Online Reference Library

By

Ashika Wijesekera

Submitted to
the Faculty of the Information Engineering Technology Program
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Engineering Technology

University of Cincinnati
College of Applied Science

March 2001

Online Reference Library

By

Ashika Wijesekera

Submitted to
the Faculty of the Information Engineering Technology Program
in Partial Fulfillment of the Requirements
for
the Degree of Bachelor of Science
in Information Engineering Technology

Author: Ashika Wijesekera	Date: 03-02-2001
Faculty Project Advisor: Prof. Russell McMahon	Date: 3/4/01
Department Head:	Date: 3/7/01

Acknowledgements

I am deeply grateful for the assistance I received from my advisor, Prof. Russell McMahon during my senior design project work at University of Cincinnati. I would also like to express my deep gratitude to Dr. Sam Geonetta for exchanging ideas and providing support and encouragement through out my senior design project and undergraduate studies. In addition, I would like to thank Prof. Robert Schlemmer for his expert guidance. I would also like to thank Dr. Ashraf Saad for introducing me to the Internet Technology.

This project would not have been possible without unselfish help rendered to me by my friends Priya Shah, Robert Brown, Chip Cellum, and Jeff Black. And I specially thank them all. I have been assisted by many ways by my fellow friends at the college of Applied Science, Pam Naber, Maralee Viox, Candise Wise, Caryle Clear, and Jen Fritz and I sincerely thank them all.

Also, I would like to give special thanks to my husband Nimal Wijesekera for his tolerance and assistance given to me through out my undergraduate studies.

Finally, I want to thank all the people associated with the University of Cincinnati who have provided me continued support during my undergraduate degree.

Dedication

"Our greatest glory is not in never falling, but in rising every time we fall."
~Albert Einstein

This project is dedicated to my parents and my husband, Nimal Wijesekera who was at my side all the way.

Table of Contents

Section	Page
Acknowledgements	i
Table of Contents	ii
List of Illustrations	iii
Abstract	iv
1. Statement of the Problem	1
1.1 Definition of the Need	1
2. Review of the Literature	3
3. User Profile	5
4. Objectives of the Project ("Deliverables")	5
5. Design and Development	
5.1 Budget	6
5.2 Time Line	6
6. Design Protocols and Proof of Design	8
7. Conclusions and Recommendations	40
8. Notes	42
9. References	44

List of Figures

Figure 1.	This shows the design protocol of the web site	8
Figure 2.	This shows the system architecture of the project	9
Figure 3.	This shows the Database Diagram web site	10
Figure 4:	This shows the site diagram of the Web site	13
Figure 5.	This shows the Home Page 1 (Default.htm)	14
Figure 6.	This shows the Sign Me Up Page (SignUp.htm)	16
Figure 7.	This shows the Sign Up Confirmation page (SignUpRecord.asp)	18
Figure 8.	This shows the Member Records (Record.asp)	20
Figure 9.	This shows the Library Catalog (Library Catalog.htm)	25
Figure 10.	This shows the Search Results (SearchResults.asp)	26
Figure 11.	This shows the Search Details (SearchDetails.asp)	29
Figure 12.	This shows the Checkout Confirmation (COConf.asp)	31
Figure 13.	This shows the Request Checkout Items (COItems.asp)	33
Figure 14.	This shows the e-mail the user received	36
Figure 15.	This shows the Help Page (helppage.asp)	39

Abstract

The purpose of this project was to develop an online database for a company having a large number of departments and employees who need references on a regular basis. An Online Reference Library is developed in such a way that employees will be able to search references and checkout these references by visiting the company's intranet site. The project incorporates a database containing the references of all the departments and an employee can browse and select those references to check out.

The basic system architecture of the project consists of three components. The Client (Browser), Web Server, and the Database Server.

Online Reference Library

1. Statement of the Problem

The technical staff in a local software corporation works together to produce world-renowned CAD/CAM software product. The staff includes mechanical engineering specialists, computer graphics experts, and computer scientists. They rely heavily on reference materials in problem solving. Specific references need to be available to different departments. The references are maintained at the department level and if the needed manual/reference is not within the individual department, a purchase request is submitted.

In order to locate a specific reference, an individual must check with each department. There is no centralized list of all the references maintained by the various departments. For example, if an individual needed to locate a manual pertaining to MS FrontPage, he/she must check with each department. Since there are many departments, this may be a difficult and time-consuming task. This often leads to the searcher to relinquish the search and purchase externally.

1.1 Definition of the Need

To develop an online database[1] so that any department can have access to other departments' references. Additionally, a checking-in/checking-out system would manage the resources, saving money and time by eliminating procurement of duplicate copies.

This project will allow employees to search references and checkout these references by visiting the company's intranet[2] site. The project incorporates a database containing references that an employee can browse and select references to check out.

Each employee will go through an authentication process involving a member ID and password. New members can get a unique login ID by self-registering. The member ID and password does not relate to the company's employee database. When an employee registered the information such as name, location, e-mail address, and chosen member ID and password, it is added to the patron table in the library database. Once logged in, it prompts the employee to the Member Record page where he/she can see the references that have been checked out by him/her. At this moment, the employee has the option to extend the references. Then the employee will be allowed to search references based on title, subject, and ISBN. The results of the search include the author information, reference location, and the availability of the reference. After the user has made the selection of references, he/she can proceed to complete the transaction by selecting the *Proceed to Check Out* button. The user will get a confirmation and also will receive an automatic e-mail with all the information such as *Request Number, ISBN Number, Title, Library name, Library e-mail*, and the *Library extension*. Requested references will be delivered to the member through the Company Internal Mailing System within forty-eight hours. If the user needs the reference urgently, he/she has the option to contact over the phone or e-mail the library for a quick delivery.

2. Literature Review

An intranet is a private network that is contained within an enterprise. The tasks of information sharing, information publishing, e-mail, and document management are the primary use of intranets. In the early days, these Web-based infrastructures provided employees with static information, but increasingly companies are choosing to use their intranets as the network over which they run mission-critical applications. Today, intranets help companies streamline and automate internal business procedures. (10)

A strong database back-end is required to store information needed for the user. A database is a collection of data that is organized so that its contents can easily be accessed, managed, and updated. The most prevalent type of database is the Relational Database Management System (RDBMS). A relational database is a set of tables containing data fitted into predefined categories. Each table (which is sometimes called a *relation*) contains one or more data categories in columns. Each row contains a unique instance of data for the categories defined by the columns. For example, a typical business order entry database would include a table that describes a customer with columns for name, address, phone number, and so forth. Another table would describe an order: product, customer, date, sales price, and so forth. A user of the database could obtain a *view* of the database that fitted the user's needs. For example, a branch office manager might like a view or report on all customers that had bought products after a certain date. A financial services manager in the same company could, from the same tables, obtain a report on accounts that needed to be paid. (13) There are several tools for data management. Some of the recognized tools are Microsoft SQL Server, Oracle, Sybase, and Informix.

Microsoft's Active Server Pages (ASP) can be created to access the data that is in the database. According to the Microsoft, "Active Server Pages is an open, compile-free application environment in which you can combine HTML, scripts, and reusable ActiveX server components to create dynamic and powerful Web-based business solutions. Active Server Pages enables server-side scripting for IIS with native support for both VBScript and Jscript"(6). You name the HTML file with the ".asp" file suffix. Microsoft recommends the use of the server-side ASP rather than a client-side script, where there is actually a choice, because the server-side script will result in an easily displayable HTML page. Client-side scripts (for example, with JavaScript) may not work as intended on older browsers.

ADO (ActiveX Data Objects) is an application program interface from Microsoft that lets programmers writing Windows applications get access to a relational and nonrelational database from both Microsoft and other database providers. For example, if you wanted to write a program that would provide users of your Web site with data from a SQL database or an Oracle database, you could include ADO program statements in an HTML file that you then identified as an Active Server Page. When a user requested the page from the Web site, the page sent back could include appropriate data from a database, obtained using ADO code. (1)

3. User Profile

The targeted users of this intranet reference library will be moderately experienced computer users and Web browsers. They will be comfortable with logically structured Web site that is easy to use. I created a Web site that is easy to use, has minimum complexity, and ease of navigation.

4. Objectives of the Project ("Deliverables")

The following were the completion objectives:

- An interactive intranet site for the user input
- An enhanced user interface for logical navigation and better look and feel
- Effective database design for data storage
- Employee login to verify the user
- The status of the member such as history of the check out references
- Extend *Due date* feature
- *Title/Subject/ISBN* wise reference search
- Availability of the references searched
- Automatic e-mailing system to confirm the request

5. Design and Development

5.1. Budget

5.1.1.	Hardware Requirements	
	Web Server with FrontPage Extensions / Database Server	$ 3,500
5.2.2.	Software Requirements	
	Visual Interdev 6.0	$ 900
	MS SQL Server 7	$ 900
	MS Office 2000	$ 125
	Windows NT Server and Workstation	$ 15
	Internet Explorer 4.o or above	Free with Windows NT
	Internet Information Server 4.0	Free with Windows NT

The software and hardware resources required for the completion of the project were available in the College of Applied Science's facilities.

5.2. Time Line

Senior Design 1 - Winter 2000

Researched

Started Database design

Started Interface design

Co-op - Spring 2000

Finalized the structure of database design

Built the skeleton of the project using Access 2000 and FrontPage 2000

Researched on the use of ASP and XML

Senior Design 2 – Autumn 2000

Reviewed interface design

Learned MS SQL Server 7

Learned MS Visual InterDev 6

Finalized the skeleton of the project.

Migrated the project to Visual Interdev 6 and MS SQL Server 7.

Demonstrated Working Quick Prototype

Senior Design 3 – Winter 2001

Completed coding

Completed Testing/Troubleshooting

Completed documentation

March 2

Submit final documentation

Submit final project

March 7

Demonstrate final project

Throughout the project:

Gathered information

Built bibliography

6. Design Protocols and Proof of Design

To ensure a consistent look and feel throughout the project, I designed a standard scheme on each page. Figure 1 illustrates this:

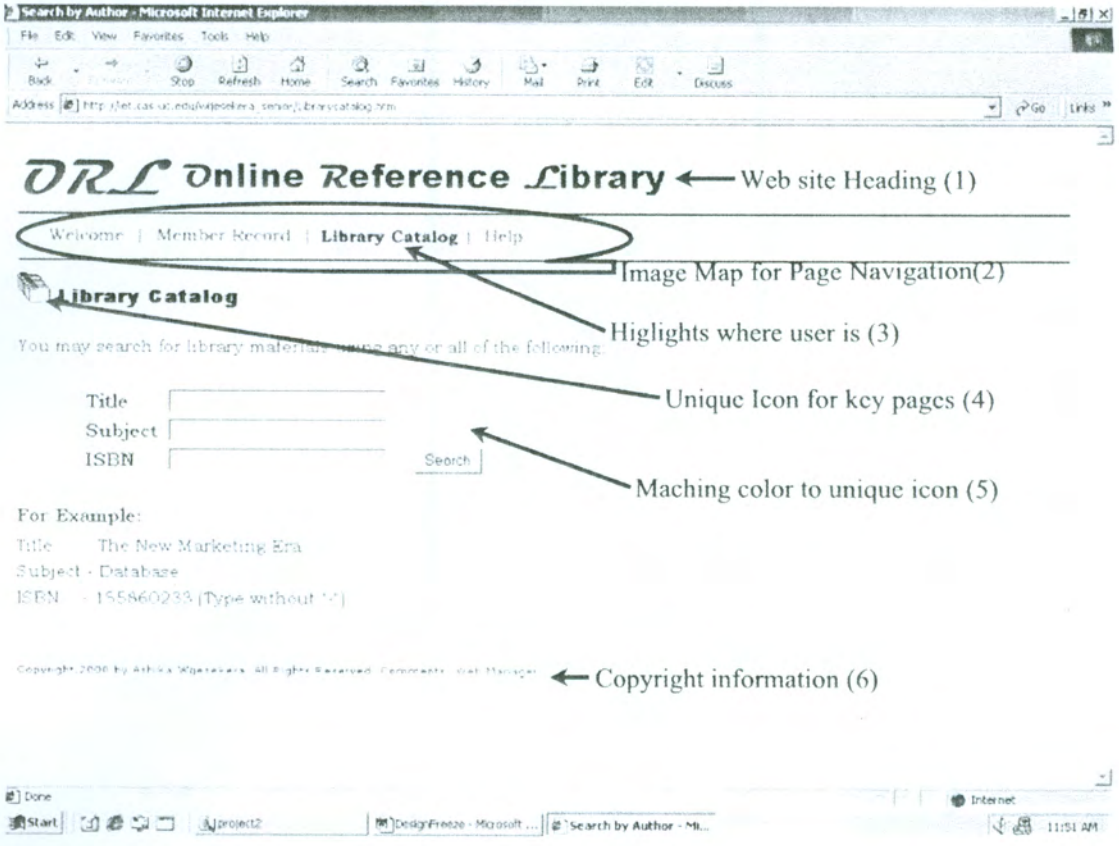

Figure 1. This shows the Design Protocols of the Web Site.

