The Florida State University
College of Arts and Science

# Yahoo Database Project
# Super Search Application

By
Xi Rao

July 2002

A masters project submitted to the
Department of Computer Science in partial fulfillment of
For the Degree of Master of Science

**Major Professor: Dr. D.G. Schwartz**

# Table of Contents

# ACKNOWLEDGEMENTS

First and foremost, thanks to my major professor Dr. D.G. Schwartz. Without his help, I couldn't have completed this project.

Thanks to everyone at the Computer Science Department, Florida State University. They taught me and helped me to earn my degree.

# Abstract

The Super Search Application is a Java/Swing/JDBC-based application that provides the user the capability to download all the html pages from a website, parse these to get the parent-child linkage information, and insert this into a database. It also provides a user interface for searching the data after it has been put into this form.

The test case is the well-known Yahoo website, where the resulting database contains all the classification, but the same technique can be applied to any other website that is structured similarly. The reason for focusing on the Yahoo website in this project was to provide a test bed of data that can be used in future projects concerned with 2D and 3D graphical displays of concept taxonomies. The Yahoo classification index is a typical such taxonomy being used for document classification.

The project was carried out using state-of-the-art software engineering (SE) techniques. This is a light weight rational united process (RUP). Originally, RUP has very heavy documentation guidelines, which is not fit the small project. So, I decide to define my own process, which is based on the RUP. I call it light weight RUP process.

The Super Search Application use layer technology to do development. The system has a GUI layer, a Data Service layer, an Event layer, and a Parser layer. Each layer has its own responsibility, and there are rules specified for communication between layers. The layering technology greatly increases the extensibility of the system, while decreasing the maintenance costs. The Super Search Application also uses numerous design patterns. Layering together with design patterns are essence of modern day Object-Oriented Programming.

The Super Search Application use Java Swing to implement the GUI. It uses Swing's built in parser to parse the html files, and it uses the freeware MySql as the database to store all the information.

# CHAPTER 1
# PROCESS

One well-known software engineering processes is the rational united process (RUP). RUP is very good for large projects, but it has heavy documentation guidelines, which do not fit well for smaller projects. So, I decide to define my own light weight RUP process to decrease the documentation burden.

The overall flow-through the process is described below. This may run either sequentially or iteratively through each stage. Some stages may be combined or performed in parallel.

Overview of Process Activities:

- Requirement development
- Scenario development
- Analysis and Design
- Combined review
- Code and Unit test
- Application test

Requirement--A high-level description of what the application is intended to do. This appears in a Vision document.

Scenario--A high-level description of interactions with application. Each scenario has a Use Case document.

Analysis--High level, domain-recognizable models of the object and responsibilities of the framework solution.

Design--Detailed model of objects and responsibilities of the classes that are used to implement the analysis results. This appears in a Software Architecture document.

# CHAPTER 2
# VISION

**Revision History**

| Date | Version | Description | Author |
|---|---|---|---|
| 09/01/2001 | 1.0 | Initial release | Xi Rao |
| 06/12/2002 | 2.0 | Revised | Xi Rao |
| | | | |
| | | | |

# Table of Contents

# Vision

## 1 Introduction

The purpose of this document is to define the high-level requirements of the Yahoo Database Project (Super Search system) in terms of the needs of the end users.

### 1.1 References

Applicable references are:

1. Use Case document
2. Requirement management plan

## 2 Positioning

### 2.1 Problem Statement

Currently, many people use Yahoo or other portal web sites to search for information on the Internet. There are two popular ways for doing this. One is the keyword search approach employed by services such as AltaVista and Google. The other is the structured index approach, where the users drills down through a hierarchical classification system such as employed by services like Yahoo. This project concerns that latter.

The primary objective is to provide a test bed that can be used in future projects dealing with 2D and 3D visual displays of concept taxonomies for use in indexing large digital libraries. For the sake of the exercise, however, searching and viewing tools are provided here as well.

In the Yahoo index it happens that there can be more than one pathway from the root down to the same item of information (html reference link). That means that the item is implicitly cross-referenced.

Most of time, when we find such information, however, the portal web site doesn't give the cross-references. This is a drawback, since cross-references can often provide important information. In particular, it can help the user to understand that how the same information may be used by other people.

The goal of this project is to capture the category classification information, including cross-references, from the Yahoo web site. The information is to be put into a database in such a way that the parent-child links between categories are explicitly represented. The project is also to provide an interface through which the user can execute keyword searches directly on the database.

Thus the system should perform an analysis the Yahoo web site. It should determine the relations between classification categories and it should also get the URL from the html files and determine the relationship of the URL with its category. When the information is obtained, it is put into a relational database. The system should provide functions for user to search the database and view the cross-referencing information.

| | |
|---|---|
| The problem of | Find the cross reference |
| Affects | User |
| the impact of which is | User can't get cross-reference from yahoo web site. |

| | |
|---|---|
| A successful solution would be | The user can find cross-reference from our product. |

### 2.2 Product Position Statement

| | |
|---|---|
| For | Current yahoo user |
| Who | User needs it to search URL. |
| The Super Search System | Is a tool |
| That | Enable finding the cross reference function |
| Our product | Could have more information than yahoo web page. The user find URL by searching as well the cross reference |

## 3 Stakeholder and User Descriptions

This section describes the users of Super Search System.

### 3.1 Stakeholder Summary

| Name | Description | Responsibilities |
|---|---|---|
| Professor | Professor | Responsible for project approval. Monitors project progress. Ensure that the system meets his need. |

### 3.2 User Summary

| Name | Description | Responsibilities | Stakeholder |
|---|---|---|---|
| User | Any people | Manage the database and use it to search and view cross-reference | Self-represented |

### 3.3 User Environment

The user needs to use MS Front Page 2000/XP to download the Yahoo web site information into the local file system.

The user needs to install the Mysql database software.

The application can run on Windows 95/98/NT/2000/XP

### 3.4 Key Stakeholder or User Needs

| Need | Priority | Concerns | Proposed Solutions |
|---|---|---|---|
| Login | Medium | Security | User name and password needed for authentication |

| | | | |
|---|---|---|---|
| Build Database | High | Data for searching | Parse the html file to get the data. Having a parser to get URL and description out and insert them into database. |
| Search Database | High | Searching | Get the user input key word and build the sql query to search the database. |
| View cross-reference | High | Viewing | View the cross references for a specific URL |

### 3.5 Alternatives and Competition

The user community was unaware of any viable alternatives or off-the-shelf solutions.

## 4 Product Overview

This section provides a high level view of the Super Search system capabilities.

### 4.1 Product Perspective

In general, the goal is to get the information from the portal web site. The user can execute searches on the local database. It should be faster than executing the same search on the Internet. The user also can get the cross-referencing information from this product.

- Fast search on the local database
- View the cross-reference
- Build test-bed for future knowledge-base projects

### 4.2 Assumptions and Dependencies

The user need to use MS Front Page 2000/XP to download the Yahoo category web site for building the local database.

More is explained in the User Guide Manual

## 5 Product Features

### 5.1 Login

1. User Login required

   User must input the user name and password correctly to login to the system.

2. Built in user name and password

   The user name and password are predefined. The user can't change the user name or password. The user can't add new users.

### 5.2 Build Database

1. Provide an interface to permit the user to input the root directory from which the information extraction will begin.

   The user can input the directory, which contains the starting HTML file. Inputting the root directory is required for building the database.

2. Recursively access the subdirectories.

   The program can recursively access directories under the root directory, which the user input.

3. Get child category names.

   The program will store the directory relationships with as parent-child relations. Here, "category" and "directory" have the same meaning.

4. Find the URL and its description from the HTML file.

5. Build the database to keep the URL and Category information

### 5.3 Search Database

1. Provide an interface for user to input a keyword.

   The user can input one keyword for searching the database.

2. Search the Database.

The system will get the user's input keyword and build a query to search the database. The current version only supports one-keyword queries.

3. Display the result in another window inside the table.

   A table will be used to display the search result. The table will have three columns: URL, description, and category.

### 5.4 View Cross-reference

1. Permit the user to choose a URL from the table

   User can choose one URL from the table.

2. Display the cross-references in another window inside the table.

   A table will be used to display the cross-references. The table will have three columns: URL, description, and category.

## 6 Other Product Requirements

N/A

# CHAPER 3
# USE CASE: LOGIN

**Revision History**

| Date | Version | Description | Author |
|---|---|---|---|
| 08/02/2002 | 1.0 | Initial Release | Xi Rao |
| | | | |
| | | | |
| | | | |

# Table of Contents

# Use-Case Specification: Login

## 1 Use-Case Name

### 1.1 Brief Description

This use case describes how a user logs into the Yahoo Database System. The actors starting this use case are users.

