Development of a Quiz

Implementation of a (Self-) Assessment Tool and its Integration in Moodle

https://doi.org/10.3991/ijet.v14i23.11484

Jakob Schweighofer, Behnam Taraghi, Martin Ebner (✉)
Graz University of Technology, Austria
martin.ebner@tugraz.at

Abstract—Technology Enhanced Learning has become more popular in recent times and many organizations and universities use it as a key instrument in various teaching and training scenarios. At the University of Technology of Graz, some courses require randomized quizzes where question variables can be assigned by arbitrary mathematical functions and this feature is missing in the current solutions. This article describes the development of a quiz application that can be integrated into Moodle by utilization of the Learning Tools Interoperability protocol (LTI). The PHP application is built to support programmable questions that can contain JavaScript and HTML code. Teachers are able to build interactive, randomized quizzes, in which the random variables can be assigned with complex mathematical functions. Furthermore, the application provides a programmable grading mechanism. With this mechanism, it is possible for students to self-assess their performance, as well as for teachers to formally assess their students' learning success automatically and send the results back to Moodle (or other LTI compatible consumer applications).

Keywords—e-Learning, Self-Assessment, Moodle, Learning Tools Interoperability, LTI Protocol

1 Introduction

Since the rise of personal computers and the internet, these two technologies have transformed nearly every aspect of life, one of which is learning. The importance of e-Learning technology has steadily grown since then and many universities use e-Learning as a key instrument in their education programs [1]. There is a broad variety of e-Learning systems present in the current online landscape. In this article the term Learning Management System (LMS) is of importance. Learning Management Systems usually incorporate user and user group management and are focused on the organization of courses by teachers [2]. The University of Technology of Graz (TU Graz) uses various online systems for educational purposes. One of these is the so-called "Teach Center", which is a Moodle instance. Moodle is a very popular LMS that gives users the possibility to manage online courses, learning materials, user groups and to perform quizzes. Unfortunately, the Moodle quizzes do not provide the possibility to include random variables that are generated by arbitrary mathematical functions. Moodle offers the possibility to use retries for its quizzes which are a desirable feature for any quiz application as this mechanism enables the learners to self-assess their learning achievement.

Self-assessment is a vital part of the self-regulated learning process and incorporates a feedback loop which is necessary for the perception of self-efficacy [3]. Learning is a cyclic process and assessment should not be seen as isolated subject [4].

For online self-assessment tools one can derive from this literature that retries are a necessary part of self-assessment tools and that the system must be designed with the whole learning process in mind. This means that students should be able to improve their performance and that teachers should be able to improve their learning material with insights derived from the results of the students.

Various studies have been conducted which show positive effects of e-Learning technologies and self-assessment. One of these studies [5] was particularly interesting because it was conducted in a similar scenario to the one present for the courses at the TU Graz, which do not yet have their requirements satisfied by the existing Moodle solution. The study was performed on one introductory university course, which was monitored for three years after the introduction of self-assessment quizzes during the semester. It revealed a statistical significant increase in final exam pass rates if a self-assessment strategy is pursued during the semester, compared to courses which did not implement such a self-assessment strategy.

To solve the problem, a custom PHP application is developed that can be integrated into Moodle. The quiz application offers a quiz building system, which enables the teachers, instructors, or other staff members to create questions that can contain HTML and JavaScript code. Thus, it enables the use of JavaScript mathematics libraries within the questions. The application offers retries for quizzes and can either be used as a pure self-assessment tool, or as a tool to perform traditional summative assessment supported by an automated grading mechanism.

2 Application Design & Features

The following feature requirements can be concluded from the problem description:

2.1 The application must be integrated in Moodle

Because Moodle is the central platform for teachers to organize their learning resources, the quiz application should be started from there. In Moodle it is possible to create external tool activities, which start external learning resources that can be anything from a homepage with information, to a comprehensive online course system. To achieve this, Moodle uses the Learning Tools Interoperability Protocol (LTI) which was developed by IMS Global. The LTI protocol standardizes the way of how different e-Learning systems communicate. The Moodle instance that launches the external tool, is called “LTI tool consumer” in this context and the external resource that is launched, is called the “LTI tool provider”. The quiz application which is described in more detail in this and in the next section also implements the LTI protocol and can be started as external tool activity from Moodle (or other LTI consumers). Therefore, the application can be used by any LTI consumer if it was set up for that consumer previously.

2.2 It must be possible to create quizzes, that can be launched from Moodle

To launch quizzes from the consuming Moodle environment, an LTI custom parameter is used. The custom parameter is called “launch_key” and is part of a launch request. It identifies a quiz within the quiz application. If a quiz with the provided launch key is found and it is valid to start the quiz, the user is redirected to the quiz start page. Figure 1 depicts the overall program architecture (simplified).

