Automatic Student Assessment in C# .NET Windows Forms/Console Applications

Mariana Evstatieva Goranova, Yordan Ilianov Danchev and Lyudmila Yordanova Stoyanova

Abstract – **We propose an approach that provides a solution to the problem for automatic assessment of exam task solutions in C #. This approach has advantages over the traditional test methods and aims to facilitate the work of both teachers and students. The teachers save time and the students have the potential to show a wide range actual knowledge.**

Keywords – **e-assessment, computer-based assessment, C# language**

I. Introduction

The e-assessment emphasizes the role of information technology relative to measuring students' learning [1]. Computers have been involved in educational assessment in the 1970s. Computer-based assessment provides the technological foundation for e-assessment. E-assessment helps the educational community to take advantage of technological developments and move forward to real assessments of complex skills and knowledge.

Fact is the growing interest in the field of information technologies. Programming is increasingly attracting the attention of young people, this requires opening new courses. All students at the end of each semester must be tested. The number of exam papers grows, and the difficulties of teachers to mark them are real. This requires the process of evaluating student's knowledge to be automated. Today there are many systems for designing and implementing an electronic evaluation of different types, but these are limited to evaluation of program tasks solutions [2]. The traditional test method is in written form – with pen and paper, which is both good and bad. On one hand, the sheet of paper and pen offers the possibility of evaluating the knowledge of a student without the intervention of external factors, but on the other – a significant portion of the test time is wasted in writing code that is generally automatically generated and less time remains for the verification of essential knowledge.

The presented application provides a solution to this problem for automated evaluation of exam task solutions in C#. It has the following advantages over traditional test methods: the assessment takes place in real time, within a few seconds; the student works on a computer and writes a code as he/she would do in a real work situation; the assessment evaluates the feasibility of an assignment problem; the time that the teacher uses for reading dozens of works will be shortened to minutes; the final mark is generated automatically as a sum of points for each task of the exam.

The access to the application is authorized. The registered users are divided into two categories: teachers and students. Both have access to the components for making test and changing current password while the members of the "teacher" category in addition have access to: user registration, creating or modifying test release, reviewing test results, starting/stopping the test.

M. Goranova is with the Department of Programming and Computer Technologies, Faculty of Computer Systems and Control, Technical University of Sofia, 8 Kliment Ohridski blvd., 1000 Sofia, Bulgaria, e-mail: mgor@tu-sofia.bg

Y. Danchev is with the Faculty of Computer Systems and Control, Technical University of Sofia, 8 Kliment Ohridski blvd., 1000 Sofia, Bulgaria, e-mail: yordan.danchev.w@gmail.com

L. Stoyanova is with the Department of Programming and Computer Technologies, Faculty of Computer Systems and Control, Technical University of Sofia, 8 Kliment Ohridski blvd., 1000 Sofia, Bulgaria, e-mail: lstoyanova@tu-sofia.bg

II. Functional Requirements

The aim of the application is to ensure an automatic assessment of programming assignments written in .NET C# Windows Forms / Console applications from students and present the results in real time, providing the ability to create and modify tests, display the results of the test with the options to filter by group or by student, to control the access to the tests.

Test is the overall unit that incorporates at least one and up to ten subtests. Each subtest has a logic that checks a part of the task and determines whether the task performs the needed condition or not. Every subtest has assigned points that are added to the user's final result upon successful passing.

A. Tests

The student can select a test and send the implementation to the server where the application can be assessed. This file has to be archived in zip format. After assessing the server returns back to the student information for his/her score in points (100 by 100) and information on the outcome of each subtest. The student has the right to send an unlimited number of solutions to the problem, but only the results of the latest sent solution are recorded. Students have only access to the tests currently active, while teachers have access to all the tests available in the database.

B. User Registration

This part of the application is only accessible to users from the "teacher" category. Here they can register new users (students) and have two options for it: registration of a single user or many users at once. While using the registration for a single user the password can be specified, or if not provided – one is automatically generated from the

username. When using the option for registration of multiple users at once the application generates password coincident with the username.

C. Starting the Test

When starting the test duration must be provided. It can be modified later if needed. The test begins at the moment of its setting and ends at the end of the duration. Users from the "student" category have access only to started tests (active) and cannot send solutions after the test duration runs out, while teachers are not affected by the test status (active or not).

D. Results

The report for the results of a particular test is presented in tabular form. There is the option to be filtered by either particular student or student group. The table contains data for: username, student name, total points, identity test number, results for each subtest, subtest point distribution, test name.

