Overview
For this project, you will implement full-text search capabilities for data stored in a MySQL database. You will also use jQuery+AJAX to implement a "search suggestion" interface and a mouse-over feature to show document details.

What to do
For this project:

- A data set of 198 records from the OpenVideo web site (http://www.open-video.org) will be posted in the Resources section of the Sakai site for you to use for this assignment (for any other use, you need to get an okay from the Open Video project). The data set will use the schema shown below in a table called p3records:

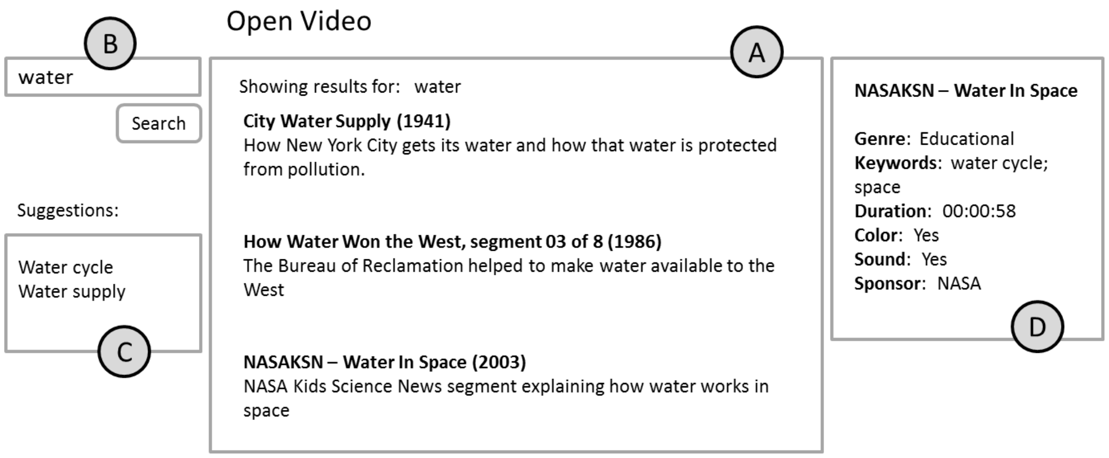
videoid	int primary key
title	varchar(255)
description	text
keywords	text
creationyear	int
sound	varchar(10)
color	varchar(10)
duration	time
durationsec	int
sponsorname	varchar(200)
contribname	varchar(50)
language	varchar(20)
genre	varchar(50)
keyframeurl	varchar(250)

- Below are two methods you can choose from to create the p3records table and to load the data for this assignment. Regardless of the method you use, make sure that you have successfully loaded all 198 records into your p3records table.

  - Option #1: A file called p3records.sql will be posted to the Resources section on Sakai. This file is a database "dump" file that you can use to re-create both the tables and data. To use it, copy the file to your home directory on ruby, then start the mysql client from the same directory, then enter "use webYOURNUMBER", and then "source p3records.sql". This will create the p3records table AND will load the data into it. Check it by doing "select * from p3records". Note that the p3records table created using this method does NOT have any fulltext index. This means that you will need to alter the table to add the fulltext index described later in the assignment.

  - Option #2: If you use this method, you will need to create the p3records table yourself using the names and data types shown above, plus the fulltext index described later. After creating the table, you can use the bulk loader to load the data from a text file called "p3-ovrecords.txt" that will be posted to Sakai. This file will contain the records one per line, fields separated by vertical bar characters '|'. The first line of the file will contain the field names and should be removed.

- Create a PHP page called results.php that will allow users to enter search terms and will return a list of matching results. results.php should produce a page that looks similar to the wireframe below:



Showing results for: water

City Water Supply (1941)
How New York City gets its water and how that water is protected from pollution.

How Water Won the West, segment 03 of 8 (1986)
The Bureau of Reclamation helped to make water available to the West

NASAKSN – Water In Space (2003)
NASA Kids Science News segment explaining how water works in space

Water cycle
Water supply

NASAKSN – Water In Space

Genre: Educational
Keywords: water cycle; space
Duration: 00:00:58
Color: Yes
Sound: Yes
Sponsor: NASA

- A user can enter search terms into the search box (B) and press the "Search" button to display a list of matching results in area (A). Use a MATCH..AGAINST to do the search with a FULLTEXT index on title, description, and keywords fields. When a user clicks the "Search" button (B), reload the entire page by passing the user's search terms as a GET parameter to results.php (e.g., results.php?search=water). For this assignment, you can display the results in the order in which they are returned by the MATCH..AGAINST. You can also display the results in one long list. You do NOT need to implement a paging mechanism.

  The results should be formatted as shown in area (A) above: title in bold followed by the year in parenthesis, then the description on a line below. Some descriptions may be long, so you should display only the first 200 characters of the description field from the database.

  Normally, the title and image of each result would be hyperlinked to the OpenVideo web page for that record. However, this year, the OpenVideo site is temporarily down for repairs, so we can't link to records on it.

  In the results area (A), display the current search term at the top of the results (e.g., "Showing results for: water"). If no search has been conducted, area (A) should be blank.