(1) An Identical Web site heading on all the pages for users to identify the Web site.

(2) Page navigation on all the pages except the home page and the sign me up page to navigate key pages (Library Catalog, Member Record, Help, and Home Page).

(3) Navigation bar highlights which page the user is in for the ease of navigation.

(4) Unique icon for key pages to describe the functionality of the page.

(5) Matching color to icon for the ease of user interaction.

(6) Copyright information on all the pages.

The basic system architecture of the project consists of three components. The Client (Browser), Web Server, and the Database Server. The architecture of the project is displayed below in figure2:

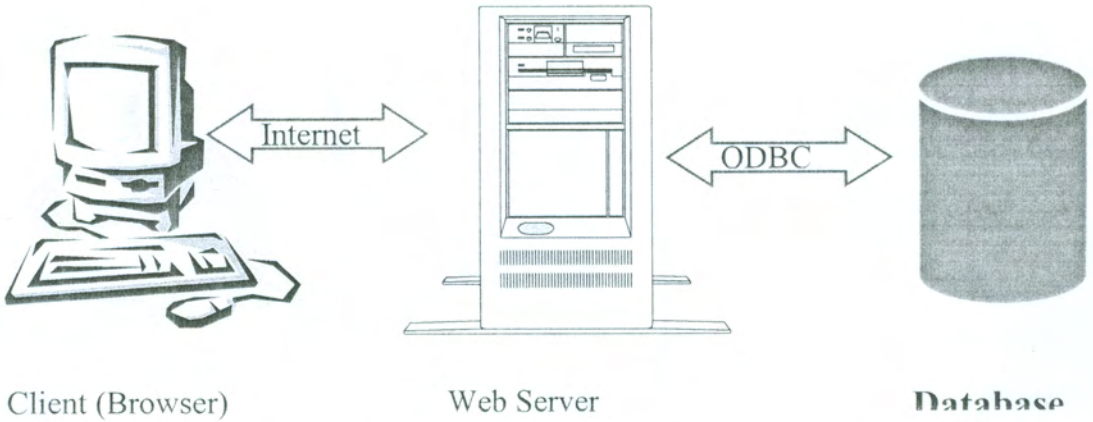

Figure 2. This shows the system architecture of the project

A Relational Database[3] called, "ashikawsd" is created in the Database Server. Because of its wide availability, SQL Server 7 was chosen as the database platform. The database is accessed using ActiveX Data Objects (ADO) [4] via ODBC[5], so the actual database platform is irrelevant to the application. There are nine tables contained in the database. Primary keys[6] and Foreign keys[7] have been setup in tables to enforce Referential Integrity[8]. The Database Diagram is displayed in the figure 3.

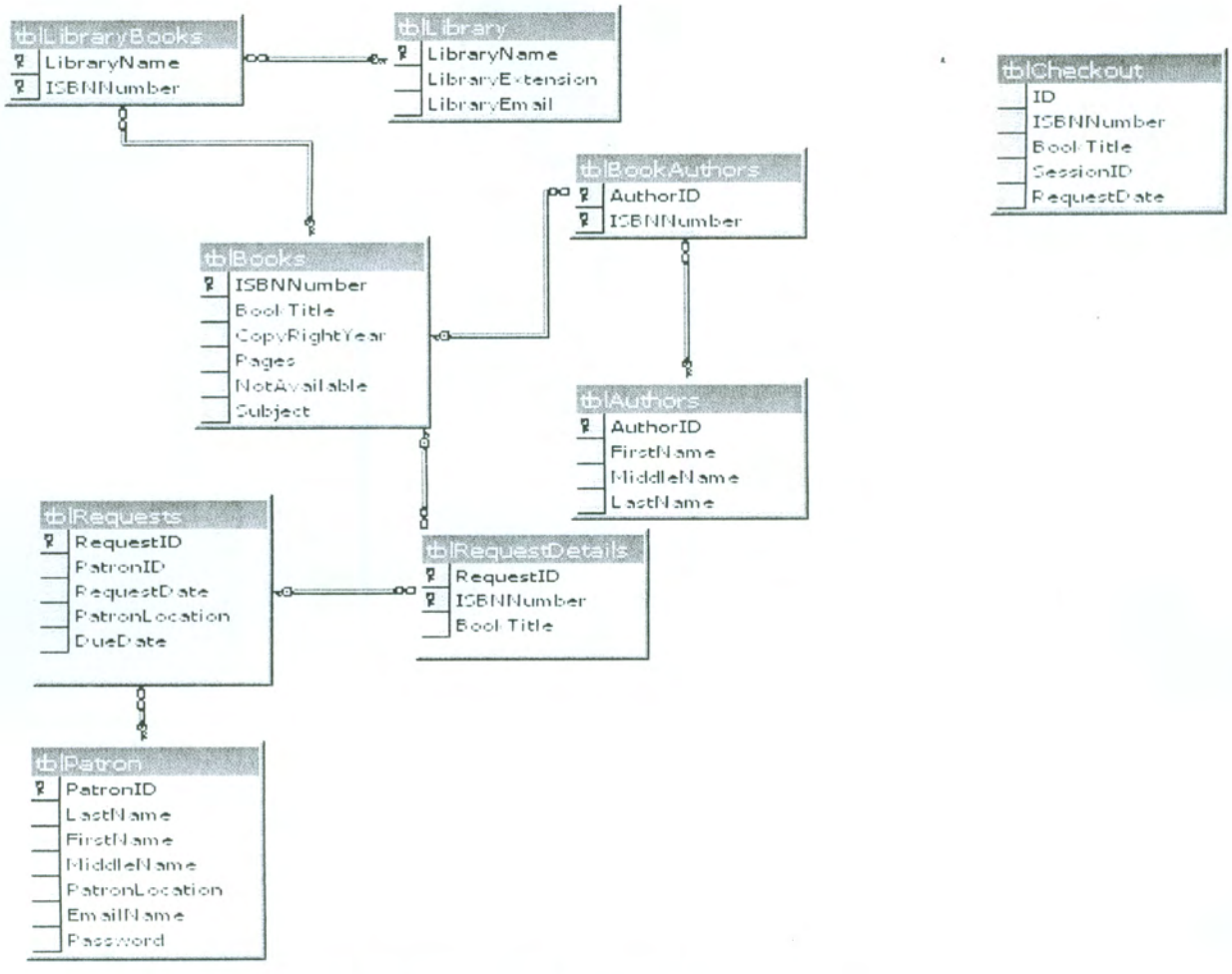

Figure 3: This shows the Database Diagram

The tables in the database contains:

- *tblBooks* table stores information concerning the references that checkout.
- *tblLibraryBooks* table contain a record for each book in each library.
- *tblLibrary* table stores all the information concerning the library.
- *tblBookAuthors* table contain a record for each book in each author.
- *tblAuthors* table stores information concerning the authors.
- *tblPatron* table stores information concerning the patron.
- *tblRequests* table stores information concerning the Requests.
- *tblRequestDetails* table contain a record for each request in each book.

- *tblCheckout* table temporarily stores information about the patron's requests during the checkout process. Once the request is made, the records in the tblCheckout are removed.

The code listed in listing 1 can be used to create the tables required for this project:

Listing 1:

```
CREATE TABLE [dbo].[tblAuthors] (
[AuthorID] [int] NOT NULL ,
[FirstName] [varchar] (50) NULL ,
[MiddleName] [varchar] (50) NULL ,
[LastName] [varchar] (50) NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[tblBookAuthors] (
[AuthorID] [int] NOT NULL ,
[ISBNNumber] [nvarchar] (13) NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[tblBooks] (
[ISBNNumber] [nvarchar] (13) NOT NULL ,
[BookTitle] [varchar] (50) NULL ,
[CopyRightYear] [smallint] NULL ,
[Pages] [int] NULL ,
[NotAvailable] [bit] NOT NULL ,
[Subject] [varchar] (50) NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[tblCheckOut] (
[ID] [int] IDENTITY (1, 1) NOT NULL ,
[ISBNNumber] [nvarchar] (13) NOT NULL ,
[BookTitle] [varchar] (50) NOT NULL ,
[SessionID] [varchar] (30) NULL ,
[RequestDate] [smalldatetime] NULL
) ON [PRIMARY]
GO
```

```
CREATE TABLE [dbo].[tblLibrary] (
[LibraryName] [varchar] (50) NOT NULL ,
[LibraryExtension] [nvarchar] (50) NULL ,
[LibraryEmail] [varchar] (50) NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[tblLibraryBooks] (
[LibraryName] [varchar] (50) NOT NULL ,
[ISBNNumber] [nvarchar] (13) NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[tblPatron] (
[PatronID] [varchar] (25) NOT NULL ,
[LastName] [varchar] (50) NULL ,
[FirstName] [varchar] (50) NULL ,
[MiddleName] [char] (10) NULL ,
[PatronLocation] [varchar] (50) NULL ,
[EmailName] [varchar] (50) NULL ,
[Password] [varchar] (50) NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[tblRequestDetails] (
[RequestID] [int] NOT NULL ,
[ISBNNumber] [nvarchar] (13) NOT NULL ,
[BookTitle] [varchar] (50) NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[tblRequests] (
[RequestID] [int] IDENTITY (1, 1) NOT NULL ,
[PatronID] [varchar] (25) NULL ,
[RequestDate] [smalldatetime] NULL ,
[PatronLocation] [varchar] (50) NULL ,
[DueDate] [smalldatetime] NULL
) ON [PRIMARY]
GO
```

A Web application called, "Wijesekera_senior" is created in the client side computer to create the Web project. A virtual directory is created in the Web server to include all the site's HTML and ASP pages. The virtual directory is named as

"Wijesekera_senior". The site is therefore is accessible by navigating to http://129.137.100.151/Wijesekera_senior. The Site Diagram is shown in Figure 4.

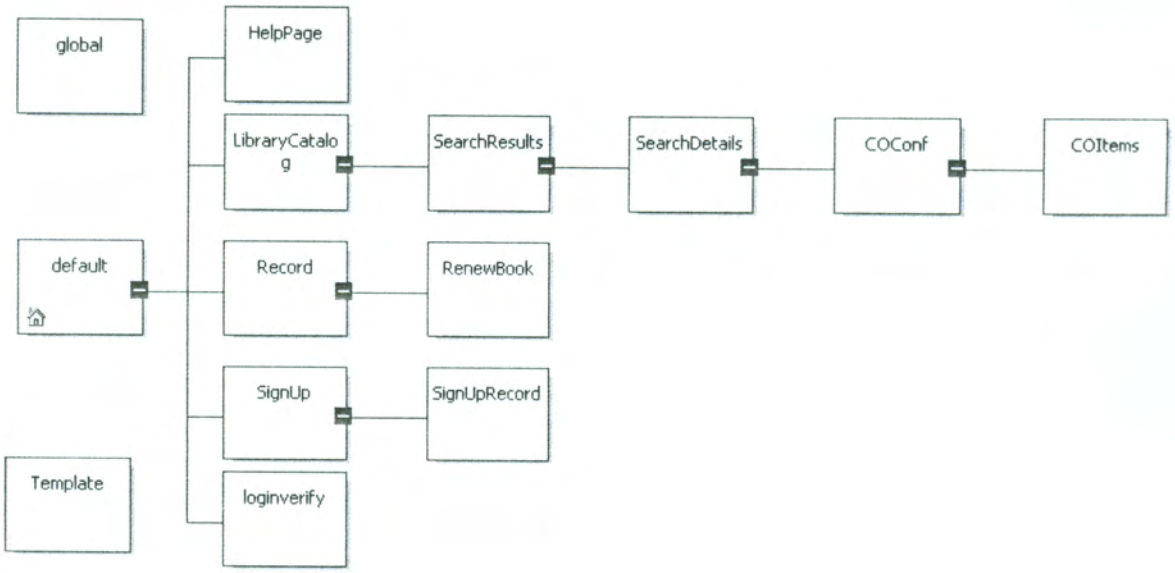

Figure4. This shows the site diagram of the Web site

I have created a *Global.asa* file, shown in listing 2 that sets up Session Variables[9] to be used throughout the site. This task is performed by placing code in the *Global.asa* file's Session_OnStart sub procedure; this code is executed whenever a user establishes a session with the server.

