Course Equivalent Web Application

Sarah Read and Cis Xing

Spring 2015 Independent Study

We began creating a web application for the Computer Science Department of Mount Holyoke College to allow faculty to organize what courses at other college count towards the MHC Computer Science major. Professors will be allowed to add new courses they have evaluated, search for previously evaluated classes, and edit courses when course details have changed. This project was started from scratch using PHP among other technologies.

Below are the details of what can currently be done with the Course Equivalent Web Application, a detailed explanation of the database design, a list of all of the technologies that were used, and a summary of the future plans for this project.

**Features:**

Search:



Figure 1. Screenshot of the search page

The search page allows users to search for courses by

* Course Title
* Course Number
* Institution
* MHC Equivalent Course

Users must search by at least one category.

In addition the user can sort the results based on the following

* Institution
* Course Title
* Course Number

The user can choose to sort by one, two or three of the above categories. If they do not select a sorting preference the courses will be sorted by institution so that the results are sorted in some form. Otherwise they would be displayed in the order they are found in the database, which doesn’t make much sense.

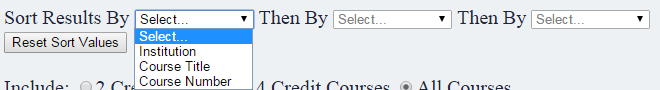


Figure 2. When the user selects the first option, that option is removed from the second and third drop downs since it does not make sense to sort by the same thing twice. What is chosen in the second drop down is removed from the third. Only the current drop down is enabled. The user can reset values at any time.

The user can also choose to include all courses, 4 credit courses, or 2 credit courses.



Figure 3. Radio button to include courses

The search results are a union of all of information entered. The results are displayed in a table.

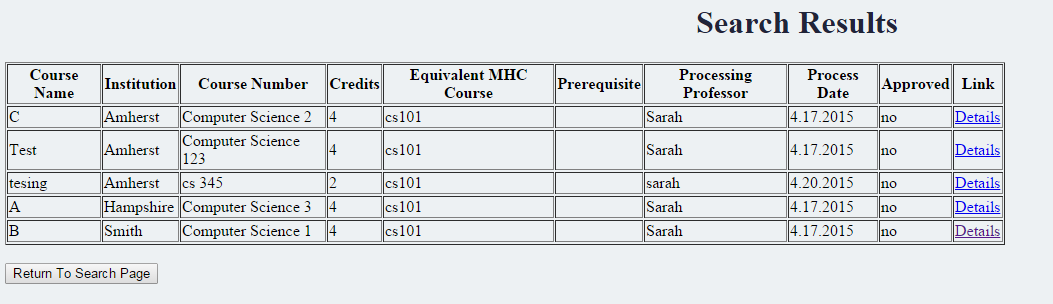


Figure 4. Search results when searching for MHC Equivalent Course CS 101.

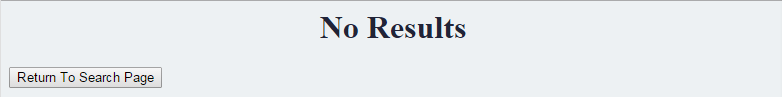


Figure 5. If there are no results the page will display “No Results”

Add New Course:

Users can add a new course to the database by clicking on the “Add New Course” button on the search page.