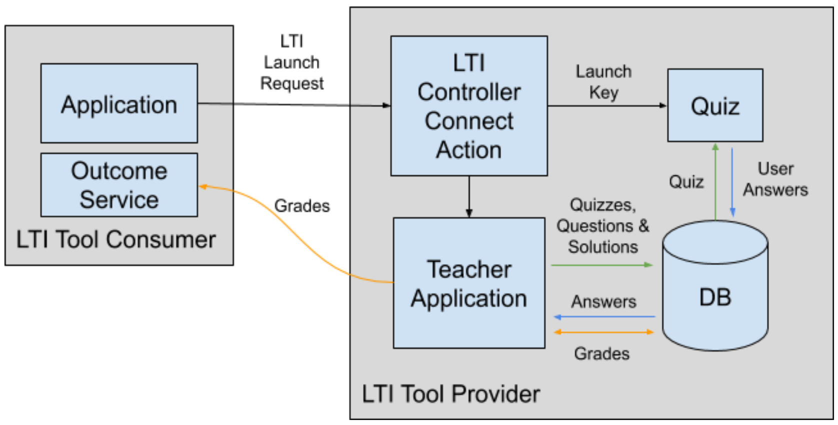


Fig. 1. LTI tool consumer and provider architecture

2.3 The application needs questions that can contain random variables, JavaScript and HTML

The questions are required to have the possibility to include variables in the question text so that each student can get different values for their questions on every attempt. These values can be assigned with JavaScript functions and are stored in a database at the time when a quiz is created. This prevents the revelation of the JavaScript code to students. After a completed quiz attempt, the students can see the correct solutions to the questions of the last completed attempt, if the teacher enables this option for the quiz. Furthermore, the teachers can define how many retries are allowed for the quizzes and in what time frame they must be done.

2.4 The application needs a retry mechanism for quizzes

Retries are a necessary part of self-assessment applications. Therefore, the quiz application must incorporate a retry mechanism to allow that quizzes can be tried multiple times. The teachers should be given the possibility to set the number of retries for each quiz so that a self-assessment (multiple tries) and a classical summative assessment scenario is achieved.

2.5 The application needs a programmable, automated grading mechanism

An automated grading mechanism is needed to efficiently grade hundreds of students and enable a self-assessment scenario. The application needs a flexible grading mechanism that can compare the students' answers with the correct solutions defined by the teacher. The implementation of that mechanism will be done by the teachers and in JavaScript.

2.6 Features

The main focus of the quiz application is to provide an easy-to-use interface to create reusable and flexible learning materials. Additionally, the teachers can set up grading functions (JavaScript) so that answers given by the students, can be graded automatically. This saves time for the teachers and makes it easier to assess the learning achievements of hundreds of students at once. The main focus in regard to the students' experience was to provide an intuitive and visually appealing quiz, that is easily comprehensible and easy to use. The application is designed to enable retries on quizzes so that it could be used as a pure self-assessment tool. In addition, it incorporates a grading interface to calculate all students' grades automatically and send them back to the connected LTI consumer and can also be used in a summative assessment scenario.

The main features of the application are: creating questions (teacher), creating quizzes (teacher), attempting quizzes (student), grading student answers (teacher) and publishing quiz results (teacher).

3 Development of the Prototype

The quiz application is a PHP web-application that was developed using Zend Framework 3 and MariaDB. The Zend Framework is one of the most long-lived PHP frameworks and well established. It makes use of the for web-projects very common MVC (Model View Controller) pattern to structure the program code. MariaDB is a performance-optimized successor of MySQL database and nearly fully compatible with MySQL. To abstract the database representation from the application the Doctrine ORM (Object Relationship Mapper) was used. This enables to program an application without the need to know how the data is stored and provides convenient methods for working with object relations. For the LTI part of the application, the official

LTI PHP framework by IMS Global was used. The LTI framework handles the requests that are received by an LTI consumer and verifies the authenticity of the incoming request by validating that the message is valid. The framework uses OAuth 1.0 body signing to sign the requests that can be then verified by calculating the applied signature again with the use of a shared secret on the receiving end of the connection. When the LTI framework has validated the request, it is the job of the LTI provider that was launched, to establish a user session (see [6]).

The quiz application was developed for two main stakeholder groups: students that do quizzes and teachers who create quizzes. The different user groups are represented in a hierarchical role-based access control (RBAC) system. When a user connects to the application by clicking on an external tool activity in the LTI consumer, the user is logged in and is assigned to one or multiple roles present in the application. These application roles correspond to the standard LTI roles “Instructor” and “Learner”.

