An Algorithm for Assessment of Students Using Gamification

D. Keremedchiev**, D. Borissova*** and G. Tuparov*

* New Bulgarian University/Department of Informatics, Sofia, Bulgaria
** Bulgarian Academy of Sciences/Institute of Information and Communication Technologies, Sofia, Bulgaria
*** University of Library Studies and Information Technologies, Sofia, Bulgaria
delyan.keremedchiev@gmail.com

Abstract - The use of game design and elements in non-gaming context known as Gamification presents one of the newest ways to improve traditional assessment. There are various tools for self and final assessment of students in the modern e-learning systems. In this paper, we propose a grading algorithm composed of two parts: self-assessment and final exam. In the self-assessment phase, students must solve tests and fill a bar (cover scores) according to their results. The tests are generated using the “Intelligent system for generation and evaluation of e-learning tests using integer programming”. This approach allows students to choose between different levels of question scores and different number of questions in the generated test. The required score for filling the bar and successful self-assessment can be obtained either with a small number of questions with higher score or a large number with a lower score. Only after passing successfully all self-assessment tests students are allowed to do final exam.

Keywords - component; formatting; style; styling; insert (key words)

I. INTRODUCTION

With the development of information and communication technologies, ease of internet access and the rise of internet-based applications many traditional activities go online. This refers to the learning processes in the modern e-learning systems as well. The assessment is a common way to give a feedback to students about the acquired knowledge. Assessments can be done in many ways. Various tools for self and final assessment exist. A popular method of assessment is the test. Tests can consist of different test elements depending on the complexity and purpose of the test. Many of those elements can be automatically checked and evaluated. Creating a test using those elements will give the students the opportunity to get an instant feedback about the acquired result. There are researches which conclude that the students who use tests for self-assessment have better performance compared with those who do not use them [1]. The result is 85.7% vs. 77.8% for the students who have used the tests. The use of game design and elements in non-gaming context, formally named gamification, as well as serious games has become very popular in the last few years. Combining these approaches can lead to qualitative leap in the efficiency of these processes. In recent studies like [2] and [3] the results from peer and self-assessment in the Moodle e-learning environment prove that gamification in e-learning provides better learning experience and better learning environment. Gamification can also be applied to assessment as a tool for instant feedback. The game-based quizzes have recently become more interesting and popular. Modern web browsers supporting HTML5, CSS3 and JavaScript can be used as an environment for e-learning tasks, assessments, quizzes and various interactive elements including gamification. The browsers with such advantages are available on all modern computer and mobile operating systems.

In this paper we propose a grading algorithm for student assessment, which combines several gaming elements. The algorithm includes "Intelligent system for generation and evaluation of e-learning tests using integer programming" [4] as a main part in the process of test assembly. This approach allows the students to control the level of complexity of the tests, the number of questions and question weights. The use of game elements will make the assessment more attractive. Ranking elements will provoke the students to join the race and work harder to obtain higher results. The algorithm itself is implemented in a web-based system for student SQL knowledge and skill assessment.

II. RELATED WORKS

An increased interest in gamification research in scientific databases has been noticed in the last few years. A literature searches in SCOPUS database which indexes relevant databases such as ACM, IEEE, Springer, etc. with the query: TITLE-ABS-KEY for the word [gamif*] is made in [5]. Some results of this paper analysis are shown in Table I.

Apart from empirical researches different algorithms for gamification exist in education. The authors of [6], [7], [8] propose different combinations of technologies and gamification elements in e-learning and report successful impact on students. According to [9] the gamification techniques considered the most potentially effective in an educational setting are: “badges (appealing to masterminds), progress bars and leaderboards (appealing to conquerors), a storyline and a visual (appealing to the seekers)” [9].