Listing 2:

```
Sub Session_OnStart

dim ProcedeISBN
dim checkout
'start with blanks for new user
session.Contents("User")=""
session.Contents("RequestedURL")=""
'This will hold the UserID that is logged in.
'Set this to an empty string because no one has logged on.
session("MemberID") = ""
session("PatronLoc") = ""
```

```
session("TotalBooks") = 0
session("Todaydate") = now()

End Sub
```

Following are the screens created for the functionality of the project:

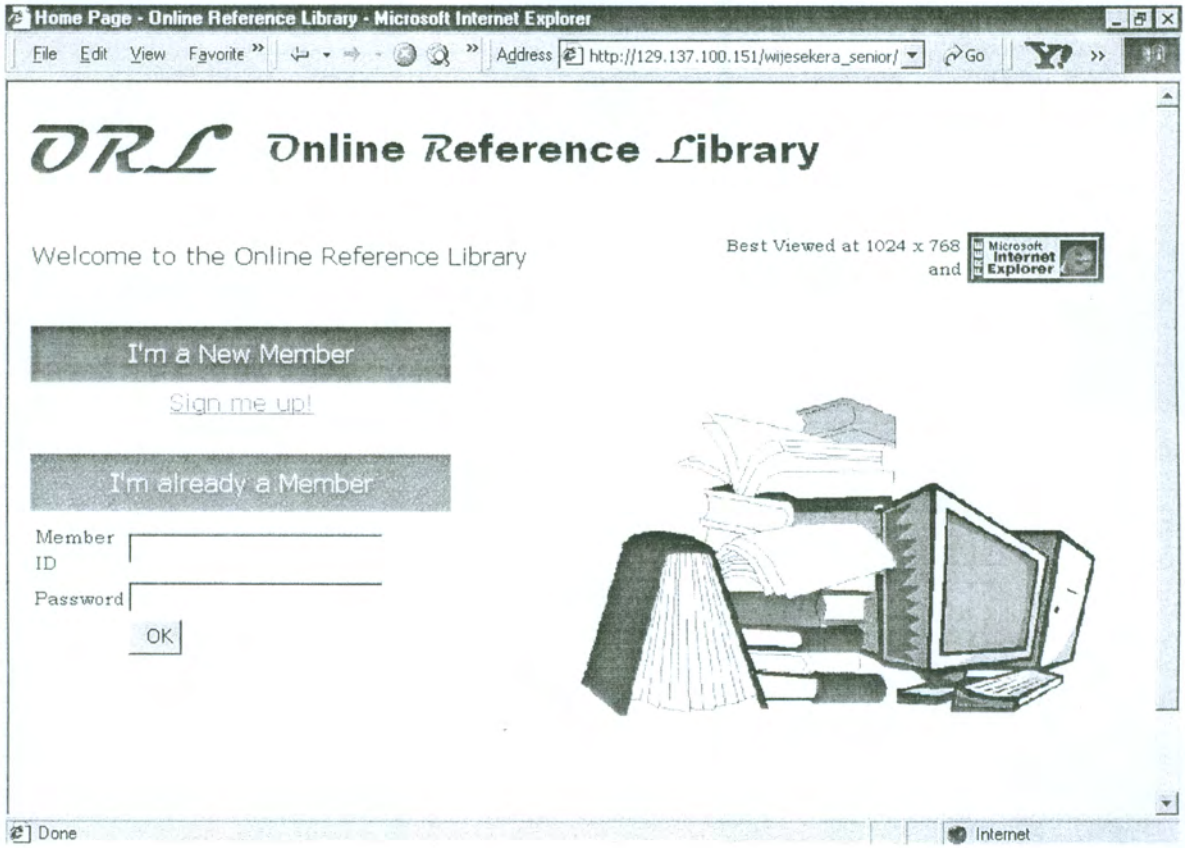

Figure 5. This shows the Home page (Default.htm)

This screen welcomes users to the Online Reference Library. There is a link to download Internet Explore 5.0, as this site is best-viewed using Internet Explorer 5.0 browser. Used Listing 3 code to accomplish this:

Listing 3:
```
<A href="http://www.microsoft.com/windows/ie/default.htm">
<IMG src="images/IE_amimation.gif"></A>
```

A Hyperlink[10] is used to navigate new users to the *SigUp.htm* page. Used listing 4 codes to accomplish this:

Listing 4:

```
<font size="4" face="Verdana">
<A target ="" href="SignUp.htm" >Sign me up!</A></font>
```

Also, there is a hyperlink for the user to send an email to the Web master. Codes in listing 5 is used to accomplish this:

Listing 5:

```
<A href="mailto:ashika_w@yahoo.com">Web Manager</A>
```

It lets current users log into the database. The screen displays two input text fields for *Member ID* and *Password* and also a *Submit* button to verify user member name and the password that is in the *tblPatron* table and direct the page to *Record.asp*. Listing 6 code is used to create the form with two input fields and a submit button.

Listing 6:

```
<form action="Record.asp" method="post" name="frmlogin">
<input name="txtID" size="23" id="txtID">
<input name="txtPWD" size="23" type="password" id="txtPWD">
<input name="cmdOK" type="submit" value="OK"></form>
```

After the *Submit* button is selected, the code in the click event of the command button is executed. This code checks if the entered login exists in the table of *tblPatron* in the Online Reference Library database. If it does not exist, a message with a hyperlink to the registration page appears. The user then gets into another screen to sign up. If the user exists in the table, it prompts to another screen with the check out records such as *ISBN Numbers, Titles, Request Dates*, and *Due Dates*

The web site has a special feature for security. If a user tried to log in to a page without logging in, the page is re-directed to the home page. For an example, if a user tried to open LibraryCatalog.asp without going to the default.asp for the user verification, the user gets re-directed to the home page. This feature accomplished by creating an ASP page called, loginverify.asp and included following code.

Listing 7:

```
<%
If Len(Session("MemberID")) <= 0 Then
  Response.Redirect "default.asp"
End If
%>
```

On all the pages except the homepage and the SignMeUp page, included following code:

```
<!-- #include file="loginverify.asp" //-->
```

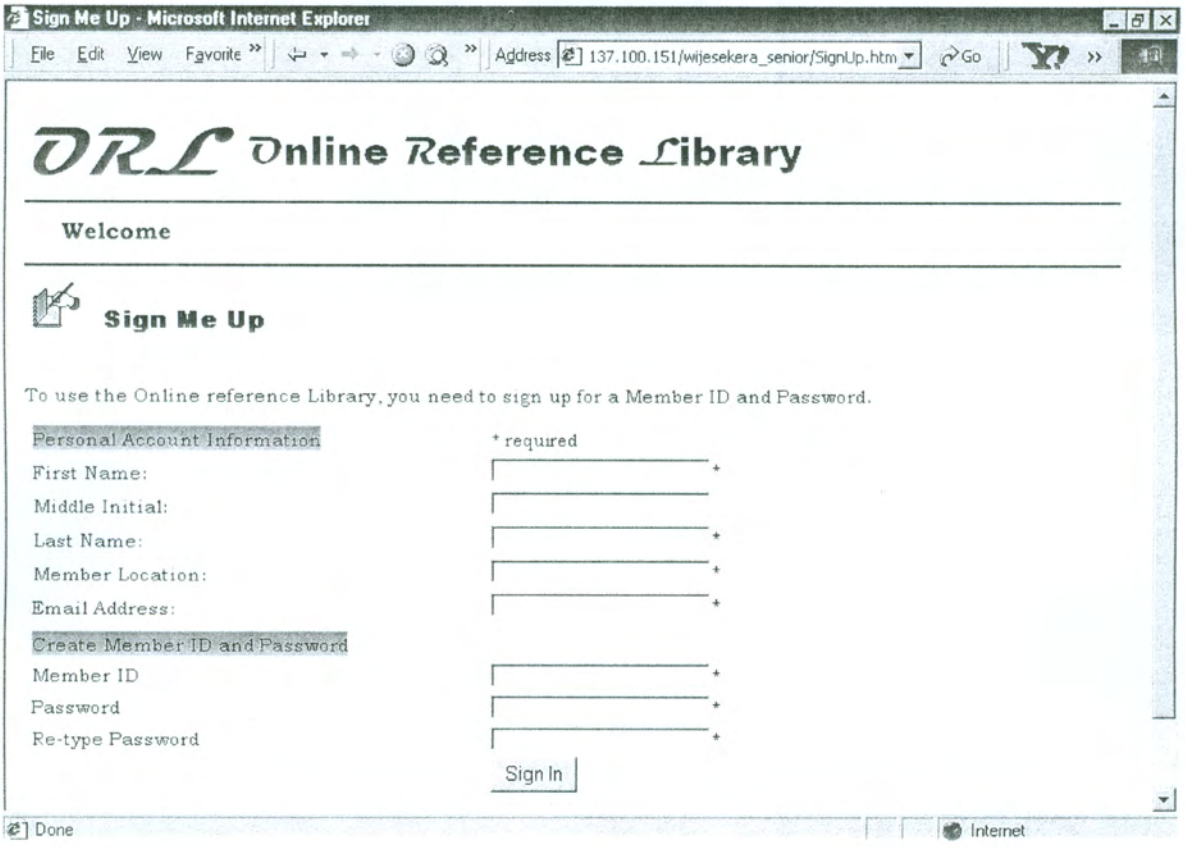

Figure 6. This shows the Sign Me Up page (SignUp.htm)

On this screen, there are input text fields for *Name, Member Location, E-mail Address, Member ID, Password, and Re-type Password.* The new user has to type all the information in order to register. Once the user clicks the *Sign In* command button, it prompts to the next screen, *SignUpRecord.asp.* Codes in Listing 8 is used to verify whether the user had typed all the required fields:

Listing 8:

```
function cmdSubmit_onclick() {
  var errorMsg = '';

  if (document.frmSubmit.txtFN.value == '')
   errorMsg = errorMsg + "You must supply a First Name.\n";

  if (document.frmSubmit.txtLN.value == '')
   errorMsg = errorMsg + "You must supply a Last Name.\n";

  if (document.frmSubmit.txtML.value == '')
   errorMsg = errorMsg + "You must supply a Member Location.\n";

  if (document.frmSubmit.txtEA.value == '')
   errorMsg = errorMsg + "You must supply an Email address.\n";

  if (document.frmSubmit.txtID.value == '')
   errorMsg = errorMsg + "You must supply a Member ID.\n";

  if (document.frmSubmit.txtPWD.value == '')
        errorMsg = errorMsg + "Your must supply a Password.\n";

      if (frmSubmit.txtPWD.value != frmSubmit.txtCFMPWD.value)
  {
   errorMsg = errorMsg + "Your Password and Password
      Confirmation do not match.\n";
   frmSubmit.txtCFMPWD.value = "";
   frmSubmit.txtCFMPWD.focus();
  }

  if (errorMsg != '') {
   alert(errorMsg);
   return false;
  }
  else {
          return true;
      }
}
```

The code that is similar to Listing 6 is used to create the input fields and the *Sign In* button. When this button is selected, it takes the user to the next page, *SignUpRecord.asp*.