A special requirement for the application was that questions must have the option to include HTML and JavaScript code. Furthermore, it should be possible to use variables in the questions. All parts that make up a question, as for example the question text, can be imagined as functions that produce the content of a question. The function “T” produces an instance of question text for a quiz attempt based on the variables for the text and is therefore called “T(Q)”. The values for the variables are calculated by a JavaScript function that is provided by the creator of the question. This function is called the question parameter generator function and produces instances of a JavaScript object. These instances are called “Q”. A second JavaScript function computes the solution for the question with the values of the instance Q. It is called the question solution generator function S(Q) and produces an instance of the JavaScript object “S” for every Q. This JavaScript code is executed when the question is saved and computes a set of the specified values and the correct solution to every value. Additionally, the user input area can also be defined in HTML. It is important to choose the names of the fields properly, as they are referenced by their names. See an example of the functions T, Q, U and S below:

3.1 Example of a question text T(Q)

```
  How strong is the force of gravity for an object with
the mass {{mass}} kg?
```

This is an example question text containing one variable “mass” which is surrounded by curly brackets to mark the variable. This placeholder is replaced by actual values from the generated parameter instance Q, which is selected randomly from the set of instances defined for the question on the quiz when the text is rendered in a preview or at a quiz attempt.

3.2 Example of a simple parameter function Q

```
  var Q = {
   mass: Math.floor(1 + Math.random() * Math.floor(150))
```

```
};
return Q;
```

For every question of a quiz, the teachers define how many parameter instances should be created. The number of instances defines how often the parameter function is executed. The example parameter function creates an object Q with a random variable named "mass" within the range from 1 to 150. One instance (Q) of this parameter will be inserted into the "{{mass}}" placeholder in the question text. In this example, the function that assigns the "mass" variable is rather simple but as soon as for example combinations of trigonometric functions are needed to compute the values, this is not possible in Moodle anymore.

3.3 Example of the corresponding solution function S(Q)

```
var S = {
 force: (Q.mass * 9.81).toFixed(2)
};
return S;
```

The solution object can contain multiple solution values that refer to a specific solution for the input Q.

Example of the user input area content U:

```
<input name="force" value="" /> N
```

The content of the user input area input field is rendered in the quiz page within an HTML form that is serialized and posted to a controller when a question is submitted.

Once the parameter and solution instances are computed, they are serialized and sent to a PHP controller, which stores them in the database. To provide a meaningful self-assessment experience, teachers can define how large the set of generated instances of Q is and also define the correct solution to the problems presented to the student. The solution can also contain HTML and JavaScript and is defined as solution guide SG(Q). To let the students know whether their answers were correct, the teacher can specify a grading function "G(U, S)". The grading function has the correct solution instance S and the user answer U available. The teacher can define a grading criterion in form of a JavaScript function that compares the correct solution with the student answer. After the comparison the function must return a value between 0.0 and 1.0, where 1.0 resembles 100% correctness. See simple examples of the grading function and the solution guide below.

3.4 Example of the corresponding solution function G(U, S)

```
if(U.force == S.force) {
 return 1.0;
} else if( Math.abs(S.force - U.force) <= 0.01 ) {
 return 0.75;
}
return 0;
```

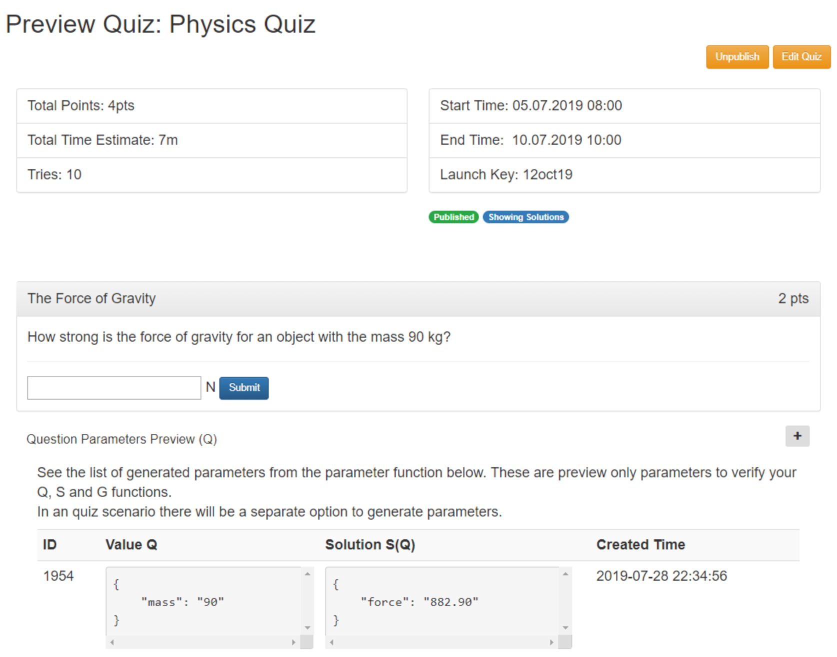
The grading function can return floating-point values between 1 and 0 where 1 is the highest and 0 the lowest possible grade. In this example the teacher defines an acceptable deviation from the correct result.