## 2 Flow of Events

The use case begins when the actor types the user name and password on the login form.

### 2.1 Basic Flow

1. The user input the user name and password.
2. The system validates the user name and password. The system decides that is a valid password.
3. The system displays the Main Form and user ends

### 2.2 Alternative Flows

2.2.1 invalid Name/Password

1. The user input the user name and password.
2. The system validates the user name and password. The system decides it is invalid password.
3. The system let user input user name and password.

## 3 Special Requirements

There are no special requirements associated with this use case.

## 4 Preconditions

The user must have a user id and password.

## 5 Postconditions

There are no Postconditions associated with this use case.

## 6 Extension Points

There are no extension points associated with this use case.

# CHAPTER 4
# USE CASE: BUILD

**Revision History**

| Date | Version | Description | Author |
|---|---|---|---|
| 08/03/2001 | 1.0 | Initial Release | Xi Rao |
| | | | |
| | | | |
| | | | |

# Table of Contents

# Build Database Use Case

## 1 Use-Case Name

### 1.1 Brief Description

This use case allows the user to build the database from scratch. The system will parse the html file to get the URL, its description and the path to reach the URL. The system will insert the results it gets from the html files into the database.

## 2 Flow of Events

The system begins when the user selects the “build” menu from the main form.

### 2.1 Basic Flow

1. The user inputs the root directory, which contains the html file for parsing.
2. The user presses the “build” button.
3. The system begins the building of the database.
4. The system will notify the user build finish. The record is in the database.

### 2.2 Alternative Flows

#### 2.2.1 The user inputs the wrong directory name

1. The user input the root directory, which don’t contain the html for parsing.
2. The user presses the “build” button.
3. The system begins the building of the database.
4. The system will notify the user build finish, but no record in the database.

#### 2.2.2 The user clicks the “browse” button

1. The user click “browse” button.
2. The system will pop up a window to let the user browse the directory structure of the file system and the files in the directory. The user can’t choose the directory from the window.

#### 2.2.3 The user clicks the “back” button

1. The user click “back” button
2. The system will go back the main menu form

## 3 Special Requirements

There are no special requirements associated with this use case.

## 4 Preconditions

### 4.1 Login

The user must be logged onto the system in order for the use case to begin.

## 5 Postconditions

The html files exist.

## 6 Extension Points

There are no extension points associated with this use case.

# CHAPTER 5
# USE CASE: SEARCH

## Revision History

| Date | Version | Description | Author |
|---|---|---|---|
| 09/04/2001 | 1.0 | Initial release | Xi Rao |
| | | | |
| | | | |
| | | | |

# Table of Contents

# Use-Case Specification: Search Use Case

## 1 Use-Case Name

### 1.1 Brief Description

This use case allows the user input a keyword for searching the URLs. The system will return the search results.

## 2 Flow of Events

The system begins when the user selects the “search” menu from the main form.

### 2.1 Basic Flow

1. The user inputs the keyword into a text field, which is used to search the database.
2. The user clicks the “search database button”.
3. The system will return the search results in another window. The search results will have the URL, URL description, and category.

### 2.2 Alternative Flows

#### 2.2.1 No results

1. The user input the keyword in a text field, which is used to search the database.
2. The user click the “search database button.”
3. The system searches for the keyword in the database, but it doesn’t find a match. Another window will show, but there will be no search result.

#### 2.2.2 The user clicks the “back” button.

1. The user clicks the “back button”.
2. The system will go back to the main menu.

## 3 Special Requirements

There are no special requirements associated with this use case.

## 4 Preconditions

### 4.1 Login

The user must be logged onto the system in order for the use case to begin.

### 4.2 Database exist

The database must exist. If the database doesn’t not exist, this use case requires the user to build the database before executing this use case.

## 5 Postconditions

### 5.1 Search result

The search result will be shown in another window, which will enable the user to view the cross-references.

## 6 Extension Points

There are no extension points associated with this use case.

# CHAPTER 6
# USE CASE: VIEW

## Revision History

| Date | Version | Description | Author |
|---|---|---|---|
| 09/09/2001 | 1.0 | Initial release | Xi Rao |
| | | | |
| | | | |
| | | | |

# Table of Contents

# Use-Case Specification: View Use Case

## 1 Use-Case Name

### 1.1 Brief Description

This use case allows the user to choose a URL from the search results. The user can view the cross-references of this URL.

## 2 Flow of Events

### 2.1 Basic Flow

1. The user chooses a URL from the search results table.
2. The user clicks the “view cross reference button”.
3. The system will open another window to show the cross-references.

### 2.2 Alternative Flows

#### 2.2.1 No cross reference

1. The user chooses a URL from the search results table.
2. The user clicks the “view cross reference button”.
3. The system fail to find a cross-reference. It will say this in the view cross reference window.

#### 2.2.2 The user clicks “close” button.

The user clicks the “close” button. The system will close the view cross-reference window.

## 3 Special Requirements

There are no special requirements associates with this use case.

## 4 Preconditions

### 4.1 Login

The user must be logged onto the system in order for the use case to begin.

### 4.2 Database exist

The database must exist. If the database is not exist, this use case requires the user to build the database before executing this use case.

### 4.3 Search result exist

The user must search for URLs using the search function of the system.

## 5 Postconditions

There are no Postconditions associated with this use case.

## 6 Extension Points

There are no extension point associated with this use case.

# CHAPTER 7
# SOFTWARE ARCHITECTURE

**Revision History**

| Date | Version | Description | Author |
|---|---|---|---|
| 03/13/2002 | 1.0 | Initial Release | Xi Rao |
| 06/15/2002 | 2.0 | Revised | Xi Rao |
| | | | |
| | | | |

# Table of Contents

# Software Architecture Document

## 1. Introduction

### *1.1 Purpose*

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions that have been made for the system.

### *1.2 Scope*

This Software Architecture Document provides an architectural overview of the Super Search system.

### *1.3 Definitions, Acronyms and Abbreviations*

N/A

### *1.4 References*

Rational Rose SoDA's Software Architecture Document template.

## 2. Architectural Representation

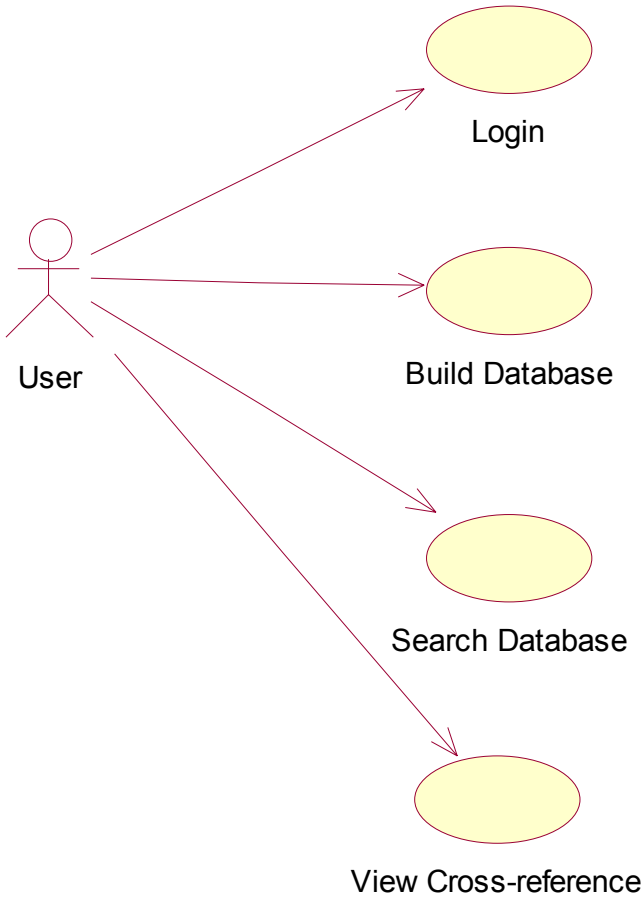
This document presents the architecture as a series of views: use case view, logical view, process view, and deployment view. There is no separate implementation view described in this document. These are views of an underlying Unified Modeling Language (UML) model developed using Rational Rose.

## 3. Architectural Goals and Constraints

A. The application should be layered, and have clear separation among presentation, control, and data storage.
B. Throughout the design, we should follow various patterns (standard object-oriented design pattern).