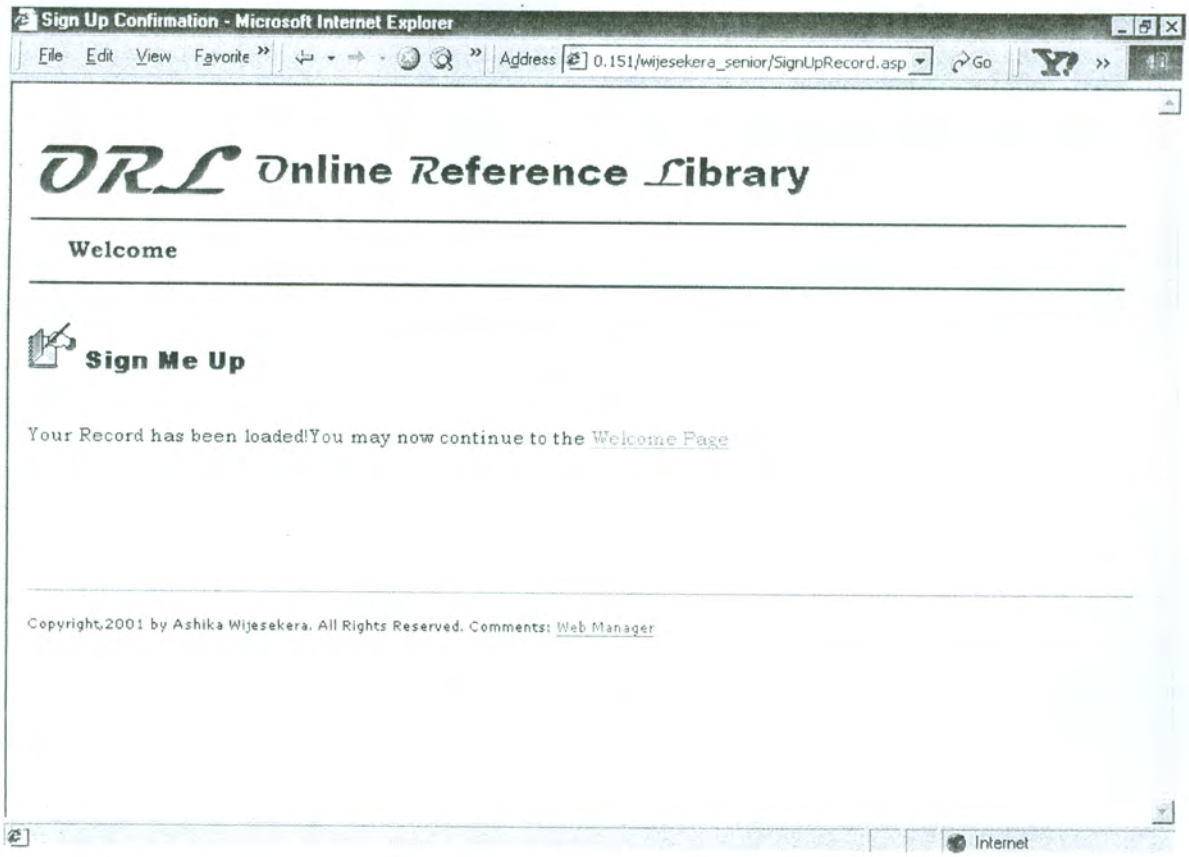

Figure 7. This shows the Sign Up Confirmation page - (SignUpRecord.asp)

On this screen, *tblPatron* is checked to see if the member name entered by the user exists in the table. If the member name is already exist, a message displayed to the user to select "Back" button on the browser to go back to the previous screen and re-enter another name. If the name is not existed, the records the user entered get executed and added into the *tblPatron* table. The user is then prompt to the Home page to login to the database. The ADO code shown in the listing 9 helped to accomplish this task:

Listing 9:

```
dim strsql, dim strconn , dim conn, dim strID, dim tmpStr,dim rs,
dim trVerifySQL
'open connection to db server next 4 lines never change
strConn = "DRIVER={SQL Server};Description=UserEntre; SERVER
=129.137.100.151;UID=ashika;PWD=ashika;DATABASE=ashikawsd"
set conn = server.CreateObject("adodb.connection")
set rs = server.CreateObject ("adodb.recordset")
conn.Open strConn
```

```
'Check to see if the Membername already exists in the table
strVerifySQL = "SELECT PatronID FROM tblPatron
where PatronID='" & Request.Form("txtID") & "'"
rs.Open strVerifySQL, conn, 3, 1
If rs.BOF and rs.EOF Then
'The recordset is empty. The membername does not exist,  they can use
the membername!!
'create sql string to be executed
strsql="Insert INTO tblPatron (PatronID, LastName, FirstName,
MiddleName, PatronLocation, EmailName, Password)
Values ('" & request("txtID") & "', '" & request("txtLN") & "', '" &
request("txtFN") & "', '" & request("txtMI") & "', '" & request("txtML") &
"', '" & request("txtEA") & "', '" & request("txtPWD") & "')"
Response.Write "Your Record has been loaded!"
'Execute the string to input into the table
conn.Execute strsql
Response.Write "You may now continue to the
<A HREF=""\wijesekera_senior\"">Welcome Page</A>"
Else
'There is a membername in the recordset, tell them to choose another...
Response.Write "The username you have chosen, "  &
Request.Form("txtID") &  " is already in use<br>" & vbCrLf
Response.Write "Please go back and choose another membername.
(Press the BACK button)"
End If
rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
```

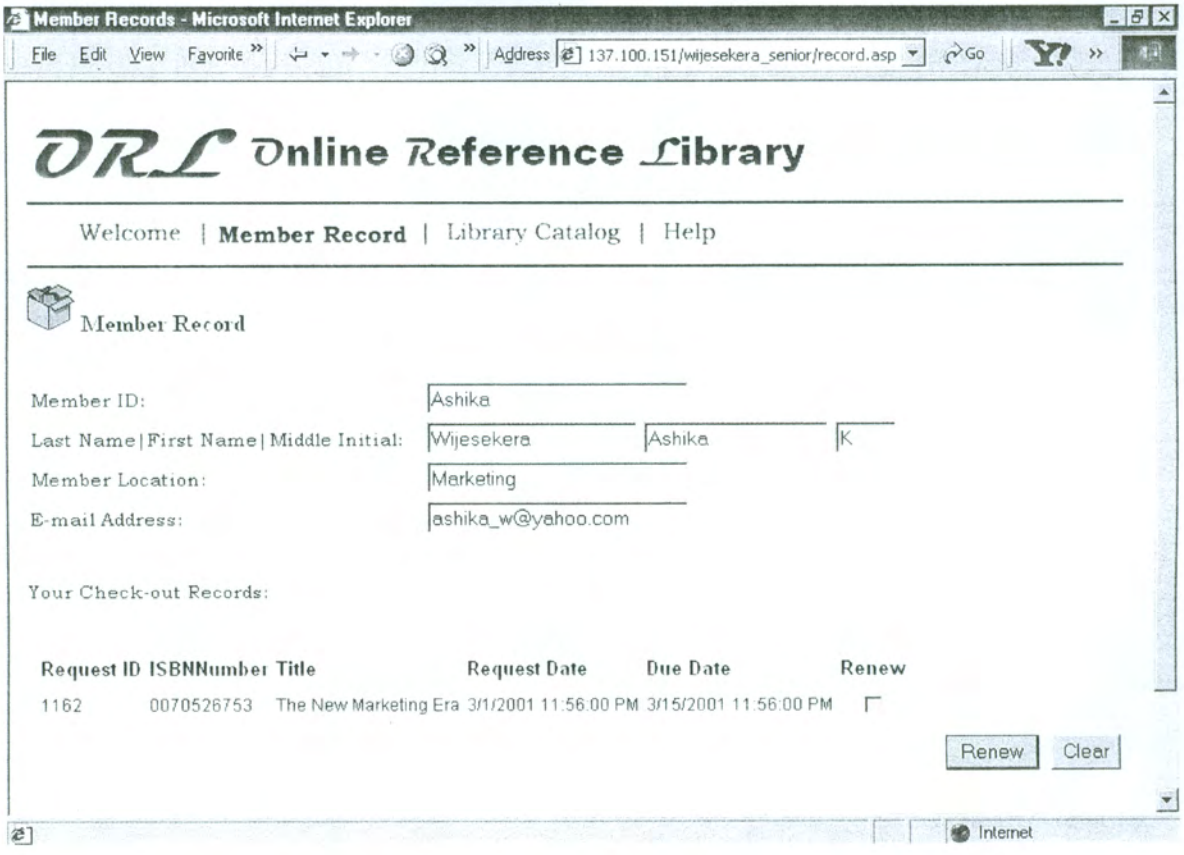

Figure 8. This shows the Member Records (Record.asp)

This page has an Image map[11] to navigate the main pages in the project. The following Listing a code is used to create the image map.

Listing 10:

```
<map name="NavBar5">
<area shape="RECT" alt="Home" coords="3,3,100,24"
        href="default.htm" title="Home">
<area shape="RECT" alt="Member Record" coords="120,3,281,24"
        href="record.asp" title="Member Record">
<area shape="RECT" alt="Library Catalog" coords="254,3,410,24"
        href="Librarycatalog.htm" title="Library Catalog">
<area shape="RECT" alt="Help" coords="420,2,490,24"
        href="helppage.htm" title="Help">
<area shape="RECT" coords=0,0,0,0 >
</map>
```

When the existing user types *Member ID, Password*, and *submit* button in the *default.asp*, the related records retrieved and displayed from his/her history. The history indicates user information such as *Member name, location, e-mail address*, and checkout information such as *ISBN Numbers, Titles, Request Dates*, and *Due Dates*. The user can extend the date by checking the *Renew* check box.

There are two Recordset Objects[13] called *Record* and *MemberRecord*, two Stored Procedures[14] called *MemberRecord* (see listing 11) and *History* (see listing 14), Grid Object[15] called *Grid1*, and six Textboxes.

The source for the Record is the *MemberRecord* stored procedure. This stored procedure required two Parameters[16]. Parameters include information that the user entered in the text boxes, txtID and txtPWD in the Home page (Figure 5). By using the code that is in the listing 12, parameters are passed to the stored procedure and the six text boxes are then filled.

Listing 11:

```
CREATE PROCEDURE MemberRecord
@vParam varchar(25), @vParam1 varchar(50)  AS
if @vParam1 IS NOT NULL
select tblPatron.PatronID, tblPatron.LastName, tblPatron.FirstName,
tblPatron.MiddleName, tblPatron.PatronLocation, tblPatron.EmailName
from tblPatron
where tblPatron.PatronID = @vParam and
tblPatron.Password = @vParam1
else
select tblPatron.PatronID, tblPatron.LastName, tblPatron.FirstName, t
blPatron.MiddleName, tblPatron.PatronLocation, tblPatron.EmailName
from tblPatron
where tblPatron.PatronID = @vParam
```

Listing 12:

```
'Passing txtID and txtPWD parameters to the stored procedure called
'"MemberRecord'. Record set is 'Record'.
Sub Record_onbeforeopen()
dim vParam
dim vParam1
```

```
vParam = Request.Form("txtID")
vParam1 = Request.Form("txtPWD")

If vParam <> "" Or vParam1 <> "" Then
  Record.setParameter 1, vParam
  Record.setParameter 2, vParam1
  session("MemberID") = vParam
else
 vParam = session("MemberID")
 Record.setParameter 1, vParam
 Record.setParameter 2, NULL
'response.write session("IDLogin")
 end if
End Sub
```

In order to show the checkout records, the *Grid1* grid object and the *Recordset1* recordset created. By using the code that is in the listing 12, parameters are passed to the *History* stored procedure (see listing 13).