3.5 Example of the solution guide SG(Q)

```
<script>
 var force = ({{mass}} * 9.81).toFixed(2);
</script>
The Force of Gravity in Newtons for an Object can be
calculated with the Formula: Fg = m * g.<br/><br/>
The correct solution for this example would be:
<b>
  {{mass}} * 9.81 =
  <script>document.write(force)</script>
</b> Newtons!
```

The solution guide is shown after a completed quiz attempt if the teacher has enabled the solutions option in the quiz creation process. It can contain HTML and JavaScript.

Figure 2 shows a quiz preview as seen by a teacher. The question title is printed at the top of the page and in the header section of the question panel. Additionally, at the top above the question panel, resides a quick info panel showing the most important information of the quiz. This includes the total points (sum of points of all questions), the total estimated time (sum of question time estimates), the number of retries (after all retries are used the quiz is locked for the learner), the start and end time of the quiz and the launch key. The launch key is important because it must be referenced as custom parameter in the LTI consumer to identify which quiz should be started with the external tool activity.



Preview Quiz: Physics Quiz

Total Points: 4pts	Start Time: 05.07.2019 08:00
Total Time Estimate: 7m	End Time: 10.07.2019 10:00
Tries: 10	Launch Key: 12oct19

The Force of Gravity 2 pts

How strong is the force of gravity for an object with the mass 90 kg?

Question Parameters Preview (Q)

See the list of generated parameters from the parameter function below. These are preview only parameters to verify your Q, S and G functions.
In an quiz scenario there will be a separate option to generate parameters.

ID	Value Q	Solution S(Q)	Created Time
1954	`{ "mass": "90" }`	`{ "force": "882.90" }`	2019-07-28 22:34:56

Fig.2. First question of an example quiz preview

When a quiz has ended and student answers are present, teachers can calculate the grades for every student by selecting the quiz and clicking a button in the grading interface. This will calculate all grades for the student answers and redirect the teacher to a page where he or she can compare all student answers to the correct answers. If necessary it is possible to override the calculated question grades manually. The next step in the grading process is to let the system calculate the final quiz grades. They are the average of the question grades over the number of questions in the quiz. Finally, the quiz grades can be sent to the LTI consumer. In this context two attributes of the LTI launch request are important and have to be remembered beforehand somewhere in the quiz process. These two parameters are called the "lis_outcome_service_url" and "lis_result_sourcedid" and are a part of the LTI launch request. The outcome service URL defines the endpoint to which grades can be posted. The sourced ID parameter contains information about the user and the resource link. This is important because the LTI consumer stores the grades per resource link (this is an ID that identifies the external tool activity in the consumer) and per user. In the quiz application these attributes are stored whenever a student attempts a quiz. This causes the grades of every student to be displayed grouped by the quiz in the LTI consumer and is optimal for performing multiple tests over the course of a semester. The partial results can be aggregated by Moodle to a final semester grade.

4 Discussion

The quiz application tries to offer the freedom of JavaScript and HTML in a quiz builder application. This gives the creators of the quizzes much freedom but also creates a more room for errors. It is arguably more difficult to code JavaScript and HTML code in an HTML text area field than in an editor. Debugging also is harder as the page that executes the provided JavaScript code contains a JavaScript code that is responsible for the generation of parameter and solution instances. The application prints all catchable errors from the provided JavaScript code on the page but unfortunately, syntax errors are not catchable but very common.

The grading interface calculates the grades for every quiz attempt of every student. It does that to ease the possible future implementation of different quiz modes. At this moment, the grading only takes the last closed quiz attempt into account. Other attempts of the students are not shown in the graphical user interface. Moodle for example, also offers options for quiz grading such as "Highest Grade".

The user input area also has limitations due to the differences in the structure of HTML input fields of different types. A checkbox field value cannot be set in the same way as the value of a text field. This is also true for other input types such as radio buttons or input selections. It would be necessary to parse the input type of every input to prepopulate the field values for given answers when a quiz attempt is resumed. The application therefore only supports the automatic population of given answers in unfinished and reloaded quiz attempts for the HTML input types text and hidden.