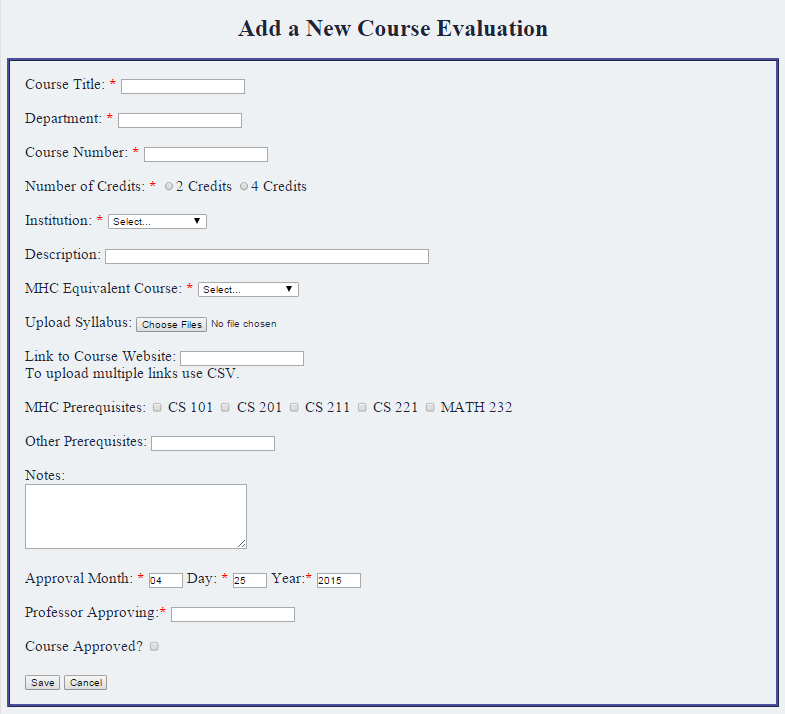


Figure 6. Screenshot of the Add Course page.

To add a new course the user must enter

* Course Title
* Department
* Course Number
* Number of credits
* Institution
* MHC Equivalent Course
* Approval Date (automatically filled in to the current date)
* Professor Approving

The user can also enter

* A course description
* Upload syllabi and other related documents (any file type, any number of documents)
* Links to a course website (use CSV to add multiple links)
* Any MHC prerequisites to the class
* Other prerequisites
* Notes

The user indicates that the course is approved by checking the checkbox next to “Course Approved?” If it is not approved the user should not check the checkbox.

The following categories are restricted:

* Month: must be an integer between 1 and 12
* Day: must be an integer between 1 and 31
* Year: must be an integer, must be no greater than the current year, must be no less than two years ago.
* Course Number: must be an integer
* Number of Credits: can only be 2 or 4 credits
* Institution: can only be Amherst, UMass Amherst, Smith, or Hampshire

Upon submitting the form a confirmation page will be displayed with a summary of the data entered. From here the user can return to the search page.



Figure 7. Screenshot of the add course confirmation page.

View Course Details:

The user can see course details by clicking on the “Details” link on the search results page

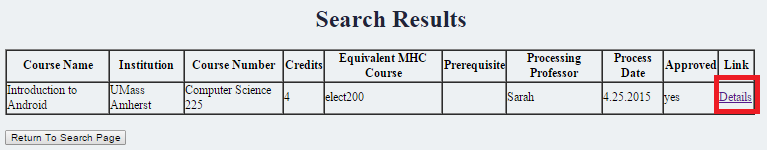


Figure 8. Search results page, red square indicates link to see course details.

The user will be taken to a page that displays the information currently in the database for the selected course.



Figure 9. Course Details page.

From here the user can download syllabi individually or download all documents at one time. The user can also go to any websites linked to the course. There are also buttons to allow the user to delete the class, edit the class, or return to the search page.

Edit Course Details:

From the course details page the user can go to the Edit Course Details page.

This page allows users to make changes to the data already in the database. The form is filled in with the current data and when the user submits the page the new information is changed in the database.