Listing 13:

```
Sub Recordset1_onbeforeopen()
dim PatronID
Call ProcessRenewals 'Process any renewals
if session("MemberID") = "" then
        PatronID = Request.Form("txtID")
        Recordset1.setParameter 1, PatronID
else
        Recordset1.setParameter 1, session("MemberID")
end if
End Sub
```

Listing 14:

```
CREATE PROCEDURE History
@PatronID varchar(10) as
select r.RequestID, r.RequestDate,r.duedate,d.ISBNNumber,b.BookTitle
from tblRequests r, tblRequestDetails d, tblBooks b
where r.PatronID = @PatronID and r.RequestID = d.RequestID
and d.ISBNNumber = b.ISBNNumber
```

The code in the listing 14 is used to verify the member name and the password that is typed in the previous screen are matched with the *tblPatron* table. If the record exists, the user information such as his/her personal information and the status of the

checkout records are displayed. Otherwise, a message is prompted for the user to go back to the previous screen and re-enter.

Listing 15:

```
Dim conn
Dim rs

strConn = "DRIVER={SQL Server};Description=UserEntre; SERVER=
129.137.100.151;UID=ashika;PWD=ashika;DATABASE=ashikawsd"
set conn = server.CreateObject("adodb.connection")
set rs = server.CreateObject ("adodb.recordset")
conn.Open strConn

'Check to see if the member is a valid member...
strSQL = "SELECT PatronID, Password FROM tblPatron
WHERE PatronID='" & Trim(Request.Form("txtID")) & "'"
rs.Open strSQL, conn, 3, 1

If rs.BOF and rs.EOF then
'The member name does not exist in the database.
Session("ErrorMessage") = "Sorry, the membername that you
        have entered does not exist."
Response.Redirect "default.asp"
Else
'There is a matching member in the database.  Check to see if the
'password matches.
If rs("Password")=Request.Form("txtPWD") Then

'If the password matches the MemberID then put the MemberID in the
'session variable and welcome them to the site.
Response.Write "Welcome " & Session("MemberID") & " to the
Online Reference Library!"

session("PatronLoc")=txtPatronLocation.value
Else
'The password must not match, tell them and let them figure it out.
Session("ErrorMessage") = "The password for member " &
Request.Form("txtID") & " is incorrect." Response.Redirect "default.asp"
End If

End If
session("PatronLoc")=txtPatronLocation.value
rs.Close
conn.Close
Set rs = Nothing
Set conn = Nothing
```

Following code is used to re-new the references. When the user is selected *Renew* button, the Due Date will be extended additional 14 days.

Listing 16:

```
Sub ProcessRenewals()

Dim connIns
Dim cmd

If LCase(Request.ServerVariables("REQUEST_METHOD")) = "post"
Then
If LCase(Request.Form("action_type")) = "renew" Then
        Set connIns = Server.CreateObject("ADODB.Connection")
        Set cmd = server.CreateObject("ADODB.Command")
        sSQL = ""
        connIns.Open "DRIVER={SQL Server};Description=UserEntre;
        SERVER=129.137.100.151;UID=ashika;PWD=ashika;
        DATABASE=ashikawsd"

        For each item in Request.Form("rid")
                sSQL = "UPDATE tblRequests SET duedate =
                duedate + 14 WHERE requestID=" & item

                cmd.ActiveConnection = connIns
                cmd.CommandText = sSQL
                cmd.Execute

        Next
        connIns.Close
        Set cmd = Nothing
        Set connIns = Nothing
End If
End If

End Sub
```

RenewBook.asp page created for the extend date feature.Following code is used:

Listing 17:

```
<%
For Each Item In Request.Form
Response.Write Item.Name
Response.Flush

Next
%>
```

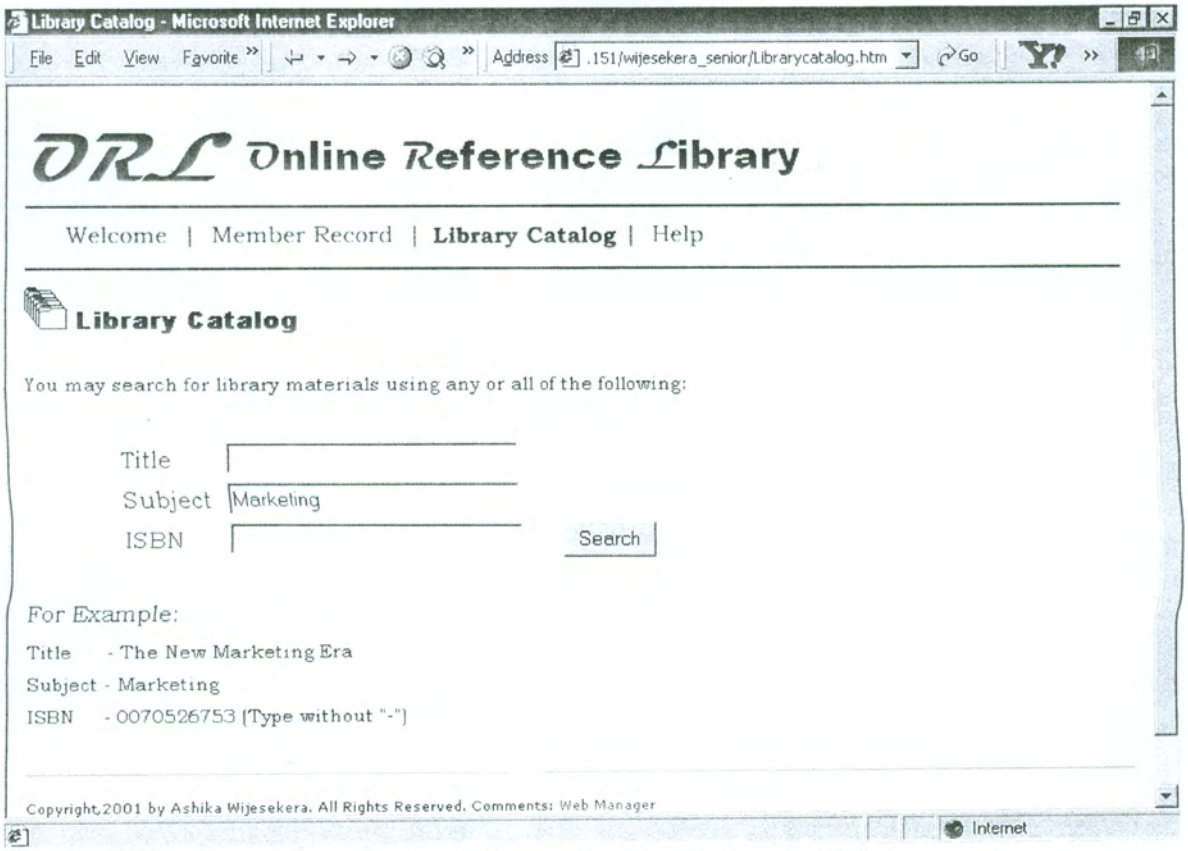

Figure 9. This shows the Library Catalog (Library Catalog.htm)

This is the search screen. The user is allowed to search references according to the *Title, Subject*, and *ISBN*. Text input fields for *Title, Subject*, *ISBN* and a *Search* command button are on this screen. After the user enters data in one or more text fields, he/she has to select the *Search* button to execute the query. When the *Search* button is selected, the user is taken into the next page, *SearchResults.asp*. The code to accomplish this task is similar to the coding that is in the listing 6.

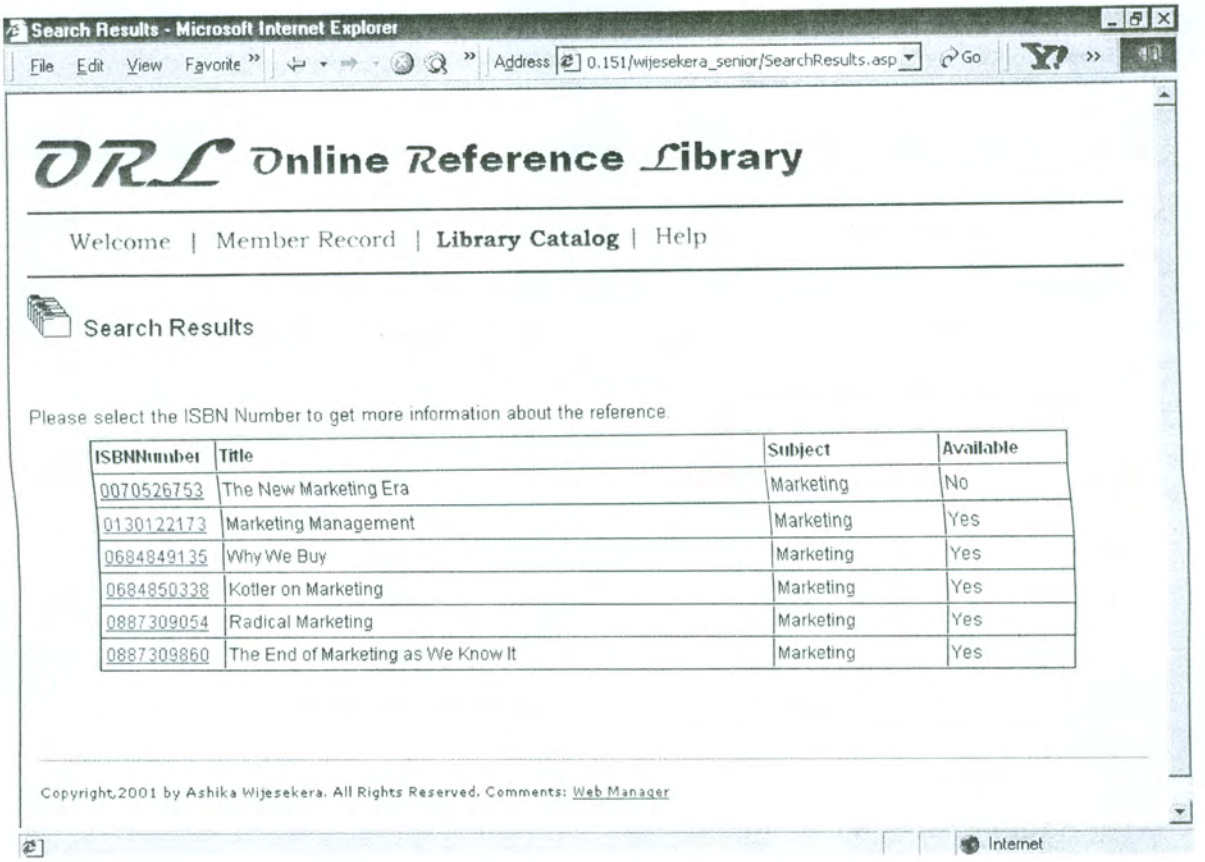

Figure 10. This shows the Search Results (SearchResults.asp)

Based on the data entered in the text fields in *LibraryCatalog.htm*, the *tblBooks* table is accessed and the related records are retrieved and displayed. There is a column next to each record indicating the reference' availability. Hyperlink is provided for each retrieved record. The user can click on a link and see the details about the particular reference. If the related records not in the *tblBooks* table, a message indicating, "No products matching that criteria were found" is displayed. The user has to select the Back button in the browser to do a new search.

There are two controls on this page. Recordset and the Grid Object. *Recordset1* is the Recordset object and the *Grid1* is the Grid object. *Recordset1* acts as the source for the *Grid1* to connect to the database's stored procedure called *SearchResults* (Listing 18). *Recordset1* uses the connection called *Connection 1* to talk to the database.

Listing 19:

```
CREATE PROCEDURE SearchResults
@p1 as varchar(50),
@p2 as varchar(20),
@p3 as varchar(13) as

DECLARE @s1 as varchar(100)
DECLARE @s2 as varchar(100)
DECLARE @s3 as varchar(100)

SET @s1 = @p1 + '%'
SET @s2 = @p2 + '%'
SET @s3 = @p3 + '%'

select  tblBooks.BookTitle, tblBooks.Subject, tblBooks.ISBNNumber,
CASE tblBooks.NotAvailable WHEN 1 THEN "Yes" ELSE "No" END
AS NotAvailable
from tblBooks
where  tblBooks.BookTitle like @s1
and tblBooks.Subject like @s2
and tblBooks.ISBNNumber like @s3
```

The values for the *@p1, @p2*, and *@p3* are retrieved from the previous page to the recordset before opening the recordset. The code shown in the Listing 20 is used to accomplish this.

