CSE3403 Fall 2008

Project #2:

Due: November 19,24,26, 2008

Weight 25%

This project involves working with ASP .NET, ADO .NET and Web Services. You will build is a 3-tier application with and use a database management system in order to store and work with your data. For this project you should use the Access DBMS.

This assignment assumes that we have a database in a typical university setting. The DB contains the following tables:

Table "*Courses*", with the following schema:
Courses: [
courseNumber,
courseMajor,
courseName,
courseCredits,
courseDescription,
]
where,

- courseNumber, is the number of the course, e.g. 1030.
- courseMajor, is the major, e.g. CSE
- courseName, is the course name, e.g. "Introduction to Computer Science II"
- courseCredits, is the number of credits, e.g. 3.0
- courseDescription, is the course description (same as in A1)

Table "*Prerequisites*", with the following schema:
Prerequisites : [
courseNumber,
courseMajor,
prerequisiteCourseNumber,
prerequisiteCourseMajor
]
where,

- courseNumber, is a course number as in the "Courses" table.
- courseMajor, is a major, as in the Courses table.
- prerequisiteCourseNumber, is the course number of a course which is prerequisite to the <courseNumber, courseMajor> pair of the above two columns of this table.

- prerequisiteCourseMajor, is the major of the prerequisite.

Note that a < courseNumber, courseMajor > pair in this table may have more than one prerequisite < prerequisiteCourseNumber, prerequisiteCourseMajor > pair.

Table "*Enrolled*". Decide the schema of table Enrolled based on the assignment's description. Table "Enrolled" stores information about courses that students are currently enrolled (i.e., students take those courses during the current term).

Table "*Completed*". Decide the schema of table Completed based on the assignment's description. Table Completed stores information about courses that students have already taken (i.e., completed during previous terms).

Once you have the above tables, then you should create a 3-tier application that incorporates ASP .NET and ADO .NET and performs the tasks described below.

You may need to have more tables in your database to accomplish the requirements of this assignment. The design of such tables, if needed, is up to you.

Tasks

(Task 1.) Given a course (course number and major), retrieve and display all the prerequisites of that course. For each course which is a prerequisite you should display the course number, major, and name of that course.

(Task 2.) [***student transcript***] Produce a transcript of a student. Given a student id, retrieve and display all courses that this student has completed, as well as those courses that the student is currently enrolled. Display: course number, major, course name, and letter grade. For those courses that are currently in progress, do not display any letter grade (since there is none yet). Instead, display a message "in progress".

(Task 3.) [***component table***] Given a student id and course[1] (course number and major), display the **component table** for the student for the specified course. A *component table* is a table that contains all the evaluation

[1] Note, the specified course may be in the Enrolled table (i.e., currently in progress), or in the Completed table (i.e. completed at a previous term).

components of a course (for example, assignment 1, assignment 2, Test 1, Midterm exam, final exam, etc), together with their weight toward the total course mark, their due date, and the date that the mark was made available to the students (for those components that the evaluation has been already completed; the corresponding date field for the rest of the components is empty). A sample component table is shown in ***Appendix A.*** In your system, every course should have a component table associated with it. Once a student is enrolled in this course, then an instance of this course's component table should be created for this student (and filled accordingly with this student's performance data). **The component table of each course is information which should be stored in the DB.** Note, different courses usually have different components and, therefore, a different component table should be available for each course. Once a student is enrolled in a course, a copy of the corresponding course's component table should be created and filled in with the marks that the student achieves, progressively[2]. Once the student completes the course[3], the component table should be saved and kept for future reference. In this task, you should also calculate and display the *percentage achieved so far in the course for this student*. This is the total number of points already achieved by the student, divided by the maximum number of points that could have been achieved up to this time in the course. (for example, if components A1, A2 and midterm have been evaluated so far and the student achieved 5, 7 and 23 respectively, and the corresponding weights of those components are 10, 10 and 30, then the so far achieved percentage (for this student, in this course) is (5 + 7 + 23) / (10+ 10 + 30) = 35 / 50 = 70%. For displaying this information, provide two *progress bars* B1 and B2, such that B1 shows

[2] Therefore, there should be one component table serving as template for each course, and then out of each such template component table there would be several component tables, one for each student that took that course. A course's template component table only shows the evaluation components for that course, and their weights. It does not have values of marks achieved by any student, neither does it have any due dates and return dates values.

[3] At that time, that student would be removed from the Enrolled table and be inserted into the Completed table.

the so far achieved percentage (i.e., 70% in our example) and B2 shows the percentage covered so far in the course (i.e., 50% in our example).