## 4. Use-Case View

The Use Case View is important input to the selection of the set of scenarios and/or use cases that are the focus of iteration. It describes the set of scenarios and/or use cases that represent some significant, central functionality. It also describes the set of scenarios and/or use cases that have a substantial architectural coverage (that exercise many architectural element) or that stress or illustrate a specific, delicate point of the architecture.




The use cases are:

- Login
- Build database
- Search database
- View cross-reference

The user initiates these use cases.

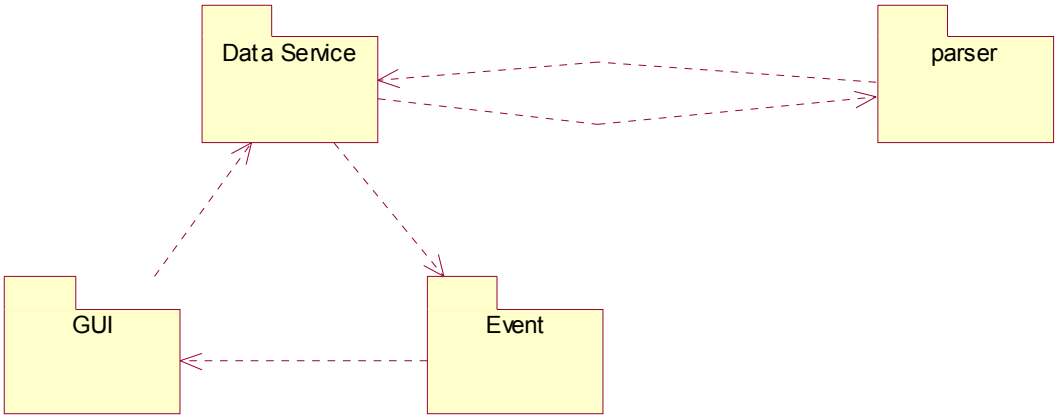
Reference the use case document.

## 5. Logical View

Logical view is a description of the logical view of the architecture, which describes the most important classes, their organization into service package and subsystems, and the organization of these subsystems into layers. Also describes the most important use-case realizations, examples, and the dynamic aspects of the architecture. Class diagrams may be included to illustrate the relationships between architecturally significant classes, subsystems, packages and layers.

The logical view of the course registration system is comprised of the 4 main packages: GUI, Data Service, Event, and Parser.

## 5.1 Overview




The logical view of the system is comprised of the 4 main packages: GUI, Data Service, Event, and Parser.

The GUI package (User Interface) contains classes for each of the forms that the actors use to communicate with the System. Boundary classes exist to support login.

The Data Service package contains the classes to map to the database table and the classes to regular operations on the database. The main goal of this package is to communicate with the database and provide the response to the GUI.

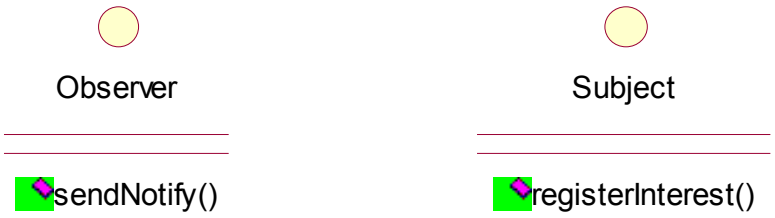
The Event package contains the events for communicating between the Data Service and GUI.

The Parser package contains the classes to parse the HTML files. It is to communicate with the Data Service package. The Data service package will tell it where to parse the file. The parser package will generate the data and return these to the data service package for insertion into the database.

Here, each package can be viewed as a layer.

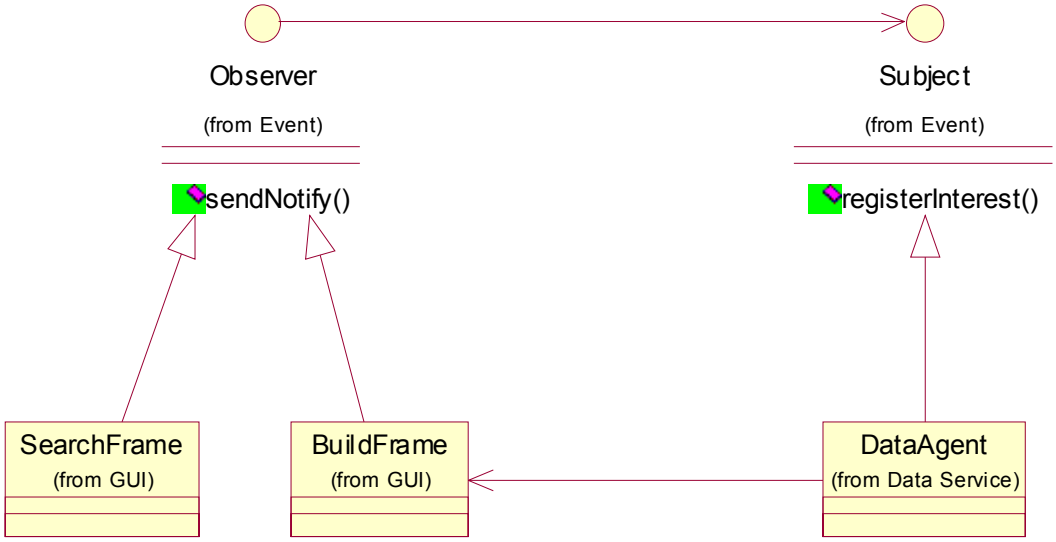
## 5.2 Architecturally Significant Design Packages

### 5.2.1 Event layer



Interface Observer and Interface Subject reside in this layer.

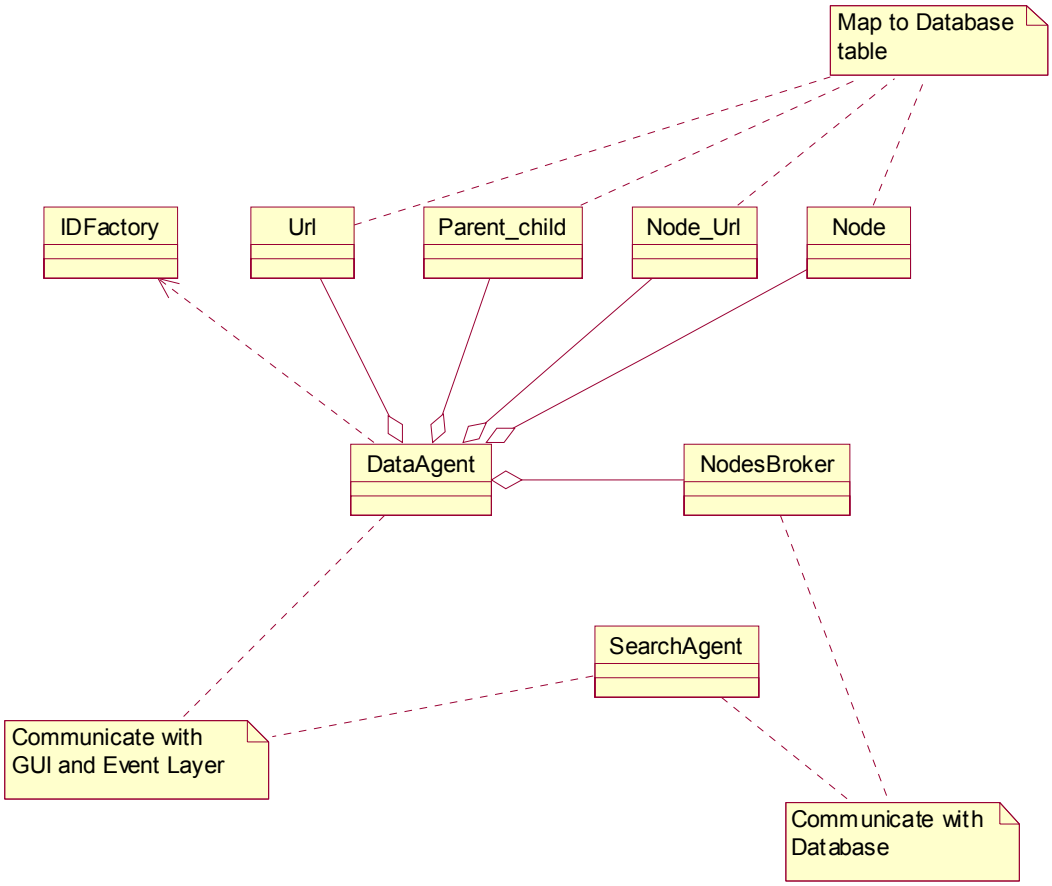
The class that implements the Subject interface will implement a “registerInterest()” method, which will have a parameter (Observer) and register itself into Observer. The class that implements the Observer interface will implement a “sendNotify()” method, which will have a parameter (String) and notify any registered Object when a specific event happens. This design is the Observer pattern.