Listing 20:

```
'Calling the stored procedure through the parameters
<script ID="serverEventHandlersVBS" LANGUAGE="vbscript"
RUNAT="Server">

Sub Recordset1_onbeforeopen()
dim vParam1
dim vParam2
dim vParam3
vParam1 = Request.Form("txtTitle")
vParam2 = Request.Form("txtSubject")
vParam3 = Request.Form("txtISBN")
Recordset1.setParameter 1, vParam1
Recordset1.setParameter 2, vParam2
Recordset1.setParameter 3, vParam3
End Sub

</script>
```

Following code that is in the Listing 21 is used to verify the searched results. If there is no records pertaining to the users searched criteria, it gives a message, "No products matching that criteria were found. Click BACK and search again." If not, "Please select the ISBN Number to get more information about the reference."

Listing 21:

```
'Searched results verification for the user
if Recordset1.eof then
   response.write "No products matching that criteria were found. Click
   BACK and search again."
else
   response.Write "Please select the ISBN Number to get more
   information about the reference."
end if
```

The Grid object has for columns – *ISBNNumber, Title, Subject*, and *Availability*. The *ISBNNumber* column's data are hyper linked to allow the user to see more information about the reference selected. Following code is used to accomplish this:

Listing 22:

```
="<ahref=SearchDetails.asp?ISBNNumber="+[ISBNNumber]+">

"+[ISBNNumber]+"</a>"
```

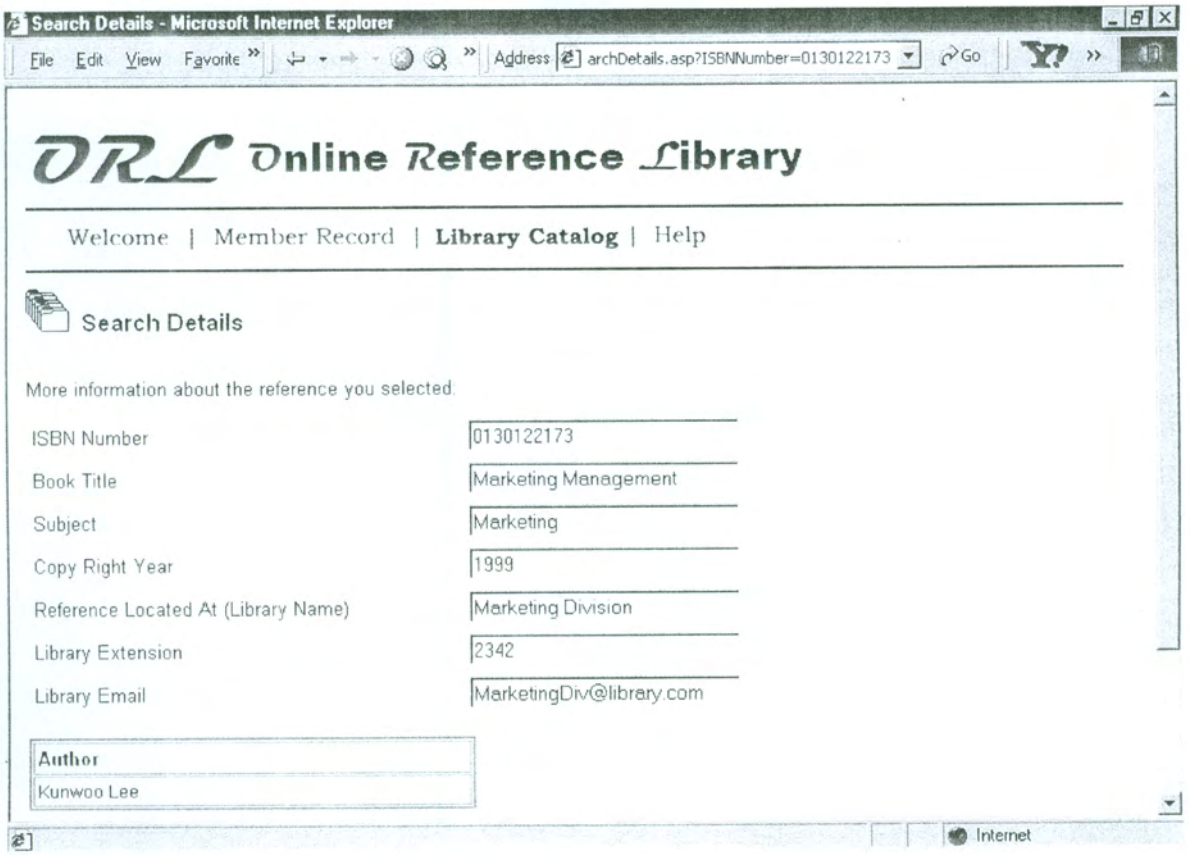

Figure 11. This shows the Search Details (SearchDetails.asp)

On this screen, more information about the reference is displayed. There are two buttons provided addition to the information:

(a) Select to Check out – After selecting the *Select to check out* button, the reference gets added to the *tblCheckout* table and the next screen is displayed.

(b) New Search – After selecting the *New Search* button, the user is taken to the *LibraryCatalog.htm* for a fresh search.

As shown in Figure11, there are six textboxes, a grid object, and two command buttons. *Recordset2* acts as a source for all the six text boxes and the *Recordset3* acts for the grid object as the source.

The source for the *Recordset2* is a stored procedure called, *SearchDetails* (see listing 23).

Listing 23:

```
CREATE PROCEDURE SearchDetails
@ISBNNumber nvarchar(13)  AS
select B.ISBNNumber, B.BookTitle, B.Subject, L.LibraryName,
L.LibraryExtension, L..LibraryEmail, L.LibraryName
from tblBooks B, tblLibrary L, tblLibraryBooks LB
where B.ISBNNumber=LB.ISBNNumber
and LB.LibraryName=L.LibraryName
and B.ISBNNumber=@ISBNNumber
```

The value of the *ISBNNumber* selected in the *SearchResults.asp* is passing to the stored procedure using the following code:

Listing 24:

```
Sub Recordset2_onbeforeopen()
dim vParam
vParam = Request.QueryString("ISBNNumber")
Recordset2.setParameter 1, vParam
End Sub
```

By writing following code, the value of the *ISBNNumber* is stored in a session variable called *ProcedeISBN* for future use:

Listing 25:

```
session("ProcedeISBN")=intISBNNumber
```

The source for the *Recordset3* is a stored procedure called, *SearchDetailsB* (see listing 26).

Listing 26:

```
CREATE PROCEDURE SearchDetailsB
@ISBNNumber nvarchar(13)  AS
select A.FirstName+" "+ isnull (A.MiddleName," ")+" "+
A.LastName as Name
from tblAuthors A, tblBookAuthors BA
where BA.ISBNNumber=@ISBNNumber
and A.AuthorID=BA.AuthorID
```

The value of the *ISBNNumber* selected in the *SearchResults.asp* is passing to the stored procedure using the following code:

Listing 27:

```
Sub Recordset3_onbeforeopen()
dim vParam
 vParam = Request.QueryString("ISBNNumber")
Recordset3.setParameter 1, vParam
End Sub
```

When the user selects the *Select to Check out* button, the code gets executed and the user takes into the next page, *COConf.asp*.

The user takes back into the *LibraryCatalog.htm* to start a fresh search if the user select *New Search* button. Following code do this:

Listing 28:
```
Sub btnewSearch_onclick()
window.navigate("Librarycatalog.htm")
End Sub
```

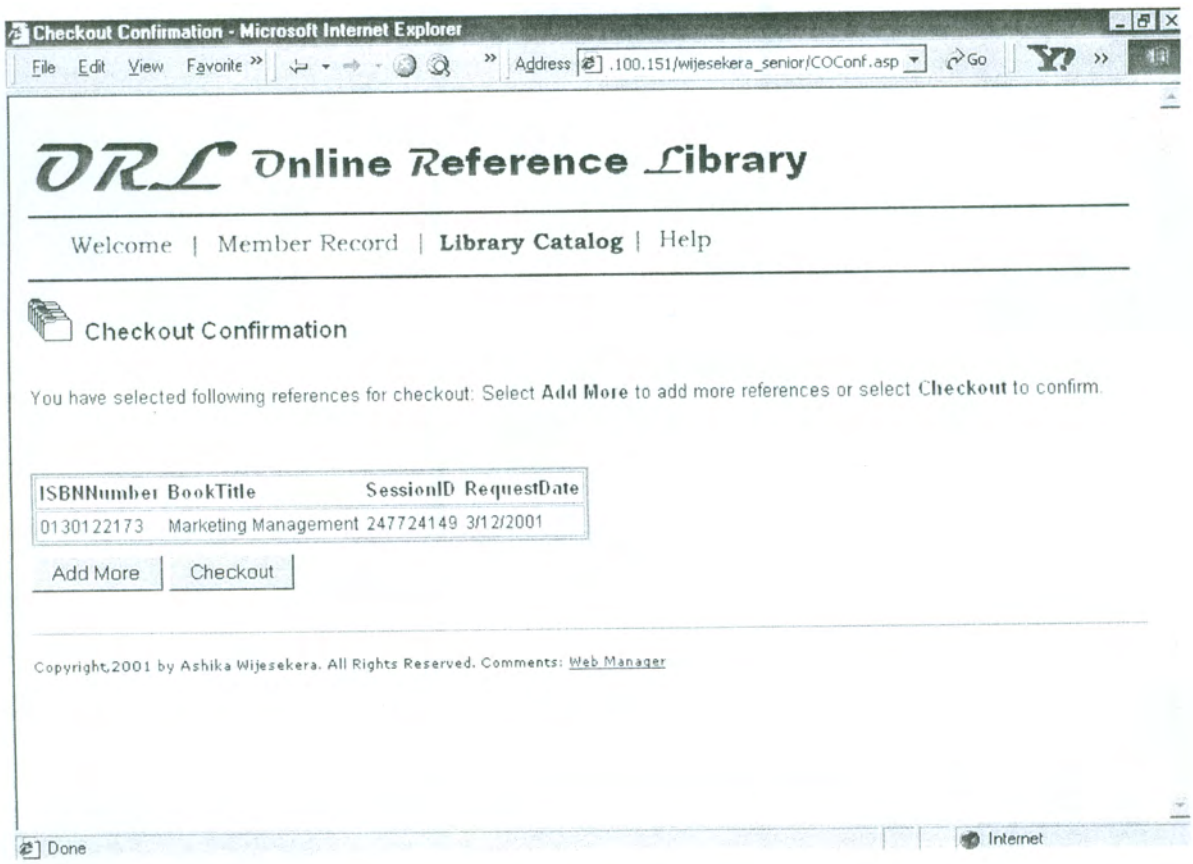

Figure 12. This shows the Checkout Confirmation (COConf.asp)

This screen shows all the references selected by the user to check out. The page includes recordset object, grid object, and two command buttons. *Recordset1* acts as the source for the *Grid1* object. *Grid1* has 4 columns - *ISBNNumber, BookTitle, SessionID, and the RequesDate.*

The two command buttons are used as:

(a) *Checkout* – After selecting this button, the records that were added to the *tblCheckout* get deleted and added to the *tblRequests* table. Then the user is taken into the next and the last screen.

(b) *Add More* - After selecting this button, the user is taken back to the *LibraryCatalog.htm* to search more references and add to the existing records.

After selecting the *Select to Checkout* button in the figure 11 and before opening figure 12, following code gets executed (Listing 29)

Listing 29:

```
Sub Recordset1_onbeforeopen()
dim strISBN, strBooktitle, strSessionID, dtDate, strSQL
strISBN = Request.Form("Textbox1")
strBooktitle = Request.Form("Textbox2")
strSessionID = session.SessionID
dtDate = FormatDateTime(Date(),0)
Dim conn
set conn = server.CreateObject("ADODB.CONNECTION")
set objCommand=server.CreateObject("ADODB.Command")
strConn = "DRIVER={SQL Server};Description=UserEntre;
SERVER=129.137.100.151;UID=ashika;PWD=ashika;
DATABASE=ashikawsd"
conn.Open strConn
strSQL = "INSERT INTO tblCheckOut(ISBNNumber, BookTitle,
SessionID, RequestDate) VALUES"
 strSQL = strSQL & "('" & strISBN &  "', '" & strBooktitle & "', '" &
 strSessionID & "', '" & dtDate & "')"
 objCommand.ActiveConnection = conn
 objCommand.CommandText = strSQL
 objCommand.Execute iRecs
 conn.Close
 set conn = Nothing
 set objCommand = nothing
 set dbconn = nothing
 recordset1.setParameter 1, session.SessionID

 End Sub
```

The source for the *Recordset1* is a stored procedure called, *checkoutitems* (see listing 30).