E. Login

To access the system the user must enter a username and a password and after entering can optionally change the password. With security in mind in case of wrong password or username the error message is always the same – wrong combination, nothing specific. A good security message can give additional helpful information to the user as long as it doesn't reveal anything private [3].

III. Application Design

Recognizing design as an explicit activity maximizes the benefit we will receive from it [4]. The application uses the model of the client-server architecture with the Microsoft technology Active Server Pages (ASP) for the server side. It has access to a local or a remote database and needs a disk or a remote server drive.

Multiple users can simultaneously use the proposed Web-based application (Figure 1). They are connected to the central web server that handles the requests and performs the necessary operations, realizing the functionality itself.

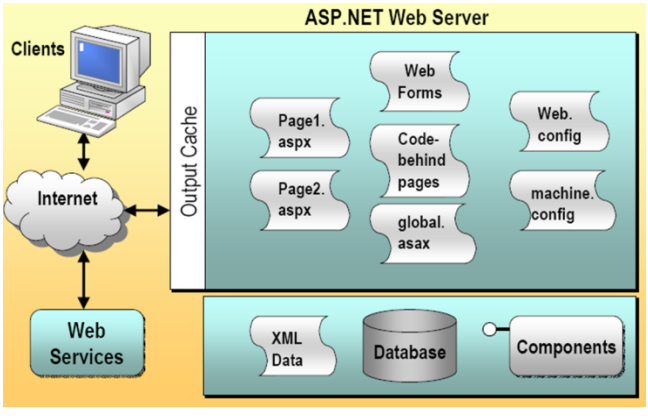


Fig. 1. WEB application architecture [5]

The basic functionalities of this client-server application include:

- Presentation – provides the user interface for the web application. This is implemented as ASP.NET Web Forms. The reason for the choice of this technology is that the presentational logic, i.e. the HTML code, is separated into .aspx pages and the server logic is located in separate .aspx.cs files (code-behind pages).
- Basic logic – manipulates the data and implements the main application functionality – the check of the program task.
- Database connectivity – allows the Web application to transfer the data to and from database sources.

The application stores information in a database and on a disk. The database contains three tables: results, tests, users. The storage on the disk is separated in two folders (Figure 2) – the contents of the tests (the folder "tests") and the solutions sent by students for assessment (the folder "users").

The "tests" folder contains a directory for each test including a new subdirectory for each subtest. These subdirectories are generated when the information for the specific subtest is filled in. The test shown in Figure 2 has identity number "1" and can have up to ten subtests. Each subtest directory contains a file or files with the code that will be inserted in the user's program.

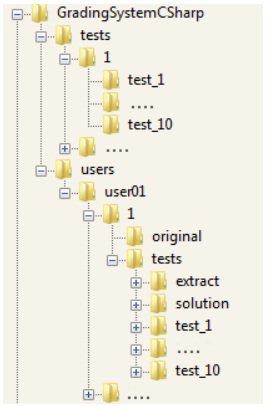


Fig. 2. Folder structure

The "users" folder contains a new subdirectory for each user with his/her username. A new subdirectory is created for each test with its identification number. Figure 2 shows the user with a username "user01" and the test with the identity number "1". The subdirectory "original" contains the original file uploaded by the user (the zip archive). The assessment process creates a second folder named "tests" with the following contents:

- extract – unzip the contents of the archive;
- solution – a cleaned up version of the solution for faster and correct prospecting;
- test_i – the modified code of the subtest i, the compiled solution and the result of the subtest.

The graphical user interface is implemented using ASP.NET Web Forms. The advantage of this technology is that it separates the presentation layer from the logic and two files are created for each page – one with the extension .aspx, where the appearance of the page is implemented

using HTML, Java Script, CSS, and a second that is .aspx.cs, where the presentation logic is implemented.

The temporary storage of the website is used for some variables. There are pages restricted only to users from the category “teachers” and a check for this is performed at the initialization of the page.

Upon successful logging in the user is redirected to the Default page. It contains a menu with hyperlinks to other pages of the application. When it is initialized the user rights are checked and according to them one of two views is shown – each containing only references to the pages the user has access to. Both views contain a button to exit the system and dispose the browser of any existing data for the user.

The Test page (Figure 3) is open to all users and is designed with the understanding that it will be the one with the most traffic and time spent on it. When the page is initialized the system checks whether the user is logged in. If not, it refers to the error page. If yes, it proceeds to load a drop-down menu with the available tests for this user. The Test page displays the user name in the upper left edge. All this is done once – only on the initial page load (i.e., when the page is not post back).