From this diagram, we can see that the DataAgent class implements the Subject interface. Its “registerInterest()” method will register any class that implements the Observer interface, so the BuildFrame class is registered. The BuildFrame class implements the Observer interface. Its “sendNotify()” method will do some real work when it is invoked. BuildFrame knows the DataAgent, but DataAgent don’t know the BuildFrame. So, the BuildFrame will let the DataAgent handle the database task. When the DataAgent finishes the task, it will notify the BuildFrame by using Observer pattern.

The benefit is that we separate the two layers. No matter how we change the GUI and Data Server layer. It won’t affect these two layer.  It also is an asynchronization message implementation.

### 5.2.2 Data Service layer



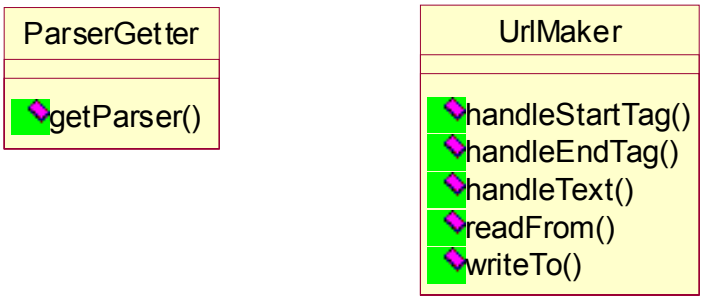

Url, Parent_child, Node_Url, and Node class are entity classes, which map to the database tables. They provide some get and set methods to access the internal data.

The DataAgent class will communicate with the GUI layer for building the database. It will use the parser layer to get the data and put them into the four entity classes. DataAgent will use the NodesBroker class to insert the data in the entity class into the database.

The IDFactory class will be used by the DataAgent class for detecting duplicate URLs and generating a unique number for every URL and Category instance.

The SearchAgent class will communicate with the GUI layer and the database. It will search the database for a specific query.

### 5.2.3 Parser layer

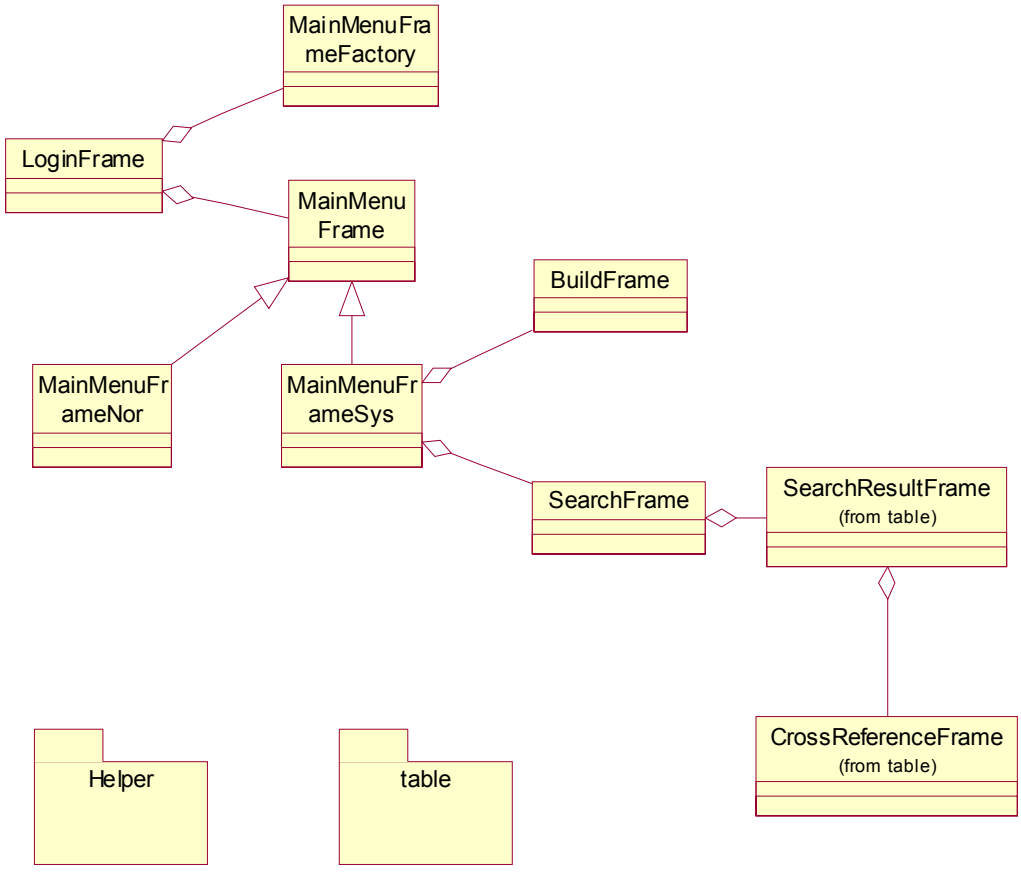



The UrlMaker class will read the html file and parse it to get the URL and its description as well as the current category and upper level category.

The ParserGetter class will return the UrlMaker instance for parsing the html document.

The parser layer is built on top of a swing html parser, which has the ability to handle standard html3.2 tags.

### 5.2.4 GUI layer




The LoginFrame class will show the login form for the client.

The MainMenuFrameFactory class will create the MainMenuFrameNor instance or MainMenuFrameSys instance depending on the client's input. This function is not

fully implemented. At present, there is only one kind of user, i.e., there is no difference between a system administrator and a normal user. The full functionality may be implemented in a future iteration. This design exists for scalability purposes.

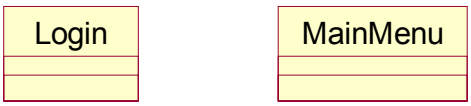
The MainMenuFrameNor class is not fully implemented. It exists for scalability purposes.

The MainMenuFrameSys class will show the menu form to the client. From this frame, it can go the BuildFrame and SearchFrame classes.

The BuildFrame class will show the build form to the client. It knows the DataAgent class. It will forward the client's build database request to DataAgent.

The SearchFrame class will show the search form to the client. It knows the SearchAgent class. It will forward the client's database search request to Search Agent.

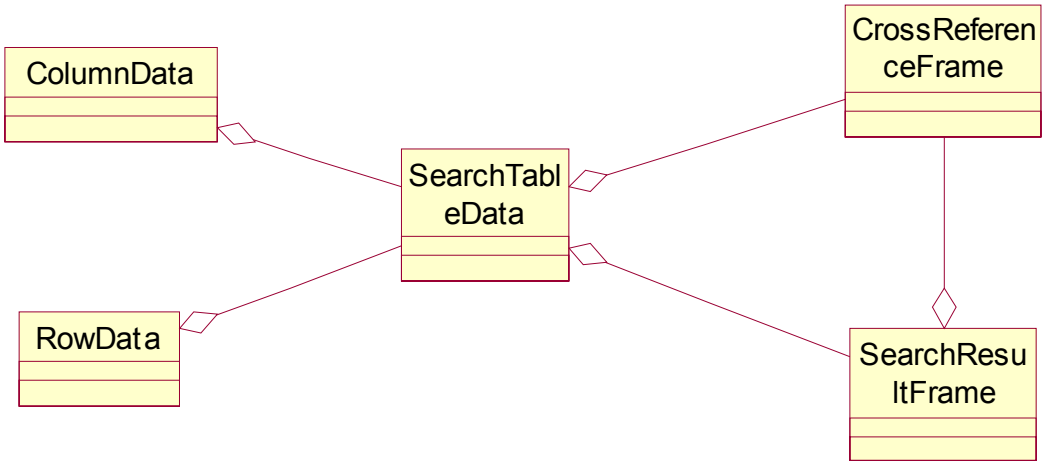
Helper package: holds some helper classes.




The Login class holds the built in user name and password. This is not a very secure way to do password authentication. It could be change in the future, such as to access an LDAP to verify the user name and password.

The MainMenu class is used by MainMenuFrameSys to create the DataAgent.

Table Package: holds the class for tables.




The SearchResultFrame class displays a form for searching for result using the JTable widget. It can go the CrossReferenceFrame.

The CrossReferenceFrame class displays a form for cross-references using the JTable widget.

The ColumnData class holds the column information for the JTable.