Moodle provides more than only one mode for the grading of its quizzes. The implementation of the grading interface computes the grades for all quiz attempts of all students, but only shows the last closed attempt to the teachers. This makes possible future implementations for different modes such as "highest grade" easier as the grade data is already present in the system but at this moment is data that is not used.

Open standards, such as the LTI protocol are very important and a great tool for providing reusable and more importantly, shareable learning resources. While implementing the features of the quiz, it became clear that when retries are involved, the flow of procedures and resulting states of a quiz were more complex than initially thought. Especially retries combined with randomized question parameters for the questions demanded a more complex database design as defined at the beginning of the implementation, as questions about the persistent storage of results arose. This is problematic when a question at the time when a quiz is taken, is modified. This could potentially invalidate the existing results. Therefore, it was necessary to duplicate the information and store all question information along with each quiz attempt by every user, which was not intended at the beginning of the development.

Another insight is that to provide a meaningful online learning experience for both, teachers and students, the quality of the application should be high. Modern leading systems in the field set very high standards and the perceived quality of the application that delivers educational resources matters [7] in the e-Learning process, which makes any implementation more demanding.

5 Conclusion

The implementation of the quiz application was successful. It is possible to create quizzes that can be launched from Moodle. The quizzes consist of questions that can contain HTML and JavaScript code as well as random variables. This will be a sufficient solution to build quizzes containing random variables that are assigned by arbitrary mathematical functions. There are still many ideas for extensions of the application, some of which are flexible JavaScript includes for questions and the implementation of learning analytics mechanisms. The first will further improve the question building capabilities of the system by enabling the teachers to use any JavaScript library they desire. The latter would enhance the teachers' ability to adapt their teaching materials by analyzing the questions by for example letting the teachers know how long the students need to complete a given question.

6 References

[1] Harandi, Safiyeh Rajaee (2015). "Effects of e-learning on Students' Motivation." In: Procedia - Social and Behavioral Sciences 181, pp.423–430. ISSN:18770428. DOI: https://doi.org/10.1016/j.sbspro.2015.04.905

[2] Ebner, Martin (2019). "Virtuelle Lernorte: eine Übersicht." In: URL: https://www.researchgate.net/publication/331981005_Virtuelle_Lernorte_eine_Ubersicht

[3] Zimmerman, Barry J. (1990). "Self-Regulated Learning and Academic Achievement: An Overview." In: Educational Psychologist 25.1, pp.3–17. ISSN: 15326985. https://doi.org/10.1207/s15326985ep2501_2

[4] Barbosa, Hector and Francisco Garcia (2005). "Importance of online assessment in the E-learning process." In: ITHET2005:6th International Conference on Information Technology Based Higher Education and Training, 2005. Vol. 2005. ISBN: 0780391411. https://doi.org/10.1109/ithet.2005.1560287

[5] Ćukušić, Maja, Željko Garača, Mario Jadrić (2014). "Online self-assessment and students' success in higher education institutions.", In: Computers and Education 72, pp.100-109. ISSN: 03601315. DOI: https://doi.org/10.1016/j.compedu.2013.10.018

[6] Vickers, Stephen P. and Simon Booth (2014). "Learning Tools Interoperability® (LTI®): A Best Practice Guide." URL: http://ltiapps.net/guide/LTI_Best_Practice.pdf

[7] Alsabawy, Ahmed Younis, Aileen Cater-Steel, and Jeffrey Soar (2016). "Determinants of perceived usefulness of e-learning systems." In: Computersin Human Behavior 64, pp.843–858. ISSN: 07475632. DOI: https://doi.org/10.1016/j.chb.2016.07.065

7 Authors

Jakob Schweighofer is a student at the University of Technology of Graz and has written his master's thesis about the development of LTI compatible learning applications. jak.schweighofer@gmail.com

Behnam Taraghi is a PhD Candidate in Technology Enhanced Learning (TEL) at Institute of Interactive Systems and Data Science. His research focus is Learning analytics and intelligent Tutoring Systems in TEL. b.taraghi@tugraz.at

Martin Ebner is with the Department of Educational Technology at the Graz University of Technology, Graz, Austria. As head of the Department, he is responsible for all university wide e-learning activities. He holds an Assoc. Prof. on media informatics and works at the Institute of Interactive Systems and Data Science as a senior researcher. For publications as well as further research activities, please visit: http://martinebner.at. Email id: martin.ebner@tugraz.at

Article submitted 2019-08-09. Resubmitted 2019-09-19. Final acceptance 2019-09-21. Final version published as submitted by the authors.