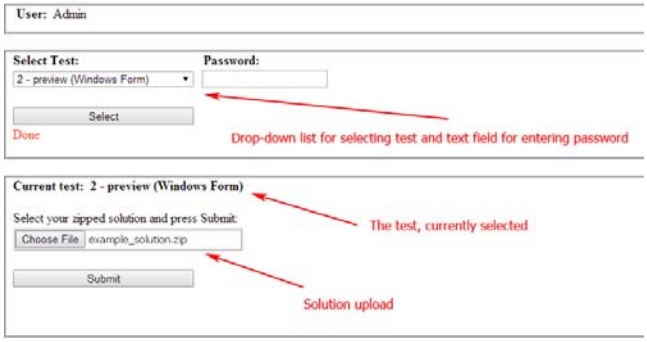


Fig. 3. Test page

The user can select a test from a drop-down list and then he/she must enter password in the text field next to it. After pressing the button “Select” (under the drop-down menu) the application checks for the combination of password and test. After a match the identity number of the test, the number of subtests, the points of each subtest, the type of the application are stored in the temporary storage. A label indicates which test is currently selected.

The last part of this page allows the user to navigate and select the solution of the task and press a button to submit it for assessment. After completion of the assessment the result is displayed in the bottom. Information necessary for the assessment is taken from the temporary storage.

The Create Test page (Figure 4) is only accessible to users of type “teacher”. This verification is done during the page initialization. The Load method loads all existing tests from the database in drop-down lists used in two of the three sections of the page provided for selection of a test.

The first part of the page allows the user to create a new test. When the user enters the required information and presses the button “Create” a new row in the database is created with an automatically generated identity number. The second part provides an opportunity to change some parameters of the test or delete it from the database. After creating a new test or modifying an existing one via the second section of the page the drop-down lists are reloaded. The third part is used to set the code of the subtests and the name of the files, in which the code will be inserted. When the user selects a test from the drop-down list and presses the “Select” button, a number of fields are opened in which the teacher can fill in the information for each subtest. When the user presses the ”Update” button at the bottom of the page the code is stored in the storage directory in the subdirectory "tests\[identity number of a test]\test_[number of subtest]” and the points are recorded in the database table “Tests” in the form of a string as the points for each subtest are separated by semicolons (;).

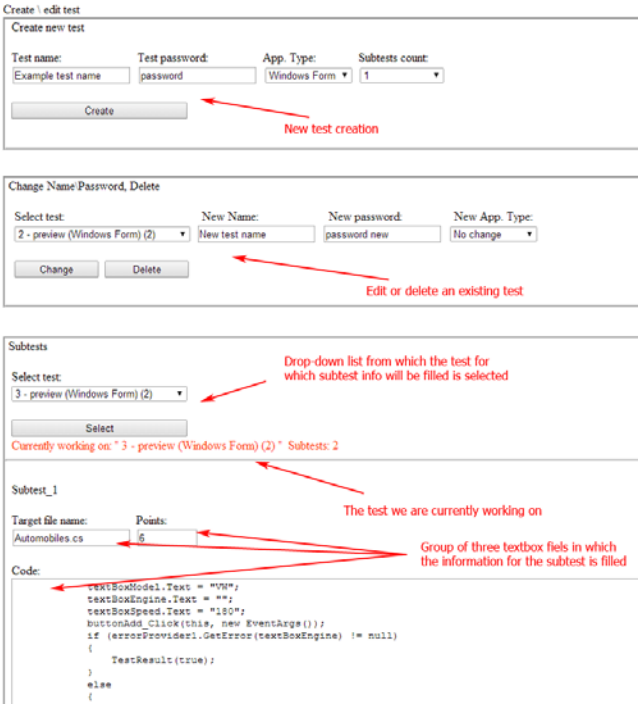


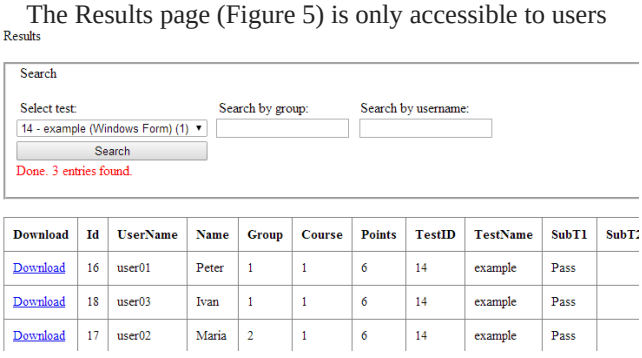
Fig. 4. Create test page

The Register page is only accessible to users with “teacher” rights. This page allows the registration of new users to the database, providing two ways:

- Registration of a single user.
- Registration of many users at a time – requires a specific format described on the page.