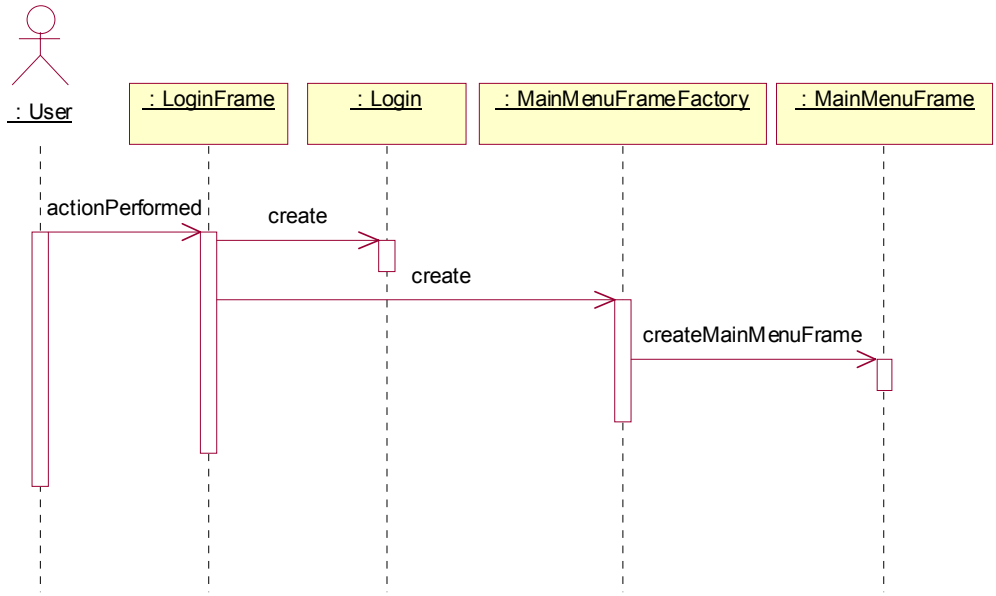
The RowData class holds the row data for the JTable.

The SearchTableData classes extend from the swing AbstractTableModel and hold the data for JTable. It includes ColumnData and RowData instances and is used by CrossReferenceFrame and SearchResultFrame.

# 6. Process View

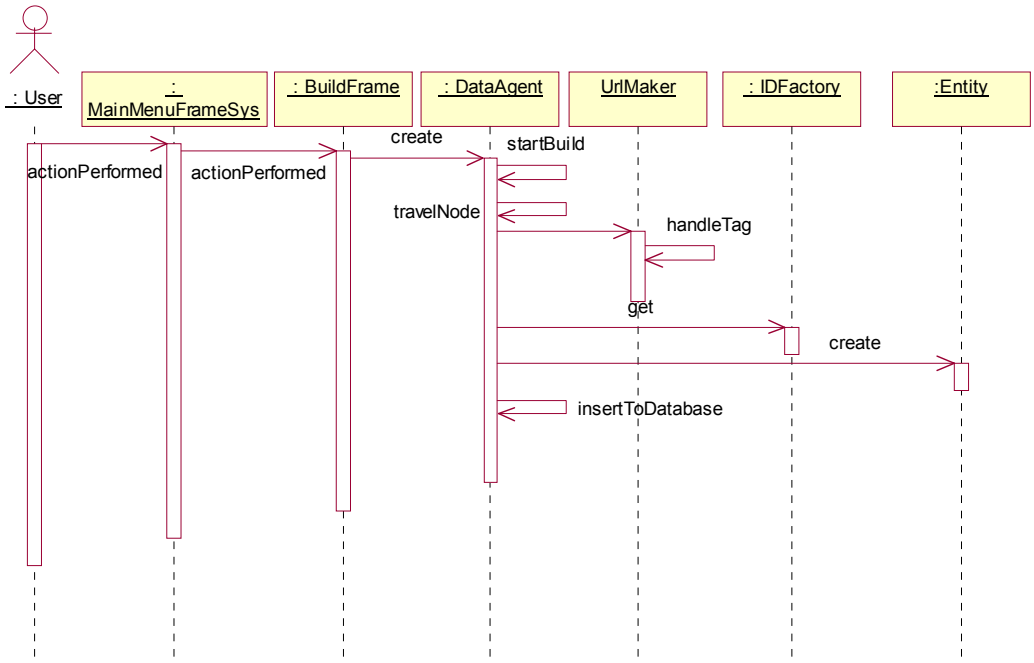
In the sequence diagram, the "create" label means to initialize an instance. The entity instances represent the four class instances (Url, Parents-child, Node_Url, and Node).

### 6.1 Login



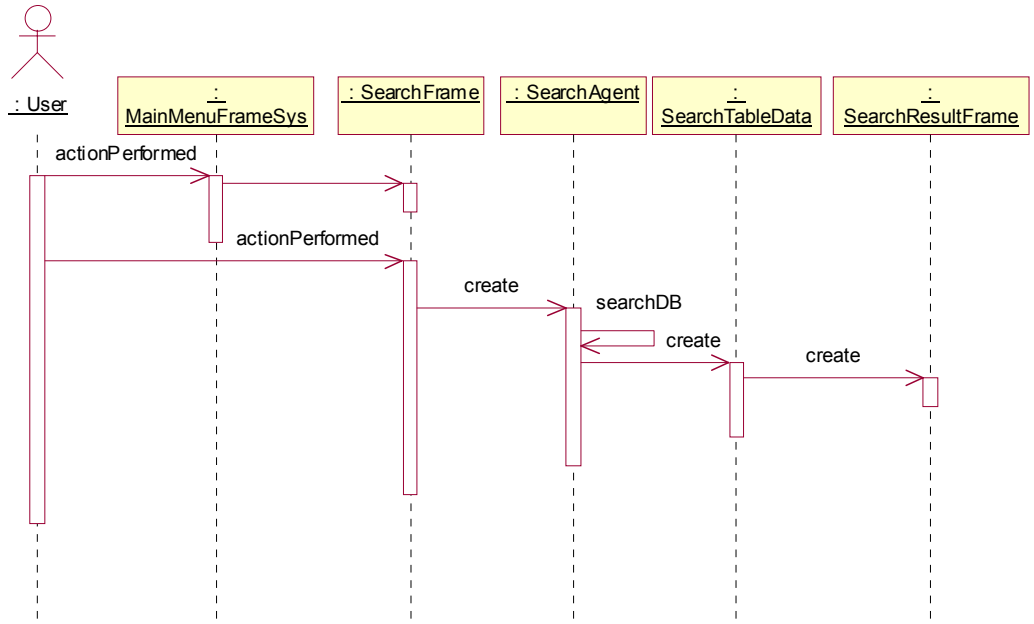

The user inputs the user name and password and presses the button to trigger the actionPerformed() method. The LoginFrame will be responsible for creating a login instance and passing it to create the MainMenuFrameFactory class, which will return either MainMenuFrameSys or MainMenuFrameNor based on the input (Login instance) by calling the createMainMenuFrame() method. The MainMenuFrameSys and MainMenuFrameNor inherit from MainMenuFrame class. This is using a Factory Pattern.