- As the user types into the search box (B), a set of search suggestions should be displayed in area (C). Details about how to do this are explained below. As a user types characters into the text box for the keyword search field (B), the current text in the search box should be sent via jQuery Ajax to the server. The server should return a list of matching phrases that will be displayed in a separate div element in area (C). This matching will be based on a list of 265 keyword phrases that will be provided in a file on Sakai in a file named: p3-keywordphrases.txt.

  After each keypress in the keyword search box, your jQuery Ajax code should be activated. It should send the current string that is in the keyword search box (i.e. the characters typed so far) to a script on the server called keyword-suggestions.php. This script should be written to compare the string sent by the Ajax call to the provided list of keyword phrases and return a list of up to 10 phrases that have the same prefix characters as the ones typed. For example, if the two characters "ma" were typed, the script would return the following keyword phrases, one per line:

  ```
  magic
  management
  manifest destiny
  manufacturing
  mapping
  marriage
  ```

These suggestions should be shown in a separate div element in area (C) of the diagram above. The suggestions list should be updated for each keypress so that (C) only shows suggestions for the characters currently typed into the keyword search field.

There are several ways you could choose to do the matching of characters typed to suggestions. Whichever method you use, make sure that you do the matching on the server-side in your keyword-suggestions.php script. One idea would be to create a MySQL table that would store a row for each of the 265 phrases and then to use a "LIKE" query to find partial matches. Another idea would be to store the 265 phrases in a PHP array and then check the input string against the phrases in the array (note: this later method only uses PHP, not MySQL).

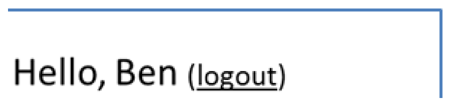
- When the user moves their mouse over a result displayed in the results area (A), details about that result should be shown in area (D). These details should be retrieved from the server using jQuery Ajax as described below.

  When the user moves their mouse over a result, implement a jQuery event handler that will call a PHP script on the server called result-details.php. jQuery should pass the videoid of the result that the mouse is positioned over to result-details.php using jQuery POST. The result-details.php script should use the videoid that is passed via POST to look up the record details in the MySQL database and then return HTML with the formatted record details as shown in area (D) above.

  When the user moves their mouse off of a result, area (D) should not display any details. In other words, if the mouse is not over a result, area (D) should be blank.

Advanced Functionality
If you implement the program as outlined above, you can earn up to a maximum of 90 out of 100 points. You can earn up to 10 additional points for implementing the advanced feature described below.

- Add a user account feature so that users must login before they are able to see the results.php page. If a user loads results.php and is not logged in, the page should NOT display the video search interface, but instead should display a login form that allows the user to enter a username and password. For this advanced functionality part of the assignment, you need to use session management as we discussed in class (i.e., session_start). You will need to have a database table that stores usernames and passwords as illustrated in the course slides. You can add the usernames and passwords to the database manually. If a user enters the correct password, then you should set a session variable so that the system will "remember" that they are logged in and continue to allow them to view and interact with the results.php page. You must include an account with the username "rob" with the password "rob" so that I will be able to log in to test the user account functionality.
  Also include a "logout" link on the results.php page that can be used to log the user out. If the user logs out and reloads the results.php page, it should not show them the video search interface, but instead should show them a login form. You may wish to include a "welcome" message in the upper-right corner of the page when the user is successfully logged in, like this:



How to do it
Implement the functionality described above using PHP. Be sure that your files work on ruby.ils.unc.edu with the Chrome web browser.

Be careful to make sure that you include a variable initialization block at the top of each file or code segment that initializes all variables used in that section of code. Also be sure that you have sanitized all $_GET and $_POST input including use of addslashes, striptags, and htmlentities AS APPROPRIATE.

Be sure to include a `youronyen-p3-readme` file that contains ONLY the URL for your "live" project, followed by a blank line.

All your code and links should work with these files, named as indicated, in the same directory (except your dbconnect.php script if you use one). In other words, if your files were placed together in some other directory, they should still work.

What to turn in and how
Make sure all of your files are in one directory on a Unix machine (for example, ruby.ils.unc.edu). cd into that directory and then type the following command to create a gzipped tar file:

```
tar cvzf youronyen-p3.tgz .
```

Note that after the "tgz" there is a space and then a period.
Check your tgz file with the following command:

```
tar tvzf youronyen-p3.tgz
```

When you are satisfied that your tgz file is okay, you can submit it electronically through Sakai by going to the Assignments area and finding the "P3" assignment. After you think you have submitted the assignment, I strongly recommend checking to be sure the file was uploaded correctly by downloading it from Sakai and verifying it again with the "tar tvzf" command. Keep in mind that if I cannot access your files, I cannot grade them.

If for some reason you need to re-submit your homework file, you must add a version number to your filename. Sakai is configured so that it will only accept 3 total submissions. Use the following file naming convention if you need to re-submit:

Your first submission:	`youronyen-p3.tgz`
Your second submission:	`youronyen-p3-v2.tgz`
Your third submission:	`youronyen-p3-v3.tgz`

Sakai is also configured with a due date and an "accept until" date. Submissions received after the due date may receive a 10% penalty per day.

If for some reason you are unable to submit an assignment to Sakai, as a last resort you may email it to the instructor along with a note about the problem you encountered. Then, as soon as you are able to, it is your responsibility to submit the exact same assignment to Sakai. The email serves as a record that you tried to submit the assignment on time, but to receive credit, your assignment must be uploaded to Sakai.