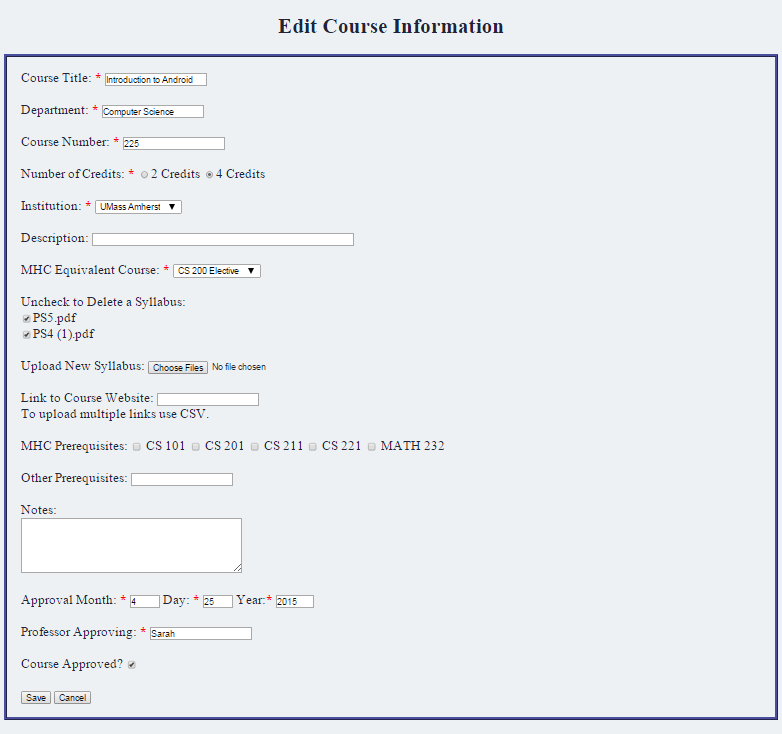


Figure 10. Screenshot of Edit Course page.

The following categories can be changed by simply changing the information in the textbox/checkbox/radio button/drop down:

* Course Title
* Department
* Course Number
* Number of Credits
* Institution
* Description
* MHC Equivalent Course
* Links
* MHC Prerequisites
* Other Prerequisites
* Notes
* Approval Month
* Professor Approving
* Course Approved

The documents can be deleted by unchecking the box next to the name. The document will be removed from the database and from the server. Documents can be uploaded in the same way they were in the “Add New Course” page.

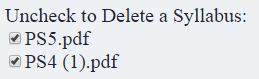


Figure 11. Uncheck the box next to the document that is to be deleted to remove it from the class.

When the user submits the page a confirmation page appears. This page displays all of the changes made to the course information.

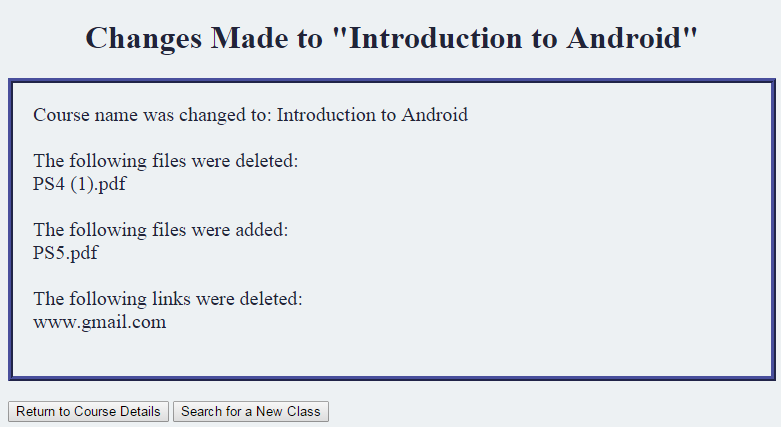


Figure 12. Confirmation page displaying the changes made to the course.

From here the user can return to see the course details or can return to the search page.

Delete:

To delete a class the user must go to the details page of the class they want to delete and then click on the “Delete This Class” button. This will take you to a page to confirm the action.

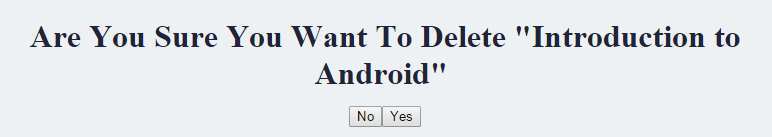


Figure 13. Delete class confirmation page.