### 6.2 Build database



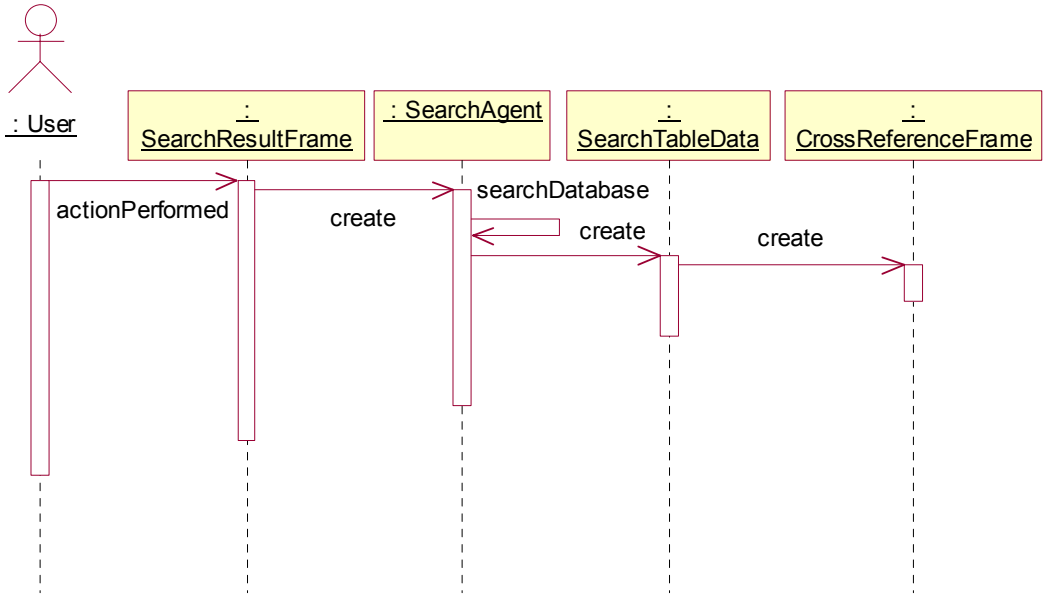

The user chooses the menu by pressing the button to trigger the MainMenuFrameSys's actionPerformed() method. The user will input a root directory for building and press the button to trigger the BuildFrame's actionPerformed() method. This will create the DataAgent. The DataAgent will call its startBuild() method to start building. The DataAgent will call its travelNode() method to recursively travel through the directories under the root directory and access the html files. The DataAgent will give the html files to the UrlMaker to let it parse them. The UrlMaker class will be responsible for getting the url and description from the html file. The DataAgent also holds the current directory name and parent directory name. The DataAgent will ask the IDFactory to assign a unique ID for each Url and Category. After that, the DataAgent will put all information into the Entity class and insert them into the database by using insertToDatabase() method.

**6.3 Search database**




The user chooses the menu by pressng the button to trigger the MainMenuFrameSys's actionPerformed() method. The user inputs the keyword for searching and presses the button to trigger the SearchFrame's actionPerformed() method. The SearchFrame will create a SearchAgent instance. It will use the keyword to build an SQL query to search the database and create a SearchTableData instance for the results. The SearchTableData will create the SearchReferenceFrame for displaying the results.

**6.4 View cross-reference**



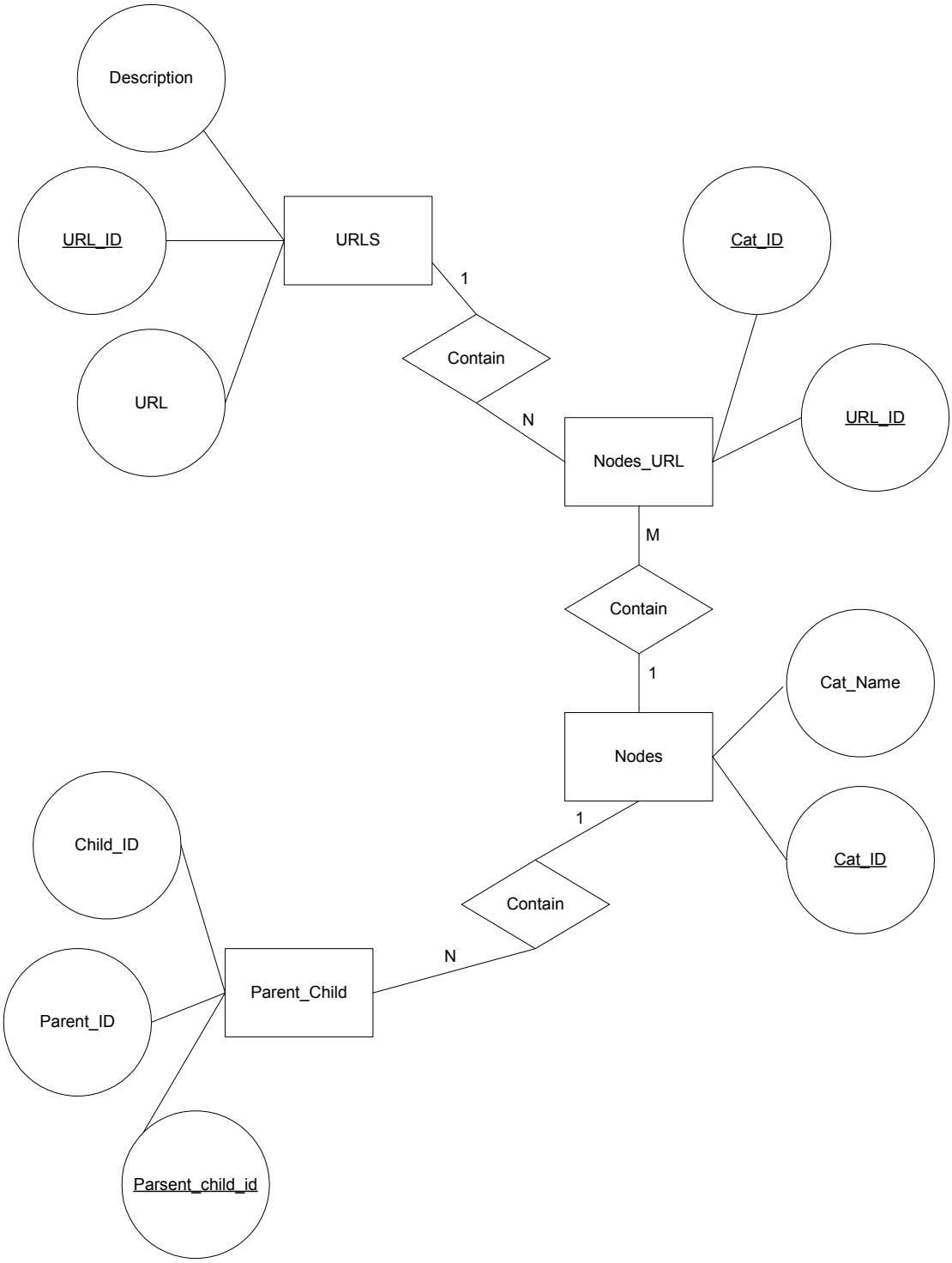

The user chooses one URL from the table in the SearchResultFrame and triggers the actionPerformed() method in the SearchResultFrame. This will create a SearchAgent to build a query and search the database again. The SearchAgent will create a

SearchTableData instance for the results. The SearchTableData instance will create a CrossReferenceFrame for displaying the results.

# 7. Deployment

The Super Search system is a typical 3-tier system containing the GUI, data service, and database. These 3 tiers live on the same machine for the current implementation.

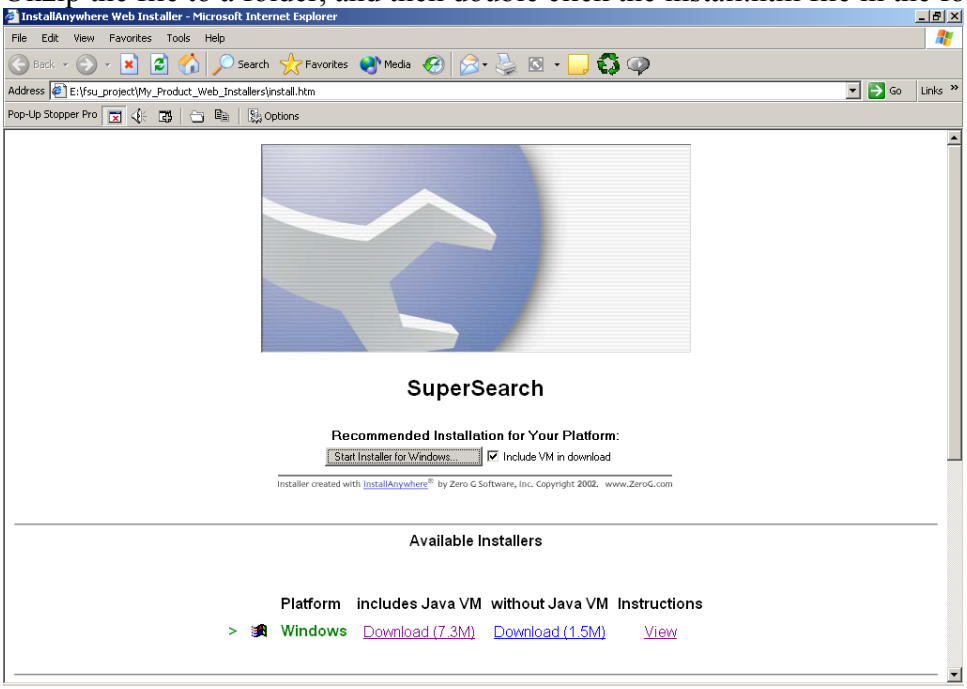
# 8. ER-diagram



# CHAPTER 8
# USER GUIDE

Section 1 – Installation

Unzip the file to a folder, and then double click the install.htm file in the folder.