The study is funded by Bulgarian National Scientific Research Fund, Contract DN-05/10, 2016

MIPRO 2020/CE
With the usage of a combination of progress bars and leaderboards the impact of the proposed algorithm will grow. Progress bars allow the students to keep track of their points and to track and display the overall goal progression. They are used as a display mechanism to motivate students who are close to achieving the goals. In other implementations of gamificated educational systems students are usually graded with badges for reaching a certain level. Badges indicate that the covered score is in a certain known scope. Using scores will add more precise tracking of the students’ knowledge. Progress bars indicate students’ overall progress. Leaderboards allow students to view their achievement compared to others. Due to privacy needs there must be an option to make the leaderboard anonymous. The names of other participants must be hidden in these cases and the only information shown must be the summary for the ranking.

There are automated grading tools for SQL knowledge and skills like SQL-Lab [10], aSQLg [11] and MySQL Sandbox for Processing SQL Statements [12]. These tools give an opportunity to test students for their SQL skills. Queries are executed and the results of the execution are compared to the correct answer. Select queries are the most common queries used. Queries for modifying the database, tables, indexes and data are more complicated to manage. Allowing students to write and test queries for modifying the structure and data can lead to security vulnerability, cheating options and decrease of the system stability, even system crash. Using preconditioned questions will reduce the number of topics that can be covered by the test. Tools like “Web-based system for automatic evaluation of SQL queries” [13] can generate a question bank in an exchangeable format for use in other learning and assessment systems. These tools give an opportunity to combine questions and database statements. With the use of these combinations queries can be run against different database statements and the correct answer depends on the current state.

### III. Grading Algorithm

An original algorithm for the assessment of students using gamification is presented in Figure 1. The algorithm starts with a test on the very first topic. The next and all the other topics are hidden/inaccessible until the minimum test score requirement for the previous topic is covered. The test is generated using the “Intelligent system for generation and evaluation of e-learning tests using integer programming” [4]. Students can choose between different levels of complexity for the test and different number of questions with different weights. The test can be solved as

![Figure 1. Algorithm for the assessment of students using gamification](image)
many times as necessary. At every attempt, students can change their preferences for the test generation options. When the minimal score set for the test is covered, the next test opens and is available for solving. The final result for every test is the best result obtained. After successful solving all the tests, students’ total score is calculated as a sum of the results obtained from all the tests. Students can do the final exam when the total score, calculated as the sum of all scores is over the minimum set.

For the implementation of the algorithm, the following gamification and graphical elements are chosen:

- Fill the progress bar – the score of every test is presented as an empty progress bar. With the solving of the test students obtain scores. The obtained score is visualized as filling the progress bar. Once the student covers the minimal amount of points needed for the test completion, the progress bar is thoroughly full and the next stage is accessible. After all stages are completed an additional progress bar shows up. This additional progress bar contains the sum of all the scores obtained by the student for the different topics during the tests minus the points already used for progress bar filling.
- Leaderboard and ranking – students can see their position in the ranking compared to the results of the other students. This leaderboard can help the students see their rank in the community. The ranking includes position, list of participants and their score. Ranking can be made anonymous due to privacy requirements and in this case only the student’s own position, score and points insufficient for the student to reach the leading position will be shown. Leaderboard and ranking is created and updated for every single score and for the total score obtained.

IV. IMPLEMENTATION

A classical web-based system implementing algorithm is created. “Assessing SQL knowledge and skills” is chosen as a main topic. Questions are created using “Web-based system for automatic evaluation of SQL queries” [13]. The topics for the tests are:

- T1: Single table query;
- T2: Multiple table query (JOINS);
- T3: Filtering data;
- T4: Grouping and aggregating data;
- T5: Functions and stored procedures;
- T6: Sub-queries;

Every question has score points set by the teacher. The test scores are set in the range [0-200]. Test generation is made using “Intelligent system for generation and evaluation of e-learning tests using integer programming” [4]. The minimal score for successful test achievement is 50% or 100 points. In the test score only the correct answers are calculated. The test score must be adjusted into marks. The marks used for the course grade are set in the range Poor (2) - Excellent (6). Scores less than 50% are considered as failure and the mark is Poor (2). For successful are counted all results with 100 or more points. There are no limitations for the complexity ranges in the test generation algorithm and the complexity levels can be as many as needed. A range of points and their adjustment into marks is shown in Table II:

The use of “Intelligent system for generation and evaluation of e-learning tests using integer programming” algorithm for test generation gives the students the opportunity to choose among different levels of complexity of the test. The interface of a test generation system can be seen on Figure 2.