After the user confirms deletion they will be notified and then they can return to the search page.

If the user does not want to delete the class they should click no and they will be returned to the course details page.

**Database:**

There are 4 MySQLite databases used for this application:

* Course Details Database: Holds most information for each class
* Documents Database: Holds the path to each document on the server and the class each document is associated with
* Links Database: Holds each link and the class is it associated with
* Counter Database: Saves the id of the last class created

Course Details:

This database holds all information about each class except for the documents and links associated with it. The data is as follows:

* Id : INT(8) UNSIGNED PRIMARY KEY
  + The key is manually set each time a new course is added. This allows the key to be easily found for use in the Documents and Links database. The last key value is kept track of in the counters database. For each new course the key is the last key value plus one.
* Name: VARCHAR(60)
  + Holds the name of the course.
* Department: VARCHAR(50)
  + Holds the name of the department that offers the course.
* Number: VARCHAR(10)
  + Holds the course number. This is later restricted to be an integer.
* Credits: INT
  + The number of credits the course is. This is set to be either 2 or 4.
* Institution: VARCHAR(30)
  + This is the institution where the course is offered. Currently this is restricted to be Amherst, UMass Amherst, Smith, or Hampshire.
* Description: TEXT
  + This is a short description of the course.
* MHC\_Course: VARCHAR(20)
  + This is the course offered at MHC that the off campus course will count as. Can be either CS 101, CS 201, CS 211, CS 221, MATH 232, CS 312, CS 322, or a CS 200 or 300 elective.
* Prereq101: BOOLEAN
  + True if CS 101 is a prerequisite, false if it is not.
* Prereq201: BOOLEAN
  + True if CS 201 is a prerequisite, false if it is not.
* Prereq211: BOOLEAN
  + True if CS 211 is a prerequisite, false if it is not.
* Prereq221: BOOLEAN
  + True if CS 221 is a prerequisite, false if it is not.
* Prereq\_Math: BOOLEAN
  + True if MATH 232 (Discrete Math) is a prerequisite, false if it is not.
* Prof\_Prereq: TEXT
  + A list of other prerequisites that professor listed for the class
* Notes: TEXT
  + Any notes about the class (such as “will count for an elective only if taken in conjunction with another class”, etc.)
* Day: SMALLINT
  + Day of approval
* Month: SMALLINT
  + Month of approval
* Year: SMALLINT
  + Year of approval
* Professor: VARCHAR(40)
  + The name of the professor who is approving the class.
* Approved: BOOLEAN
  + True if the class is approved to count as the desired course, false if not.
* Reg\_Date: TIMESTAMP
  + Time entered into the database.

When the user edits a class the data is updated in the database and the primary key remains the same. When a class is deleted the information is removed. The primary key will not be given to a different class given the current implementation.

Documents:

This database holds the information needed for each document:

* Id: INT(8) UNSIGNED AUTO\_INCREMENT PRIMARY KEY
  + Primary key for the document. This automatically created.
* Class\_Id: INT(8) UNSIGNED
  + The id of the class that the given document is associated with
* Syllabus: TEXT
  + The path to the syllabus on the server
* Syllabus\_Type: VARCHAR(30)
  + The type of document (.pdf, .txt, etc.)
* Syllabus\_Size: INT
  + The size of the document
* Syllabus\_Name: VARCHAR(60)
  + The name of the document
* Reg\_Date: TIMESTAMP
  + The time when the document was saved

The reason there is a separate database for the documents is because the user can upload as many documents as they want for each class. When multiple documents are uploaded for one class the application loops through each of them and creates a new entry for each. All documents associated with the same class will have the same class id. The path to each document will depend on the class id and the number of classes associated with the class, to ensure unique paths for every document.

When a document is removed from a class it is deleted from the database and the server. There is no time when any of this information is updated in the application.