When the button “Add” of the corresponding section is pressed the data validation is performed. If the validation is successful and there is no an existing user with this user name the registration data is added to the database.

The Results page (Figure 5) is only accessible to users



Download	Id	UserName	Name	Group	Course	Points	TestID	TestName	SubT1	SubT2
Download	16	user01	Peter	1	1	6	14	example	Pass	
Download	18	user03	Ivan	1	1	6	14	example	Pass	
Download	17	user02	Maria	2	1	6	14	example	Pass	

Fig. 5. Results page

with “teacher” rights. Here by selecting a test from the drop-down list and pressing the “Search” button the results for the test will be presented in tabular form sorted by a group and a username. Search by a group or a username is also available.

The Start/Stop page is only accessible to users with “teacher” rights and is used to start and stop the tests. (Students can see and perform just started (active) tests).

The Load method loads a drop-down menu with all available tests. Detailed information about them is presented in tabular form.

After selecting a test from the drop-down menu, setting a duration (a value greater than 0 to start the test and a value less or equals than 0 to stop the test) and pressing the button the corresponding action is performed. The table information for the test is adjusted.

The Error page contains one line error message. When user does not have access to a specific page the system redirects him/her to this page.

IV. Assessment

The assessment is achieved by inserting the subtests program code in the user’s solution. For Windows Forms application the code is being inserted in the Load method. If such method does not exist it is generated and assigned to the corresponding event automatically. For console application the code is being inserted in the Main method. In both cases the insertion is done right at the end of the method so if there is a code written by the user it won’t be affected and will be executed first – before the inserted code. After the Load/Main method a print function is added.

The inserted code is written by the teacher and it can check any functionality of the solution, using the tools provided by the language. For passing result the print function must be called. This method of testing requires all names of classes, interfaces, fields – everything that might be used for the assessment to be clear in the task papers.

All the above is achieved first by making a copy of the solution in a new folder for each subtest. After that the file in which the code will be inserted is found. The application is searching for the closing bracket of the Load/Main method and the code is added before this bracket. This is followed by a build of the solution and its execution. During the execution the result is written in a file in the subtest folder.

In the following example the ErrorProvider control is used to validate the user input in a Windows Forms application. Part of the application needs to record the essential data of a car, including model, engine and speed (Figure 6). The application enforces the following business rule: if the needed data is not entered the ErrorProvider control sets error for the corresponding text box. The Subtest_1 validates the user input for the text box “Engine”. The left part of the figure shows the filled in form for the subtest. The field “Target file name” is the name of the class that contains the logic of the form and in this case is “Automobiles.cs”. The field “Points” can contain only integers between 0 and 100 and its function is self-explanatory. The field “Code” contains the code that is going to be inserted in the Load method. The right part presents the inserted code in the Load method of the student Windows solution and the bottom right part – the tested functionality. The Subtest_1 has been passed successfully – the student code matches the business rule specified by the application’s requirements and 6 points are added automatically to the final score.

VI. Conclusion

The proposed application presents one solution to the problem of automatic assessment of student programming assignments. It tests and gives marks to the students’ programming work objectively. The application provides controlled access by both authorization and password protection of the individual test, fully automated evaluation that takes only few seconds, all the information is stored and can be easily retrieved. The presented application benefits both students and teachers – by giving the first the opportunity to show their knowledge in ordinary conditions and the second – by drastically shortening the time needed for assessment. The proposed approach is the first step in the design of a test system. Additional efforts are needed to ensure the security of the system.

References

[1] http://www.tel-thesaurus.net/wiki/index.php/E-Assessment

[2] C. Douce, D. Livingstone, J. Orwell, *Automatic Test-based Assessment of Programming: A Review*, Journal on Educational Resources in Computing, Vol. 5, No 3, 2005.

[3] M. Howard, D. LeBlanc, *Writing Secure Code*, Second Edition, Microsoft Press, 2003.

[4] S. McConnell, *Code Complete: A Practical Handbook of Software Construction*, Second Edition, Microsoft Press, 2004.

[5] MSDN Training 2310B: Developing Microsoft ASP.NET Web Applications Using Visual Studio .NET.

Fig. 6. Subtest filled in form