Listing 30:

```
CREATE PROCEDURE checkoutitems
@p1 varchar(30) as
SELECT ISBNNumber, BookTitle, SessionID, RequestDate
FROM tblCheckout
WHERE SessionID = @p1
```

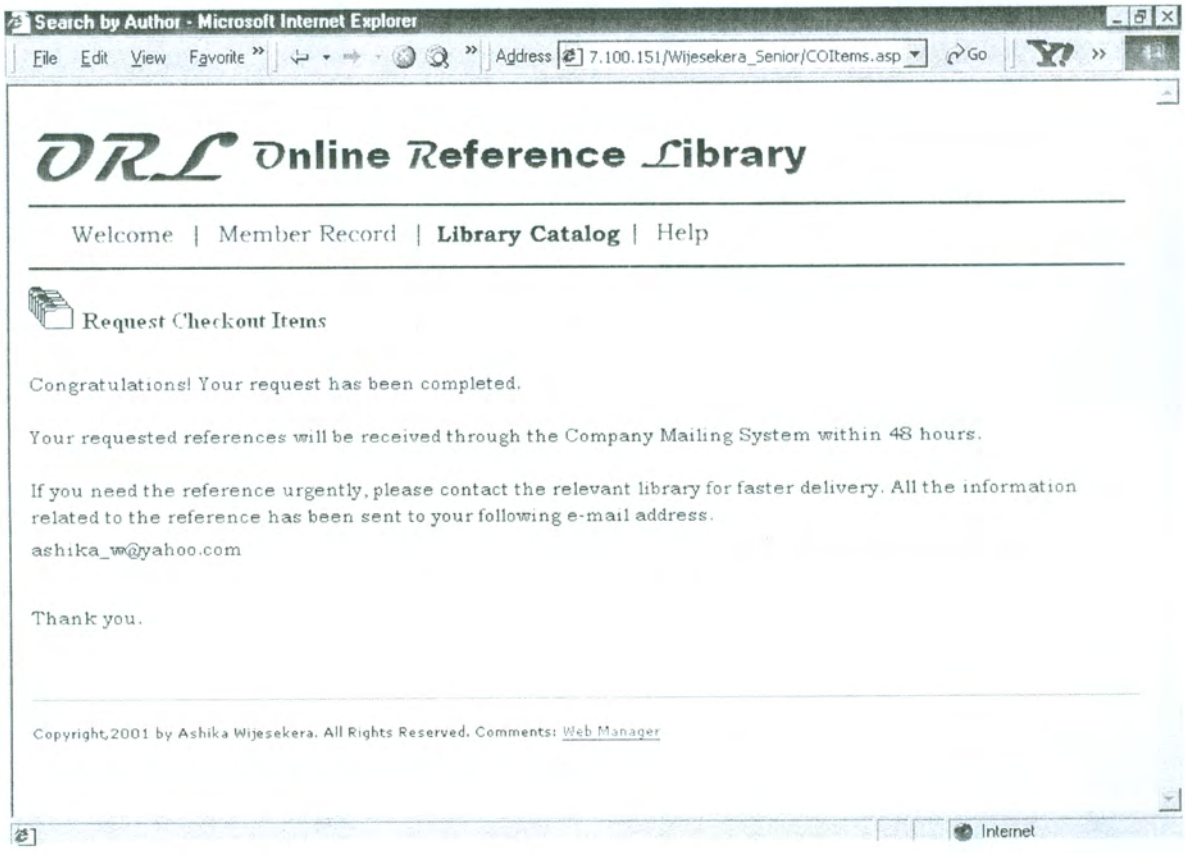

Figure 13. This shows the Request Checkout Items (COItems.asp)

On this screen, a message is provided to confirm the checkout process and sends an automatic e-mail to the user with all the information such as *Request Number, ISBN Number, Title, Library name, Library e-mail*, and the *Library extension*. Requested references are delivered to the member through the Company Internal Mailing System

within forty-eight hours. If the user needs the reference urgently, he/she has the option to contact over the phone or e-mail the library for a quick delivery.

The request has obtained by executing a stored procedure (Listing 32). This stored procedure requires four parameters to be passed. The code in listing 31 is used to call the stored procedure and pass the parameters. The 'insert' statement in the stored procedure inserts a new record into the *tblRequest* table to generate a new order. Then records are inserted into the *tblRequestDetails* table depending on the records existing in the tblCheckOut table for the specific session. After the relevant records are added to the *tblRequestDetails* table the records in the tblCheckOut table specific to the current session are deleted.

Listing 31:

```
Dim objConn, iRecs, objCommand, rs

'open connection to db server
'next 4 lines never change
strConn = "DRIVER={SQL
Server};Description=UserEntre;SERVER=129.137.100.151;UID=ashika;
PWD=ashika;DATABASE=ashikawsd"
set conn = server.CreateObject("adodb.connection")
set rs = server.CreateObject ("adodb.recordset")
set objCommand = server.createobject("ADODB.Command")
conn.Open strConn

objCommand.ActiveConnection = Conn.ConnectionString
objCommand.CommandType = 4
objCommand.CommandText = "Checkout"
objCommand.Parameters ("@PatronID")=session("memberID")
objCommand.Parameters ("@PatronLoc")=session("PatronLoc")
objCommand.Parameters ("@p1")=session.sessionID
objCommand.Parameters ("@dat")=session("Todaydate")

set rs = objCommand.Execute(iRecs)

dDueDate = rs("duedate")
lRequestID = rs("requestid")
```

```
rs.Close
set rs = nothing
if conn.State = 1 Then
        conn.Close
End If
```

Listing 32:

```
CREATE PROCEDURE checkout
(@PatronID as varchar(25), @PatronLoc varchar(50),
@p1 as varchar(30), @dat as smalldatetime)
                AS

begin transaction

declare @p2 int
declare @isbn nvarchar(13)

 insert into tblRequests
(PatronID, PatronLocation, RequestDate, DueDate)
values (@PatronId, @PatronLoc, @dat, dateadd(dd,14,@dat))

select @p2 = @@identity

insert into tblRequestDetails ( RequestID,ISBNNumber,BookTitle)
select @p2,tblcheckout.ISBNNumber, tblCheckout.BookTitle
from  tblCheckout
where tblCheckout.SessionID = @p1

--added next to lines to change the not available bit--
select @isbn = isbnnumber from tblCheckout where sessionid = @p1
update tblBooks set NotAvailable = 0 where ISBNNumber = @isbn

delete from tblcheckout where sessionID = @p1


commit transaction
select DueDate, requestid from tblRequests
where requestid = @p2
```

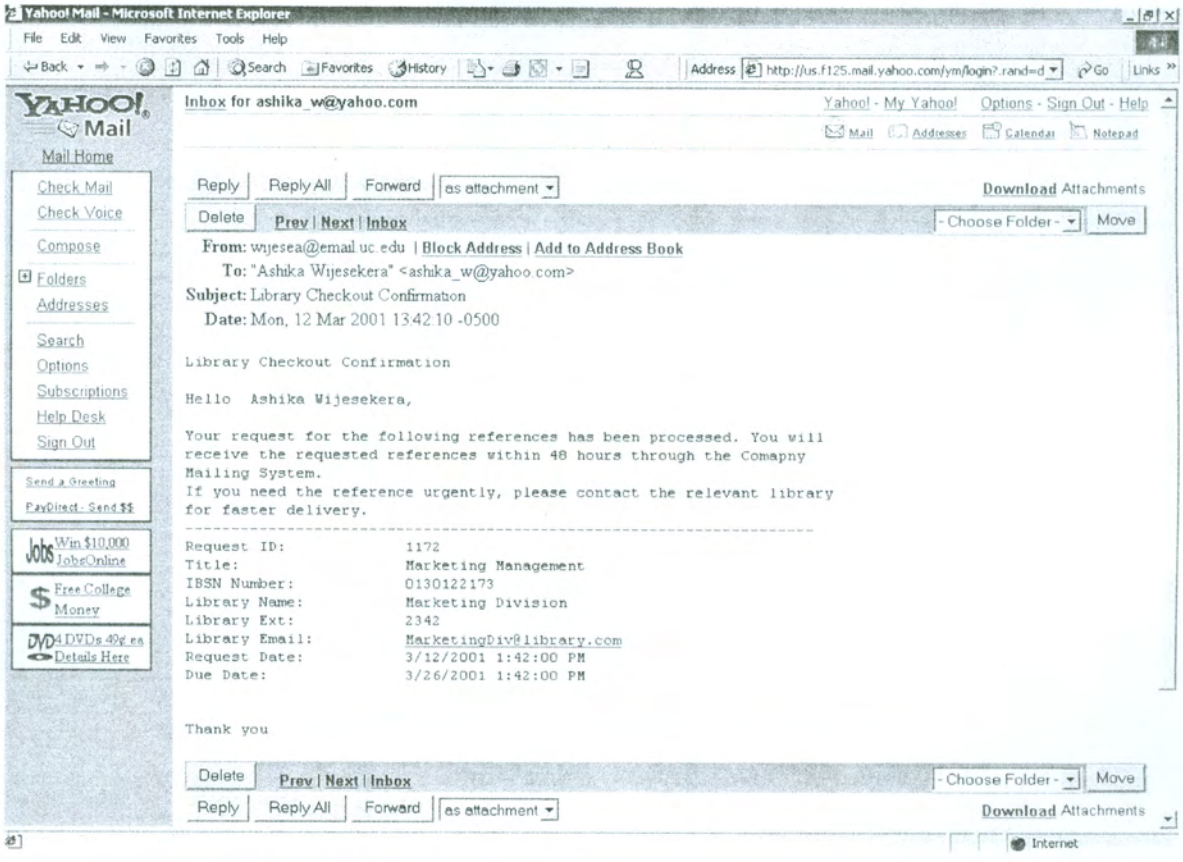

Figure 14. This shows the e-mail the user received

Following code is used to create the e-mail object and send to the user. Used free e-mail Component called, AspEmail 4.5.