Links:

This database keeps track of the links associated with classes, similar to how the database for documents works. The following data is saved:

* Id: INT(8) UNSIGNED AUTO\_INCREMENT PRIMARY KEY
  + The primary key for the link
* Class\_Id: INT(8) UNSIGNED
  + The id of the class that the given document is associated with
* Link: TEXT
  + Link for the website related to the given class

When a link is removed from a class it is deleted from the database. There is no time when any of this information is updated in the application.

Counter:

The counter keeps track of the last used id for the Course Details database. The database only ever has one row of information that is updated each time a course is added. This ensures unique primary keys for each course and also allows the program to know what the key is in order to allow the documents and links databases to know the value of the key. The data saved is as follows:

* Id: INT(8) UNSIGNED PRIMARY KEY
  + Primary key for the entry
* Cur\_Val: INT(8)
  + The Id value of the last course that was added

**Technologies:**

The following technologies were used in this project:

* PHP: Most of the code was written in PHP, including data manipulation and the creation and editing of the sql databases.
* HTML: Many HTML forms were used in this project.
* CSS: One very simple style sheet was created to make all pages looks consistent.
* JavaScript: A few JavaScript functions were used to check information in the HTML forms before the page was submitted.
* MySQLite: Used for the database.

**Future Work:**

Course Equivalent Web Application:

There is still some work that needs to be done on this application, such as:

* Allow any institution to be used instead of just the other 4 colleges in the consortium.
  + In the future it would be good to include AIT Budapest and other institutions where students commonly study abroad or take summer courses.
  + This is currently not an option because we have not thought about a good way to execute this. Originally we discussed issues related to spelling of the institution, which caused us to create the application using the dropdown. For example: would professors know to enter UMass Amherst instead of University of Massachusetts Amherst or just UMass? With many possible spellings how would we make sure searching the database was done to include all possibilities?
  + Possible ways to fix this would be to create a way to add an institution to the dropdown so that spelling is always consistent.
* Allow users to choose whether to search by the union or intersection of the search parameters.
  + Currently the search page is hard coded to show the union of all search terms. This would not be too hard to change.
* Allow users to search using more parameters
* A lot of information is hard coded that should in the future be changed. For example:
  + Institution and MHC Equivalent Course dropdown options are hardcoded in the form. In the future it would be better to write a loop in php that would add each element to the form. We would then save the information needed in the global\_vars.php file and refer to it whenever needed.
  + Database names, elements, etc. are hardcoded and used all over the place. It might be beneficial to also change these to variables to names could be changed as needed.
* Change the Month, Day, Year entry on the add course and edit course pages to be a calendar.
  + It was created as a textbox to test everything and has yet to be changed to a calendar selector.
* The table on the Search Results page is messy
  + If the page is full screen the table does not fill the entire screen
  + If the page is half screen the table does not fit to the screen size, but rather the user has to scroll right to see everything.
  + When the style sheet is used on this page and the border is included the table does not fit within the border. The border is not currently being displayed on this page, but for consistency it should.
  + Also when there are not prerequisites the column should display “none” and instead is left empty
* Add security measures
  + Most of the application is not protecting against SQL injection and other things. This is important to add, especially if users other than the professors can gain access.
* Error messages should be removed
  + When a user refreshes a page where the data was sent to the page via “Post” the data is not sent again. If error messages are turned on then the user will see an error for each item that is drawing on the data sent via post. This needs to be fixed.

Related Application Development:

The next step would be to add functionality to allow student to request for courses at other colleges to be counted towards their major. The students should not have direct access to the data on course equivalence. This will result in the creation of accounts. Students will be able to log in and request a class to count and professors will be able to log in and see the database of current evaluated classes and the current student requests.

The department would also like to add to this application a way to allow students and professors to keep track of each student’s progress towards completing the major. This app would keep track of all majors and minors completed requirements.