From this screen, you can press the start installer button. If you don't have java on your machine, you can check the "include VM" choice. Even if you have the JDK on your local machine, you still can chose include VM. this will not affect your already existing JRE. The VM version is 1.3.1. After that, follow the instructions; you can easily install this application.

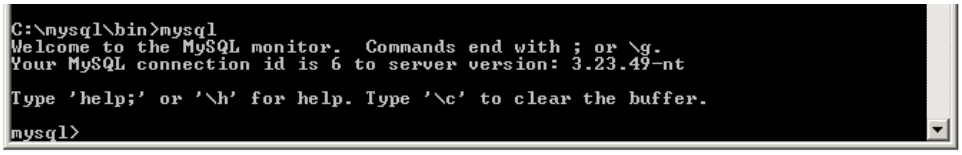
Section 2 – Install MySql Database

The Super Search application requires a database engine. You need to install the MySQL database on your local machine. For this you can go to www.mysql.com to download the MySql for windows version.

After you install MySql, you will find the icon 

on your right lower side. Then you need to copy all the sql files under SuperSearch\sql to the mysql\bin directory.

Section 3 – Configure MySql




Go to the MySql directory and run the MySql command.

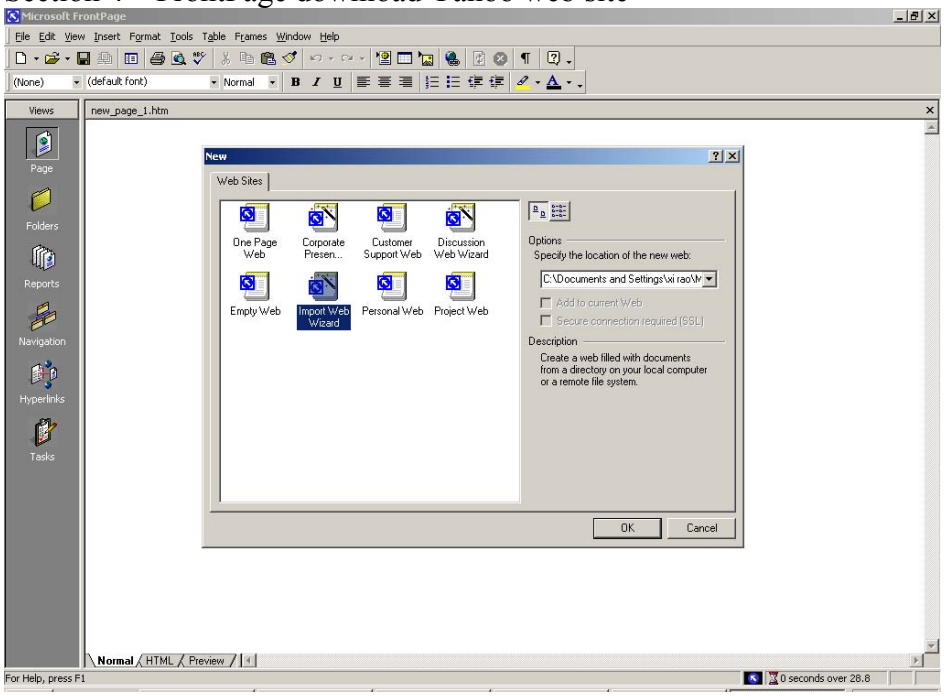
```
C:\mysql\bin>mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7 to server version: 3.23.49-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> create DATABASE samp_db;_
```

Create the database for Super Search Application. You can type the exit command to quit the MySql command interpreter.

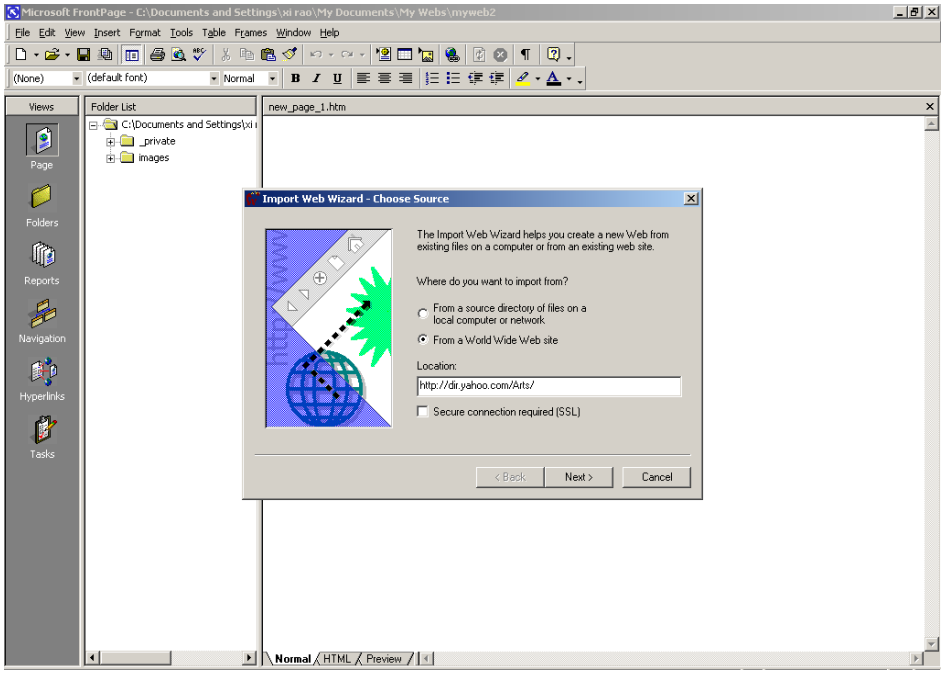
```
C:\WINDOWS\System32\cmd.exe
Bye

C:\mysql\bin>dir *.bat
 Volume in drive C has no label.
 Volume Serial Number is B815-6B7B

 Directory of C:\mysql\bin

03/09/2002  11:44 AM               167 create_all.bat
               1 File(s)            167 bytes
               0 Dir(s)   1,046,450,176 bytes free

C:\mysql\bin>create_all.bat

C:\mysql\bin>mysql samp_db 0<drop_all.sql

C:\mysql\bin>mysql samp_db 0<create_NODES.sql

C:\mysql\bin>mysql samp_db 0<create_NODES_URLS.sql

C:\mysql\bin>mysql samp_db 0<create_URLS.sql

C:\mysql\bin>mysql samp_db 0<create_PARENT_CHILD.sql

C:\mysql\bin>_
```

Run create all.bat file to create tables inside samp_db database for the Yahoo Super Search use. The create_all.bat and other .sql files are copied from the SuperSearch\sql directory.

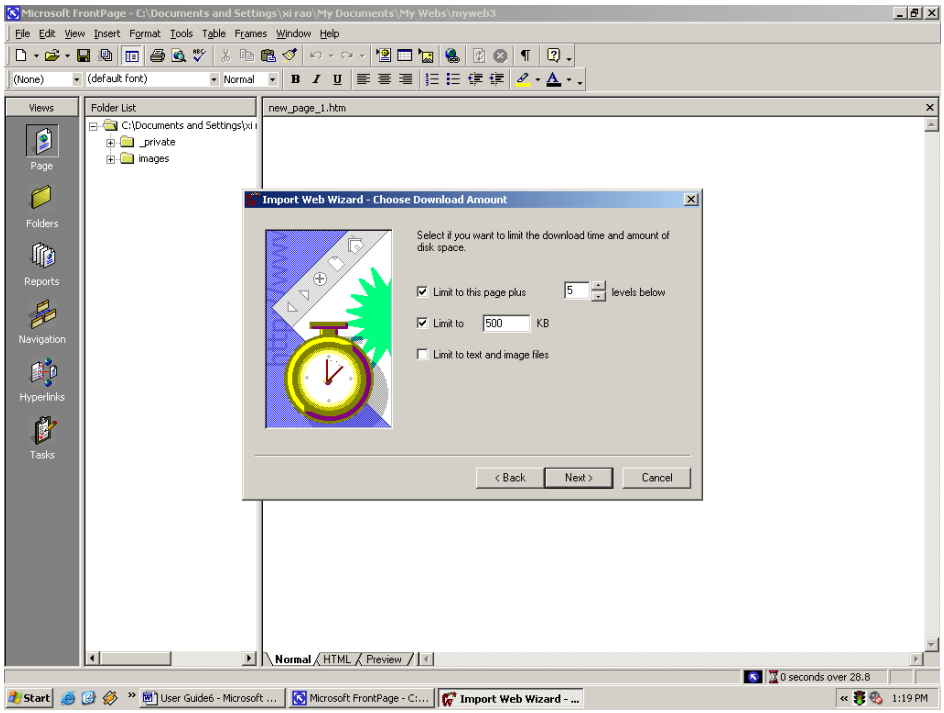
Section 4 – FrontPage download Yahoo web site