(Task 4.) [***course component table transcript***] As in task 3, but for all students in a particular course. The input for this task should be a course (i.e., course number, major). The output should be component tables; one component table for each of the students enrolled in this course. Also, calculate and display the *average* of this class (for this course), for each of the components (that appear within the component table), as well as the overall average (i.e., the average of the totals of each component table).

(Task 5.) [***student component table transcript***] As in task 4, but for all courses of a particular student. The input for this task should be a student id. The output should be:

a. The component table for each of the courses that this student is currently enrolled. Also the average of the totals (achieved so far) of the component tables of this student.

b. The component table for each of the courses that this student has completed. Also the average of the totals of the component tables of this student.

(Task 6.) **[** ***inter-University student record service*** **]** Assume that several universities maintain student component tables and they have agreed to make this information available among themselves, on a demand basis, as Web Services[4]. For this task, we assume that there is an established consortium of universities such that each member-university has agreed to provide a Web Service that facilitates retrieval of a student's component tables for all the courses that the student attended in that university[5]. A typical architecture of such a Web Service is shown in ***Appendix B***. The consumer of the Web Service (caller of method `requestComponentTables`)

[4] Realistically, this can be quite useful, for the purposes of resolving issues of advanced standing, i.e., when a student wants to transfer from a university U1 to another university U2 and be given credit for some/all of his course the he/she completed at U1 (so that he/she does not have to repeat the same or a closely equivalent course in U2).

[5] We assume that all member-universities maintain student records that allow the retrieval of the related data for the purposes of constructing component tables.

supplies a student name and invokes the method through the provided Web service. The Web Service executes the method (which entails retrieving data from a database, assembling the component tables and then placing them into an array) and returns an array containing all component tables for (all courses of) that student. Then, the Web Service consumer receives the array (which is the returned value of the `requestComponentTables` method), extracts the component tables found in that array, and displays those components.

A general view of the architecture of the entire application (tasks 1-6) for this project is shown in ***Appendix C***.

Interface

The above tasks should be performed via an ASP .NET windows application with C# as the code behind part and which also uses the ADO .NET part of the .NET Framework. Also, a Web Service needs to be developed for task 6, and incorporated into your main ASP.NET application. Your program should have a GUI equipped with buttons and other components of your choice, capable to perform the above tasks.

Handling your data

In this project, you should use the Access DBMS to store your data as applicable, and access it via ADO.NET.

Note that you would need to have two databases for this project. One is the database that supports tasks 1 through 5. The other is the database that supports task 6. The latter database resides at the Web Service site.

Some specifics for the user interface:

1. **To perform task 1:** provide a TextBox in which the user should type the course number and another TextBox in which the user should type the course major.

Provide user-friendly labels that guide the user as to what is required to be typed in each textbox. Display the result (prerequisites of the input course) in a textbox (with scrollbars as applicable).

2. **To perform task 2:** provide a textbox for input of the student id. Display the result in a textbox (with scrollbars as applicable).
3. **To perform task 3:** provide textboxes for entering the course (number and major) and the student id. You can choose the display format of the component table to be as you like, but include all specified columns and values. Also, include the student id (and name, if you like) and the course (course number and major). And do not forget the progress bars, as specified in the description of task 3.
4. **To perform task 4:** provide textboxes for the input (course number and major). The display format of the output is up to you, but makes it viewer-friendly, of course.
5. **to perform task 5:** similar to performing task 4.
6. **To perform task 6:** provide a textbox for the input of the student name and a button for triggering the retrieval (which will be done via the Web Service). The display format of the output is up to you, as in the previous tasks.
7. **Your program should be accessible by opening the corresponding application in visual studio (which should be found under the `WebApplicationProject` folder (see under "what to submit")), establish the reference to the database (if broken due to the file transfers) and then build and run it.**

What to submit

Submit the following items:

1) `Project.mdb` – this is the Access database that contains the tables, as described above.
2) `WSDB.mdb` – this is the Access database that resides at the Web Service site and supports task 6.
3) `WebServiceProject` – this should be a folder created by Visual Studio .NET when you develop your Web Service for this project. In your computer, this

folder typically resides under C:\Inetpub\wwwroot. (i.e. same place as the Web ASP application). This folder contains all the code that is produced when you develop your web service. Make sure that under this folder you have generated the folder "Web References" that contains the `.wsdl` files.

4) `WebApplicationProject` – this should be a folder created by Visual Studio .NET when you develop your program. This folder contains the "project", the "solution" and the source code (.aspx and .aspx.cs) files, as well as many other files generated by visual studio.