Listing 33:

```
<%
Set oDBConn = Server.CreateObject("ADODB.Connection")

Set oInfoRS = Server.CreateObject("ADODB.Recordset")

oDBConn.Open strConn
sSQL = "SELECT * FROM view_emailRequestInfomation WHERE
requestid=" & lRequestID

oinfors.Open ssql, odbconn, 2, 2

Response.Flush


If Not oInfoRS.EOF Then

        'Now that we have the recordset lets get some information...
        sFirst = (oInfoRS("firstname") & "")
```

```
        sLast = (oInfoRS("lastname") & "")
        sPatronID = (oInfoRS("patronid") & "")
        sEmail = (oInfoRS("emailname") & "")
        sTitle = ""
        sISBN = ""
        sLocation = ""
        sLibName =""
        sLibExt = ""
        sLibEmail = ""
        sReqID = ""
        sReqDte = ""
        sDueDte = ""


        sBody = "Library Checkout Confirmation" & vbCrLf & vbCrLf
        sBody = sBody & "Hello  " & sFirst & " " & sLast & "," & vbCrLf
& vbCrLf
        sBody = sBody & "Your request for the following references has
been processed. You will receive the requested references within 48 hours
through the Comapny Mailing System." & vbCrLf
        sBody = sBody & "If you need the reference urgently, please
contact the relevant library for faster delivery." & vbCrLf
        sBody = sBody & "----------------------------------------------------
------------" & vbCrLf
        if oInfoRs.EOF then
        'There is not a recordset
        else
                Response.Write  oInfoRS("emailname")
                sTitle = oInfoRS("booktitle") & ""
                sISBN = oInfoRS("isbnnumber") & ""
                sLocation = oInfoRS("PatronLocation") & ""
                sLibName = oInfoRS("LibraryName") & ""
                sLibExt = CStr(oInfoRS("LibraryExtension")) & ""
                sLibEmail = oInfoRS("LibraryEmail") & ""
                sReqID = oInfoRS("RequestID") & ""
                sReqDte = oInfoRS("RequestDate") & ""
                sDueDte = oInfoRS("DueDate") & ""

                sBody = sBody & "Request ID:" & vbTab & vbTab &
lRequestID & vbCrLf
                sBody = sBody & "Title:       " & vbTab & vbTab & sTitle
& vbCrLf
                sBody = sBody & "IBSN Number:" & vbTab & vbTab &
sISBN & vbCrLf
                sBody = sBody & "Library Name:" & vbTab & vbTab &
sLibName & vbCrLf
```

```
                sBody = sBody & "Library Ext:" & vbTab & vbTab &
sLibExt & vbCrLf
                sBody = sBody & "Library Email:" & vbTab & vbTab &
sLibEmail & vbCrLf
                sBody = sBody & "Request Date:" & vbTab & vbTab &
sReqDte & vbCrLf
                sBody = sBody & "Due Date:" & vbTab & vbTab &
sDueDte & vbCrLf
                sBody = sBody & vbCrLf
                sBody = sBody & vbCrLf
                sBody = sBody & "Thank you"

  end if

        oInfoRS.Close

        Set oMail = server.CreateObject("Persits.MailSender")
        oMail.Host = "email.uc.edu" 'Server to send the message
        oMail.From = "wijesea@email.uc.edu"
        oMail.AddAddress sEmail, sFirst & " " & sLast

        oMail.Subject = "Library Checkout Confirmation"

        oMail.Body = sBody
        oMail.Send


        Set oMail = Nothing
        Else
        Response.Write "This request has alredy been processed."
End If
If oInfoRS.State = 1 Then
        oInfoRs.Close
End If

If Conn.State = 1 Then
        conn.Close
End If

Set oInfoRS = Nothing
Set conn = nothing

%>
```

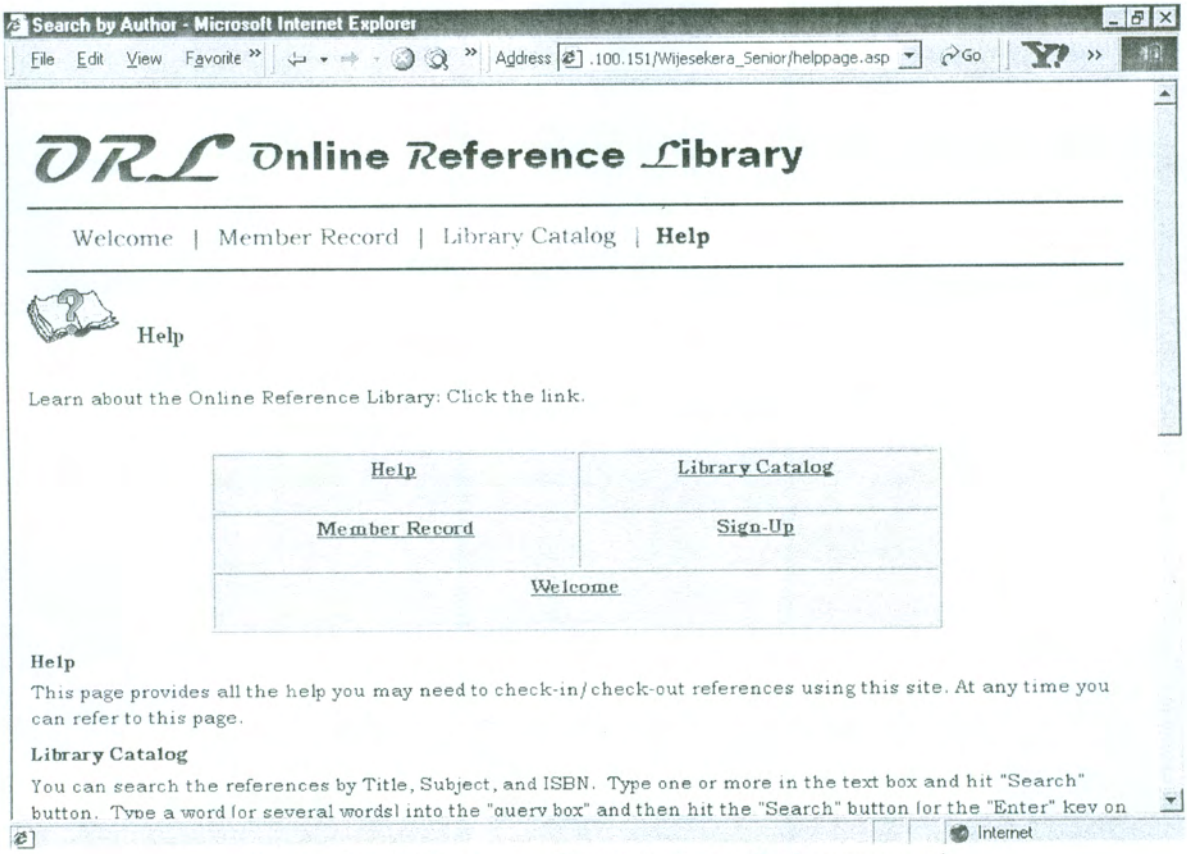

Figure 15. This shows the Help Page (helppage.asp)

This screen will provide information to the user about how to navigate the site, search for references, view member records, extend due dates, and etc.

7. Conclusion and Recommendations

The ORL meets the objective defined above in the "Objectives of the Project". The Web site is located at http:129.137.100.151/wijesekera_senior. The site is best viewed using Internet Explorer 4.0 or above as the web browser.

I used ASP code in harmony with HTML and used some site flow principles to make a site that is both functional and easy for the user to navigate. Microsoft Visual Interdev software, HTML, JavaScript, and VBScript are used to create the Web functionality. I used Microsoft Visual Interdev as the application development tool since it offers an integrated Web development system for creating dynamic, database-driven Web sites.

This site is also integrated with SQL Server and ADO code. I used SQL Server 7 as a backend because it is a widely used relational database management system in the industry today. SQL Server is used to create the database to store all the data and ADO code is used to dynamically access the data back and forth from the database and the Web server.

As it stands, the ORL site works as a fully functional web site. You could take the same code and with a few changes in the text, could create another library catalog for a different company. However, any Web site can be improved, and this section contains some suggestions for how the site can be extended.

My problem with the ORL site is that it does not allow each department to update their references using the same site. Presently, the department has to log into the database and add their references to the *tblBooks* table. To allow departments to update

their references, another ASP page with username and password system must be added to the site.

Another problem with the site, as it's currently implemented, is that it let's the users to extend the due date as many times as they want. This needs to be re-coded where they have limited number of times to extend the reference.

Another suggestion to make is that when users search references, it should show where the reference located when it is not available. Then the user can request the reference from the borrower.

I also would like to explain that Microsoft Visual Interdev is not the best tool for developing a Web site. For example, I used Visual InterDev's Design Time Controls to access the data. These DTC controls write the underlying ADO code for the developer. These controls are not flexible because the developer is not allowed to change or alter the underlying ADO code. Complex requirements cannot be fulfilled using the DTC controls. Writing all the ADO code by yourself gives you more control over the results. Therefore, I recommend using ADO code for the data access screens instead of using Recordset objects.

Moreover, the scripts needs to be included more error checking and handling when there are more users involved.

The other major concern is the Copyright. Many of the icons (pictures) are taken from the previously published and copyrighted materials. No permission for the materials exists.

Notes

1. **Database** - A database is a collection of data that is organized so that its contents can easily be accessed, managed, and updated.

2. **Intranet** - An intranet is a private network that is contained within an enterprise.

3. **Relational Database** - A relational database is a collection of data items organized as a set of formally-described tables from which data can be accessed or reassembled in many different ways without having to reorganize the database tables.

4. **ADO** - ActiveX Data Objects is an application program interface from Microsoft that lets programmers writing Windows applications get access to a relational and nonrelational database from both Microsoft and other database providers.

5. **ODBC** - Open Database Connectivity is a standard or open application-programming interface (API) for accessing a database. By using ODBC statements in a program, you can access files in a number of different databases, including Access, dBase, DB2, Excel, and Text. The main proponent and supplier of ODBC programming support is Microsoft.

6. **Primary Keys** – It is a constraint that ensures all rows of a table are unique by ensuring that one or more columns doesn't permit duplicate values to be entered. And also it disallows NULL for the column(s) defined in the constraint.

7. **Foreign Keys** – This associates one or more columns of a table with an identical set of columns that have been defined as a primary key constraint in another table. When the column values are updated in the table in which the primary key is defined, the columns defined in another table as a foreign key are automatically updated.

8. **Referential Integrity** – This prevents users or applications from entering inconsistent data. There are various rules that you can apply when you set referential integrity.

9. **Session Variables** – It is something that starts the moment a user requests a page from your Web site and ends soon after the user leaves. Each visitor to your Web site is given an individual session.

10. **Hyperlink** - On the Web or other hypertext systems, hyperlink is an element that links to another place in the same document or to an entirely different document. Query

11. **Image Map** – This provide another form of hyperlinking – one that permits assigning different URLs to different parts of an image. Another name for image map is Hotspots.

12. **Scripting Object Model** – This object model enables you to write programs that use properties and methods that automate aspects of the document during runtime.

13. **Recordset Objects** – Groups of records are called recordsets. A record is comprised of data in individual columns. Information you store or retrieve from a database is in the form of records.

14. **Stored Procedures** - An operation that is stored in the database server. Typically, stored procedures are written in SQL. They're especially important for client-server database systems because storing the procedure on the server side means that it is available to all clients. And when the procedure is modified, all clients automatically get the newer version

15. **Grid Object** – This displays the records collected and delivered by the recordset object. You must add this control to the document and link it to the recordset so that grid knows what data to display in the HTML table in the client.

16. **Parameters** – It is an item of information - such as a name, a number, or a selected option - that is passed to a program by a user or another program. Parameters affect the operation of the program receiving them.

References

(1) "ActiveX Data Objects". www.whatis.com. August 17, 2000

(2) Amundsen, Michael. *Using Visual Interdev 6*
Indiana:QUE, 1999

(3) Brockwood, Ted. "Getting Started With SQL".
Http://www.webdevelopersjournal.com/articles/sql.html. March 1999.

(4) Brown, Robert. Classmate, College of Applied Science, University of Cincinnati.
Personal Interview. January 27, 2001

(5) Buyens, Jim. *Running Microsoft FrontPage 2000*
Washington: Microsoft Press, 1999

(6) Cooke, Kevin. "Introduction to Active Server Pages".
Http://hotwired.lycos.com/webmonkey/98/39/index2a.html. September 30, 1998.

(7) Dice, Richard. "Choosing the Right Database Sysytem".
Http://hotwired.lycos.com/webmonkey/98/24/index3a_page3.html?tw=backend.
February 15,2000

(8) Fleming, Candace C. and Barbara Vonhalle. *Handbook of Relational Database Design.* New York: Addison-Wesley Pub Co, 1998.

(9) Greenspan, Jay. "Your First Database".
Http://hotwired.lycos.com/webmonkey/99/13/index0a.html February 15,2000

(10) Henry, Amanda Mitchell. "Intranet Management Made Easier".
Http://intranetjournal.earthweb.com/dlink.resource-jhtml.72.1274. October 12, 1999

(11) Morris, Charlie. "Developing Databases For the Web and Intranets".
Http://www.webdevelopersjournal.com/books/developing_databases.html.
October 27, 1999

(12) Mullich, Joe. "Data-driven intranets quick, but without glitz".
Http://www.intranetjournal.com/deployment/intranet_database_080699.html.
August 6, 1999.

(13) "Relational Database". www.whatis.com. December 19,1999

(14) Shah, Priya. Classmate, College of Applied Science, University of Cincinnati.
Personal Interview. February 10, 2001

(15) Walther, Stephen. *Active Server Pages 2.0.*
United States of America: Sams Publishing, 1999

(16) Wyncoop, Stephen. *Using Microsoft SQL Server.*
Indiana: QUE, 1999