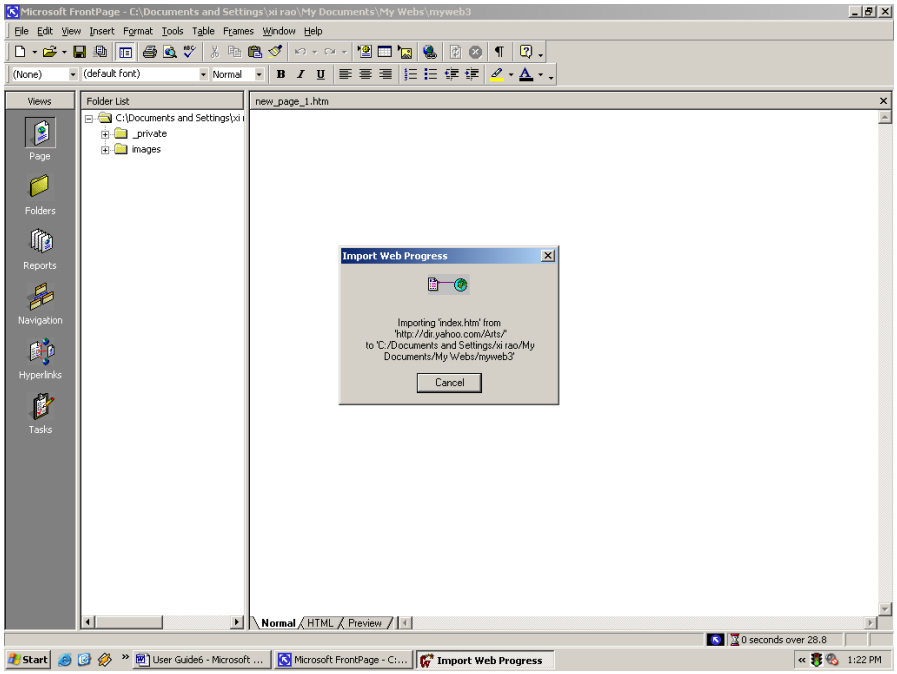

1. Start MS FrontPage2000 and choose File->new->web-> import web wizard.
2. Visit www.yahoo.com to choose one category (e.g., Art). The following page will show the URL in the address bar (http://dir.yahoo.com/Arts/).




3. Input the address for the Art category.

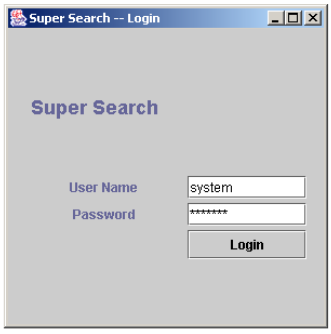



In this screen, you need to specify a limit for pages level and a limit for size. You just increase these to a very big number (as large as you want).

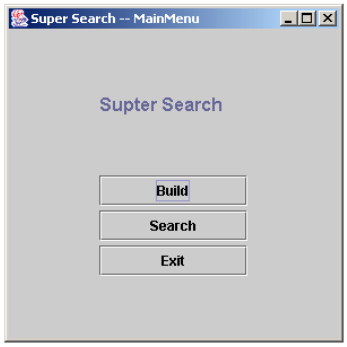



If you see this screen, it means that MS Frontpage is downloading the html file for you. If you want to skip this step, you can use the SuperSearch\yahoo directory as the root directory for building the database.

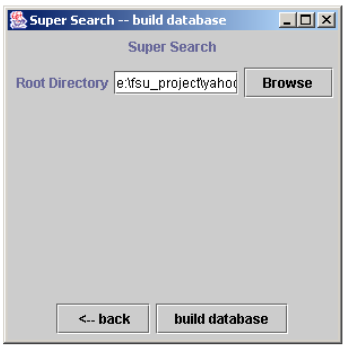
## Section 5 – Login



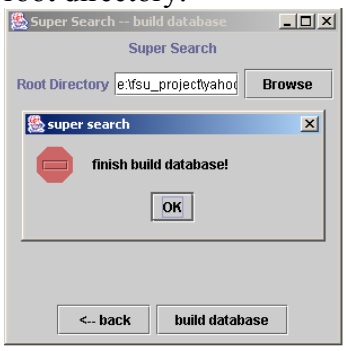

In this screen, you can input the user name and password. The Super System has a built in user name (system) and password (manager).




## Section 6 – Build

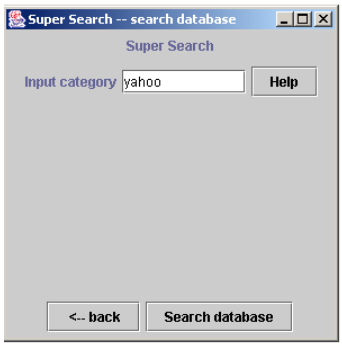



After you click the build database button, you will begin to build. This will take some time depending on your input, i.e., depending on how many html files are under your root directory.

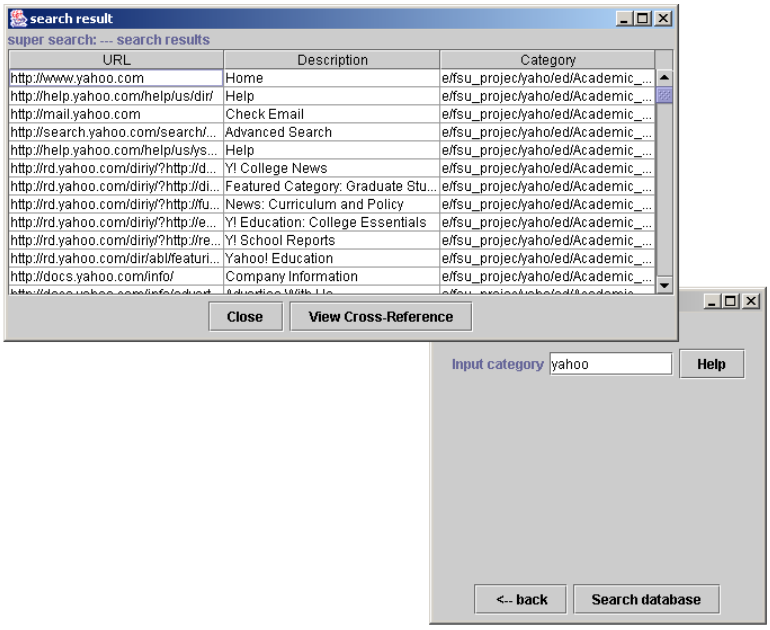



This screen shows a dialog that will notify you when the build is finished.

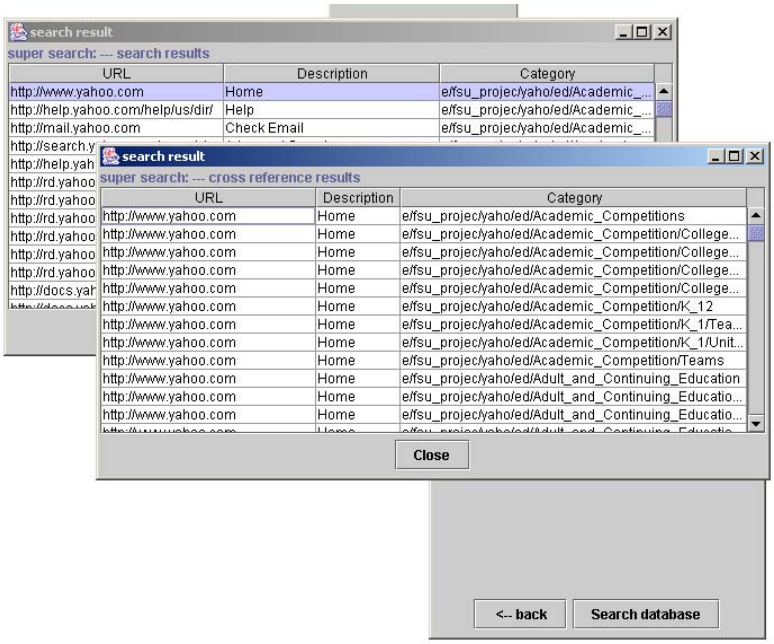
Section 7 – Search and view cross-reference




You can input the keyword here for a search.




From the above table, you can choose the URL to view the cross-references.




This table shows the cross-references.