The chosen intelligent system for the test generation emphasizes on the optimal selection of questions for the generation of tests with different levels of complexity. As described in [4] a predefined set of questions for the test generation is needed. Those questions must have different weights and points for a proper test generation. This optimization problem is solved with the use of integer programming. When the student chooses a L1 test the upper boundary of the generated test will be set to 200 and the lower boundary will be set to 175. Changing the number of questions and their weight preference also influences the sum of maximum points which can be obtained from the test. The authors of the algorithm have noticed that choosing the “more questions but light-weighted” will result in a generated test with a sum of points closer to the lower boundary. While choosing the “less but heavily-weighted questions” option results in a generated test with a sum of points closer to the upper boundary. In Table III the approximate scores of the generated tests are shown:

![Interface for test generator](image)

TABLE II. POINT TO MARK ADJUSTMENT

<table>
<thead>
<tr>
<th>Complexity level</th>
<th>Mark</th>
<th>Obtained points</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>Excellent (6)</td>
<td>175-200</td>
</tr>
<tr>
<td>L2</td>
<td>Very Good (5)</td>
<td>150-174</td>
</tr>
<tr>
<td>L3</td>
<td>Good (4)</td>
<td>125-149</td>
</tr>
<tr>
<td>L4</td>
<td>Satisfactory (3)</td>
<td>100-124</td>
</tr>
<tr>
<td></td>
<td>Poor (2)</td>
<td>0-99</td>
</tr>
</tbody>
</table>
TABLE III. GENERATED TEST SCORES

<table>
<thead>
<tr>
<th></th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>192</td>
<td>198</td>
<td>195</td>
<td>195</td>
<td>196</td>
<td>197</td>
</tr>
<tr>
<td>L1</td>
<td>178</td>
<td>177</td>
<td>180</td>
<td>178</td>
<td>177</td>
<td>179</td>
</tr>
<tr>
<td>L2</td>
<td>168</td>
<td>170</td>
<td>170</td>
<td>171</td>
<td>173</td>
<td>172</td>
</tr>
<tr>
<td>L2</td>
<td>155</td>
<td>152</td>
<td>154</td>
<td>155</td>
<td>151</td>
<td>152</td>
</tr>
<tr>
<td>L3</td>
<td>145</td>
<td>145</td>
<td>148</td>
<td>146</td>
<td>149</td>
<td>144</td>
</tr>
<tr>
<td>L3</td>
<td>130</td>
<td>128</td>
<td>125</td>
<td>128</td>
<td>127</td>
<td>127</td>
</tr>
<tr>
<td>L4</td>
<td>120</td>
<td>119</td>
<td>122</td>
<td>119</td>
<td>121</td>
<td>124</td>
</tr>
<tr>
<td>L4</td>
<td>105</td>
<td>103</td>
<td>106</td>
<td>106</td>
<td>101</td>
<td>104</td>
</tr>
</tbody>
</table>

As can be seen in Table III the approximated results have points within the boundaries but never equal to them. Moreover, a different test generation can result in different points for the test. This randomness will lead to more attempts to get the best result possible. After a test is completed, an animation which fills the progress bar with the obtained result is started. The progress bar width is set to the minimum of 100 points for opening the next topic required. The result can be seen as shown on Figure 3:

After the successful completion of all stages the additional progress bar shows up. This progress bar is larger than the previous and requires more points. Initially this progress bar is filled up with the obtained points minus the points already used. The filled sections are the same color as the colors of the completed tests. Furthermore, there are labels showing which test points are used to fill the section. This solution helps the student to easily find out which topics have more and which have less points. When the whole additional progress bar is filled the student can have access to the final exam.

