

Assignment 4: Web Services Based Testing System

Course ISE 431: Distributed Information Systems

Due January 24, 2010 at 11:59pm

This assignment is a followup to the testing system that you developed for the previous assignment. The major difference here is that instead of basing your system on Java's Remote Method Invocation RPC system you will base it on C# Web Services.

To remind you, the functional description of the testing system is included below.

1 System Description

The remote testing application will enable a teacher to prepare a test based on multiple choice (American) questions which students may answer on their computers. The goal of the testing system is to enable a teacher to prepare a multiple choice test, enable the students to take the test securely, and automatically grade the tests.

As a beginning you will need to design a Question class which represents a multiple choice question. Questions have a textual part (the question) and may have five (1-5) options to select from, each with a short text to display next to the number. Questions also have a number which enables the student and teachers to more easily refer to them later.

Tests are made up of a list of questions. There is no fixed length for the test, but you may presume it to be less than 10,000 (in other words a standard `int` will suffice to index them). You may assume that each Question will not appear more than once on the same Test.

A teacher prepares a list of questions along with an answer key as a formatted text file which the Test server will read and load into memory. The teacher also prepares a list of students who may take the test as a formatted text file. Each student is identified in the file with a username and password. The usernames and passwords are both eight characters long and are made up of alphanumeric characters. The file also contains a special Teacher username and password.

Once the Test server has loaded the information from the files, students may log in to the Test server to begin taking the test using their student interface. Students logs in by providing a username and password. The interface then shows the first question on the test which the student can then answer (using buttons, radio buttons, text entry, etc). Students should be allowed to skip questions if they wish. When the student has finished the test he clicks on a "Finished" button (or menu item, etc) to indicate completion. The student's answers are then sent back to the Test server for grading. The student is shown an "Ok" message to indicate that the answers have been sent.

If a student logs again to take the test again, all of his first answers are erased. Only the new answers are recorded.

Teachers can log in to the Test server using a teacher interface by providing the Teacher username and password. The teacher can then retrieve information about:

- How many students took the test
- See a list of the students who have and have not taken the test
- See the grades for each student (percentage and correct/total ratio)

Assume that multiple students will be taking the test at the same time. Teachers **will not log in** to see results while there are students currently taking the test.

The Test server will only offer one test at a time.

2 What to do

Your job is to build all of the parts needed for the above system using Web Services in C#. You must implement at the very least:

- A Question struct/class
- A Test struct/class
- A Grade struct/class which is to be used (at least) to pass grade information to the Teacher interface. The Grade struct/class should have (at least) the following members:
 - Student Id
 - Number of test questions answered correctly
 - Total number of test questions
 - Percentage grade for the student
- A Web Service class (may be Test or something else)
- A student user interface
- A teacher user interface (may be the same as the student if it provides the necessary functionality)
- A mechanism for user authentication using passwords over Web Services
- A parsing mechanism to read questions, user information, and service configuration from files.

You may use C# .NET or any other XML Web Services infrastructure for this assignment.

There are three text files on the course web page which give sample inputs for questions, user accounts, and server configuration. The server configuration file should be used by the server to set up its listening and by the client interfaces (student and teacher) to know on which IP and port to communicate.

You may decide when and how to grade the student submissions (whether immediately on submission or when the teacher logs in to see the grades).

Note: You should be aware that storing state between web service calls is not so straightforward. You may use simple file parsing and writing to help you. Alternatively you may read some of the .NET documentation to read about how to store data at the application level. This is not an essential part of the assignment, so you may solve it in any manner you find appropriate.

2.1 User Authentication

To guide you with respect to the authentication requirement, I will give you an example authentication scheme. For a student with username “tommy123” and password “okj”:

- The student accesses an unauthenticated function `Question[] getQuestions()`
- The student interface shows the questions allowing Tommy to answer them
- The student interface sends the answers to the server: `bool receiveAnswers(string username, string pwd, string[] answers)`

- The server returns “true” if the authentication succeeds and “false” otherwise.

Teachers should also have authenticated access. If the above scheme is used, the teacher access may be as follows:

- The teacher accesses a function: `Grade[] getGrades (string username, string pwd)`

Of course the scheme is not entirely secure since an attacker could guess the passwords and we are not using encryption, but it is sufficient for this assignment.

You may use the above authentication scheme or come up with your own of equivalent security.

2.2 Robustness

Make sure that your programs are robust against failures and errors. In particular, make sure the program properly handles the following situations, showing an error message, not crashing:

- Invalid username or password
- Invalid answer to a test question
- Multiple students taking the test at once.

2.3 Scenario Creation

Create a test scenario with the following parameters:

1. A Test with 10 multiple choice questions each with 5 options to select from.
2. Create an answer key for the test
3. Create a list of 5 students with usernames and passwords who may take the test
4. Create a teacher account
5. Have all five students take the test remotely. Make sure some students don't do well on the test - don't make it the case that all 5 get 100% on the test.

3 What to turn in by January 24, 2010

You may work in groups of two students. I will permit one group of three since the class has an odd number of students.

Each group must turn in:

- A list of the students in the group
- A detail of which student worked on what and for approximately how many hours
- The full source code for all parts of the system.
- A compiled version of the system (using the config file so I shouldn't need to recompile anything)
- A user guide sufficiently detailed for me to be able to install the system on my own computer and run it.

Each group must show me its system working, either on the lab computers or on a personal PC. We will arrange a time and place for the display at a later time. I reserve the right to bring my own scenario files (questions and user accounts) which you will use to test your system.

4 Grading

The weights for the grading will be assigned as follows:

- 10% authenticated communication
- 40% student interface correctness
- 40% teacher interface correctness
- 10% robustness (ability to recover from badly formed answers, parsing, management of concurrency)