate Recordset */  
Iterator<URLDetail> iterator = urlDetailRecordset.iterator();  
while (iterator.hasNext()) {  
  
    URLDetail urlDetailRecord = iterator.next();  
}
```

Dexter

Making search web 2.0 ready!

```
/* Check Filters */

if (isFilterON()) {

    if (isFilterConditionSatisfied(urlDetailRecord)) {

        addRecord(urlDetailRecord);

    }

} else {

    addRecord(urlDetailRecord);

}

}

/* Return ArrayList */

return resultList;

}

private String getURLID(long id) {

    Long iD = new Long(id);

    return (iD.toString());

}

private String getUserName(User user) {

    String firstName = user.getFirstName();

    String lastName = user.getLastName();

    String fullName = firstName + " " + lastName;

    return (fullName);

}
```

Dexter

Making search web 2.0 ready!

}

```
private boolean isFilterConditionSatisfied(URLDetail record) {
```

```
    boolean flag = false;
```

```
    if (!timeframe.equals("")) {
```

```
        flag = checkTimeFrame(record);
```

```
        if (!flag) {
```

```
            return flag;
```

```
        }
```

```
    }
```

```
    if (!visibleURL.equals("")) {
```

```
        flag = checkChannel(record);
```

```
        if (!flag) {
```

```
            return flag;
```

```
        }
```

```
    }
```

```
    if (!category.equals("")) {
```

```
        flag = checkCategory(record);
```

```
    }
```

```
    return (flag);
```

}

```
private boolean isFilterON() {
```

Dexter

Making search web 2.0 ready!

```
if (timeframe == null && visibleURL == null && category == null) {  
    return false;  
}  
else {  
    if (timeframe.equals("") && visibleURL.equals("") && category.equals("")) {  
        return false;  
    }  
    return true;  
}  
  
private void populateResultBean(URLDetail record, SearchResultBean bean) throws  
NamingException {  
  
    /* Get Remote Objects */  
    ActivityFacadeRemote activityRemote = (ActivityFacadeRemote)  
EJBUtility.lookup("ActivityFacade");  
  
    /* Get URL Record */  
    URL url = record.getUrl();  
  
    /* Check if Current User has Modded the URL */  
    if (user != null) {  
        Activity activityRecord = activityRemote.findByUserUrl(user, url);  
  
        /* Populate Activity Parameters */  
        if (activityRecord != null) {  
            bean.setActivityRecord(activityRecord);  
        }  
    }  
}  
}
```

Dexter

Making search web 2.0 ready!

```
bean.setIfmodded(true);

/* Check Vote */

if (activityRecord.isIfVotedUp0) {

    bean.setIfvotedup(true);

}

}

}

/* Populate Bean */

bean.setCacheUrl(record.getCacheURL());

bean.setContent(record.getContent());

bean.setID(getURLID(url.getId()));

bean.setTitle(record.getTitle());

bean.setTitleNoFormatting(record.getTitleNoFormating());

bean.setVisibleUrl(record.getVisibleURL());

bean.setUnescapedUrl(url.getUsescapedURL());



/* Populate iDexter Specific Parameters */

bean.setCategory(url.getCategory().getCategoryName());

bean.setChannel(getChannelName(url));

bean.setNumberofreports(url.getNoOfReports());

bean.setSubmissiondate(DateUtility.getDaysPassed(url.getSubmissionDate()));

bean.setUserFullName(getUserName(url.getUser()));

bean.setVoteUp(url.getVoteUp());
```

Dexter

Making search web 2.0 ready!

```
bean.setVoteDown(url.getVoteDown());  
}  
  
private void sortSubmittedResults() {  
    Collections.sort(resultList);  
}  
  
private String getredirectFlag() {  
  
    if (UserUtility.userExists(session)) {  
        return ("usersuccess");  
    } else {  
        return ("nousersuccess");  
    }  
}
```

Dexter

Making search web 2.0 ready!

FUTURE POSSIBILITIES

Dexter can be extended to include the following changes to make it a better product.

- Google/OpenID Authentication
- Syncing with Facebook Friends
- Inline Youtube Preview within Google Results
- Adding Yahoo/Live Search
- ‘DexterBar’ – A toolbar that would solve submission issue.
- Widget for Windows/OSX
- RSS/Atom Support
- iPhone and Android App for Mobile Support
- Email Notification for Shouts/FriendRequests using JavaMail
- ‘Did you mean?’ support in Google search
- ‘Digg style Suggestions’ for related keywords

Dexter

Making search web 2.0 ready!

CONCLUSION

Dexter is a social search engine, bookmarking site, content recommendation engine and a social networking site all rolled into one. As it amalgamates so many features Dexter has the potential of becoming your web home.

Search Result are sorted according to the votes every URL receives. Result once voted up stay in your profile for future reference. You can add friends and collaborate in search as URL they are voting up are visible to you.

The iDexter Module allows user to surf the URL which has been submitted into the database thus you get quality links to surf. You can filter results to suit your preferences. It can be a great tool when you just want to explore the internet as the URL which have high vote up are essentially recommended by so many people.

In addition to the other features Dexter allows you to do social networking. You can send messages to your friends see what they are searching and in effect stay connected.

In the end I would like to add a word of caution- Dexter like all social networking sites is bounded by a need to have a minimum critical mass of users. Without having a substantial user base the search results and the recommendation can become lopsided and easily manipulated but as it would achieve the critical mass effects to manipulate the results would yield no result.

Dexter

Making search web 2.0 ready!

APPENDIX A: GLOSSARY

Dexter – A Java EE Web Application based upon Struts and Hibernate Framework building an overlay on top of conventional Web 1.0 search engines to provide human collaboration and participation possibility

Dexter Surf - One of the two possible environments of interaction with the web-app. Surf provides content suggestion based upon popularity of voted search results in a particular timeframe, category and channel.

Dexter Search – One of the two possible environments of interaction with web-app. Search restricts to the classic search scenario waiting for the user to push through a keyboard for relevant response. Search being simplistic to Surf is considerably faster and recommended for normal search scenarios.

Channel – Filter the search results based on a particular website (Wikipedia, Orkut) or a media type (images/videos)

Category – Filter the search results based on field of interest (technology, Fun, Offbeat)

Timeframe – Filter the search results falling in a particular week, month or year.

Votes – A method for the user to participate and help in providing the apparent usefulness of a search URL by voting up or down. Votes eventually affect the positioning of the search URL the next time a search for the same keyword is done.

URL Submission – A possibility to submit an arbitrary URL a user stumbles upon to when browsing and would want the Dexter database to have, being unsure of which keyword the search engine would show the URL against.

Shouts – Messages sent from one Dexter user to another. (Not Instant Message)

Favorite – Bookmark a search result for later.

Tag – keywords stored against a favorite for effective retrieval.

Report Comment – comments could be reported on profanity to the moderator for possible deletion.

Dexter

Making search web 2.0 ready!

Report URL – Search result URL's could be reported on profanity or incorrect category submission to the moderator for possible actions.

Dexter

Making search web 2.0 ready!

APPENDIX B: BIBLIOGRAPHY

We referred the following books during the implementation of the project ‘Dexter’

- Thinking in Java by Bruce Eckel
- Apress Pro JSP 2/2.1
- Hibernate in Action by Christian Bauer and Gavin King
- Struts in Action by Cedric Dumoulin and Ted Husted
- Java: The Complete Reference by Herbert Schildt