Using a progress bar composed by parts previously completed with the sum of the results obtained is more sensitive and robust compared to badge-based algorithms, where the requirements are such as “Obtain at least 3 bronze and 3 silver badges”. Badge-based algorithms set the lower boundary and do not estimate results higher than this boundary. In score-based algorithms all acquired points are calculated in the final result. These algorithms will motivate students who have not completed the assessment bars, but their total result is close to the minimum requirements. Moreover, using a requirement such as “Total score of XXX points” is very clear. Translated to badges it may be presented as: “3 silver and 3 bronze badges or 1 gold, 1 silver and 4 bronze badges or 1 platinum and 5 bronze badges, etc.” Figure 4 shows the partially filled final progress bar:

The detailed leaderboard per test is just on a click and it shows the student’s result compared to all obtained results for the test. Figure 5 shows the leaderboard interface:

The anonymous ranking interface is seen on Figure 6.

The front-end of the proposed system and all visual elements such as progress bars, rankings and leaderboards are created using Bootstrap. Bootstrap is a front-end framework initially developed by Twitter and one the most popular open source frameworks in the world [14]. In Bootstrap 4 there are important features such as responsive functionality and mobile first approach. Using flexbox as a main design technique is modern browser oriented. Bootstrap has a predefined set of icons, badges, crates, accordions and many other graphical elements ready to use. Using Bootstrap will lead to rapid web development and faster creation of ready-to-use front-end interfaces.

Every completed test can be started again either with a different set of options or with the same one. The mixing of different levels of complexity and the number of questions according to their weight can produce many different tests. It can be seen on Table III that the highest scores for a test can be obtained with the “More questions but heavily-weighted” option. Another feature of the test generation algorithm is the randomness provided. This randomness can cause variations in the maximum test result. In such cases full-score solved tests by different students can obtain different results. This may produce additional activity from the best students and they will do the tests again and again until they obtain the maximum...
possible result. This technique and user behavior can be seen in social gaming platforms, where users do the same level many times to beat their friends high-score and to lead the ranking [15].

V. CONCLUSION AND FUTURE WORK

VI. IMPLEMENTATION

The current article describes an algorithm for the assessment of students using gamification. The selection of a different level of complexity and a different number of questions with different question weights by the students gives them the feeling of full control of the process. Using “Intelligent system for generation and evaluation of e-learning tests using integer programming” algorithm for question selection in the test assembly process adds an additional randomizing element to the possible maximum test score obtainable. This will result in additional activities by the best students to obtain the highest possible result. Using gamification elements and animations for the interface makes the implementation of the algorithm more sophisticated and with better appearance.

The proposed algorithm can be easily implemented as a classical web-based application as seen in IV. The algorithm is flexible and can be set with a wide range of options. A scheme for points-to-mark adjustment according to test complexity levels can be included during the setup. The modern attention-grabbing techniques and interactive interfaces add additional impact to the assessment process. The algorithm is based on scores and with this approach every single point matters. Progress bars will help the students to track closer their knowledge and skills.

It will be interesting to compare the results and students’ behavior obtained using this algorithm with the results obtained by badge-based algorithms. Data such as number of tries, combination of selected options, acquired points for the initial progress bars, tests that are re-taken after the total progress bar shows up and number of tries after leaderboard checks can be collected and examined. These student activities can be recorded to xAPI [16] server for further analysis.

Creating a mathematical model for completing the tasks with a minimal number of tries and solving the model can also be an interesting problem. The maximum possible test result variations and the acquired points according to the selected test generation options and student knowledge may vary in a larger range.

ACKNOWLEDGMENT

The study is funded by Bulgarian National Scientific Research Fund, Contract DN-05/10, 2016.

REFERENCES

[16] Rustici Software, xAPI Overview, https://xapi.com/overview/, last visit on 01.02.2020