5) A **readme.doc** file which provides the items listed below. This file is very important for the evaluation of your project, since it shows all the work you have done. Therefore, the more detail and care you put into this part, the better your project will be.

   a. Sample runs that demonstrate all the features of your application. Screenshots are highly recommended (in fact, how else can you do it?)
   b. Description of all the components of your application, including the.aspx, .cs, and .asmx files. Provide a brief summary of what each of those components does and how it relates to the rest of the system. An architecture outlook similar to the one of Appendix C, but with showing all the related components of your system, would be nice and sufficient.
   c. Schemas of your database tables, for both databases.
   d. Instructions on how to install and run your application.

How to submit your project.

Once you complete your project, you should copy all the files in a CD and submit the CD. Write your names and student numbers on the CD. Wrap the CD safely (either using an appropriate plastic case or some reasonably secure covering) and drop it in the 3403 drop box found outside the CSE dept main office. Make sure that you write your CD in such a way that is accessible by all standard CD drives. If you do the assignment as a group, submit only one CD. Make sure that you keep a copy of your software because your CD will be kept as a record of your work. Also, drop all required hard copies in the

same drop box as your CD. Identify clearly each submitted piece with your name(s) and student IDs.

How you can work.

You can do this project in groups of up to 5 (five) students per group. In case that you work in a group, you should also provide an individual report that describes – in your opinion, the amount of work done by each of the group members, including yourself. This report will also be used in assessing the work of each student individually, within your group.

Evaluation.

The evaluation of your project will be done on the basis of:

- The detail of the work you did.
- The comprehensiveness of your work.
- The quality of your written report.
- The overall quality of your work.
- As stated in class and in the course web site, evaluation will also be relative to all other projects.

Possible risks.

- If one or more members of your group happen to drop the course, the remaining members of the group are responsible for completing the project.
- Once a group is formed, members of a group cannot move to another group, unless there is agreement in doing so by *all* involved (i.e., all members of the group that will lose a member and all members of the group that will gain a member).
- Exchanging ideas between different groups is OK, but be reminded that evaluation is relative. (Your communicated ideas may give a better advantage to someone else).
- Copying code from one group to another is not allowed.

- Copying (non-copyrighted or copyrighted-but-allowed-to-use) code from the web is allowed, provided that you supply the reference (web address) from where you copied, give credit to the original author of the code, and specify clearly the parts of your code that have been copied.
- If applicable, copying chunks of text from articles/books/web and pasting it into your written report is not allowed, in general. If you need to do so, you should clearly indicate that this is copied text, by surrounding it with quotes and also by providing reference to the source from where you copied, including the exact page of the article where the copied text is found.

Appendix A (sample component table)

Component	Weight	Achieved	Due	Returned
A1	5	3	July 10, 2008	July 15, 2008
A2	5	4	July 20, 2008	July 30, 2008
A3	5		Aug 20, 2008	Aug 28, 2008
Test 1	15		Sep 30, 2008	Sep 15, 2008
Test 2	15	12.5	Oct 10, 2008	
Test 3	20		Nov 2, 2008	
Final Exam	35		Dec 20, 2008	
Total	100	19.5		

Notes:

- The field [A3, Achieved] is empty, meaning that this student did not submit A3. Same for the field [Test1, Achieved].
- The field [Test3, Returned] is empty, meaning that the marks for Test 3 have not been returned yet.

Appendix B (inter-University Web Service architecture)

This Web Service is offered by each member-university of the inter-University consortium.

< Web Service offered at Some University>

1. receive request. A request is a call to method `requestComponentTables( string studentName).`

   This method should have the header

   ```
   public ComponentTable [] requestComponentTables( string studentName).
   ```

   I.e., the method receives as parameter a studentName, and returns an array of ComponentTable objects. It is obviously assumed that both the Web Service as well as the consumer of the Web Service are aware about class ComponentTable[6].

2. Query the local database for data of that student.
3. Retrieve data (this is like Task 5, performed by a web service).
4. Create Component Tables with the retrieved data and place the Component Tables into an array.
5. Send the array to the consumer of the web service.
6. (Done.)

[6] In a real-life application, this may not be very realistic, i.e. the consumer of the web service may not be aware of the structure of class ComponentTable. In that case, additional information needs to be communicated. This can be done via appropriate SOAP messages, or XML schema documents, if necessary. However, for keeping the complexity of this assignment at a reasonably easy level, we make is so that we don't have to deal with these issues.

Appendix C (general architecture of this project)