EMITTER International Journal of Engineering Technology
Vol. 10, No. 1, June 2022, pp. 14~30
DOI : 10.24003/emitter.v10i1.667

An Improvement of Computer Based Test System Based on TCExam for Usage with A Large Number of Concurrent Users

Yunarso Anang[1], Rahadi Jalu Yoga Utama[1], Masakazu Takahashi[2], Yoshimichi Watanabe[2]

[1]Department of Statistical Computing, Politeknik Statistika STIS, Jakarta, Indonesia
[2]Department of Computer Science and Engineering, University of Yamanashi, Yamanashi, Japan
Correspondence Author : anang@stis.ac.id

Received January 18, 2022; Revised February 20, 2022; Accepted March 26, 2022

Abstract

Computer-based test or assessment has been used widely, especially in the current COVID-19 pandemic, where many schools are conducting distance learning as well as distance examination. The need for a computer or software system to support education is inevitable. A range of solutions, from the free/open source software systems to the paid/proprietary ones have been publicly available. Still, an organization with limited resources prefers to find free or low-budget, while yet demanding reliable solutions. We have reported the use of the computer-based test in a new student recruitment test which is held country-wide. We developed the system based on TCExam, a free and open source computer-based test software, and successfully fulfilled the requirements, but with some tweaks. We found that the TCExam has a performance degradation when used by a large number of examinees concurrently, especially during specific phases during the test. This paper reports the result of our investigation to address the problem and suggests some modifications to the base codes as well as a recommendation of the hardware configuration. We evaluated the modified system in a simulated environment. We successfully achieved up to 56% performance gain using the modified system.

Keywords: computer-based test, TCExam, php, sql

1. INTRODUCTION

In 1845, Horace Mann, an American educational reformer, introduced his vision for reforming American education by suggesting the Boston Public School Committee to conduct a common written exam instead of oral exams for their children [1]. Using a common exam, he hoped that all children could have equal opportunities to achieve a good result in exams. Such standardized tests became central to, not only of how our educational system, but also in how general assessment work afterward.

These days, standardized tests conducted by utilizing computer systems are widely used, to provide a more efficient and transparent process

of testing [2]. The system is well known as the computer-based test (CBT). Compared to the paper-and-pencil test (PPT), there are some distinct benefits from CBT. Those benefits include cost-savings on printing and shipping of the paper materials and the increase of the accuracy of data collection. Even for tests with multiple-choice answers where optical mark recognition (OMR) can be used to automate the data collection, the chance of error in recognizing the test answer still exists. On the other hand, in CBT, the data are collected directly to the computer system and the process of scoring can be simplified. The benefits of CBT can also be seen as the benefits to the examinees where many CBTs offer immediate test scoring. However, in the real world, CBT is not necessarily better than PPT. The need for the appropriate development of the CBT system should not be undertaken lightly.

Anang et al. reported a case study of the implementation of a CBT system in a college's new student recruitment process [3]. They described how the system was developed introducing software engineering practices. The system was developed based on TCExam, an open source CBT software. TCExam is a free web-based and open source software (FOSS) which has the capability to administer CBT [4]. In addition to the ISO/IEC 9126, a standard for "Information Technology—Software Quality Characteristics and Sub-Characteristics" [5], which has been replaced by ISO/IEC 25010:2011 [6], TCExam also introduces other specific quality features. However, from Anang et al.'s report, it has a performance degradation in specific phases during the test when the number of concurrent examinees exceeds a specific number. The problem has successfully been troubleshooted by splitting the server or by differentiating the schedule of the test, but the main cause of the problem remains untouched and needs to be addressed.

This paper reports about our investigation to address the performance degradation problem that exists in TCExam when utilized in a large scale of test involving a large number of concurrent users. We identified three major phases which cause an impact on the system performance. To address those issues, we have modified the original code which is mainly written in the PHP programming language. We evaluated the result of the modifications by conducting an experiment in a simulated test enviroment of a large-scale concurrent use using parallel processing on LINUX machines as the user client's PC. We compared several configurations and show and describe the results using plots.

The rest of this paper is organized as follows. Section 2 describes works related to CBT and the use and application of TCExam in the CBT system. Section 3 describes the originality of the study. Section 4 describes the outline of our work in investigating the performance degradation problem in TCExam. Section 5 provides the experiment and the result as well as the evaluation. Finally, we conclude with remarks in Section 6.

2. RELATED WORKS

Studies in CBT discussed mostly the effectiveness of using the computerized test compared to the non-computerized one. There is a comprehensive literature study of CBT conducted by Russel et al. that reports the result of their study on the examination of the potential benefits of converting PPT to CBT [7], studies on the examination of the validity of CBT and its effects on test performance and motivation [8][9][10], and a study on the effect of user interface design on the result [11].

As the use of computers to conduct tests is becoming more prevalent in the educational assessment domain, to establish a valid and reliable CBT, the International Test Commission (ITC) Guidelines on Computer-Based Testing and Internet Delivering Testing [12] (hereinafter called ITC Guidelines) stated that equivalent test scores should be established for the conventional PPT and its replacement, the modern CBT. The guidelines also mentioned explicitly that when designing a CBT version of a non-computerized test, equivalent control should be provided to the examinee such as the ability to skip or review test items as on the non-computerized one. Considering the result of previously reviewed studies as well as the guidelines, the developers should give a higher priority of concern to the design of the user interface in order to achieve the same scoring results of using the CBT compared to the PPT.

Thurlow et al. in their synthesis report [13] stated nine considerations for developing and implementing CBT such as incorporating inputs from various stakeholders, considering the system as a whole from the computer infrastructure to test room and personnel, the need to elicit the specific accessibility features, conduct field test, and develop training for administrative personnel and examinees. Those considerations depict all considerations used in practicing software engineering as described in the Guide to the Software Engineering Book of Knowledge (SWEBOK) [14], where ISO/IEC/IEEE Systems and Software Engineering Vocabulary (SEVOCAB) [15] defines software engineering as a "systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software" and an "application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software". As SEVOCAB defines a computer-based software system as "a software system running on a computer", so a CBT system can be treated as a software system, thus the use of software engineering approach is appropriate to develop the system. Hereafter, we describe works related to the development of the system from the software engineering point of view.

He and Tymms described the development of a software system for CBT [16]. After reviewing several existing commercial off-the-shelf (COTS) CBT systems, for the reasons of limited budget and expert knowledge, they decided to develop the system in-house. Their requirements are to develop a

system which is both easy and economic use and yet still provide the necessary functions. While the decision making was relevant to the economic aspect, instead of developing in-house, there is still a solution to use an OSS as we also reviewed when we were in the process of deciding the candidate solutions. Furthermore, the paper did not state whether the cost spent for developing the system was less or in par with the COTS system. The rest of the paper provides a brief description about the system specifications, design, and implementation but we could not find the important part as of how they evaluate the system before they use it and how was the implementation (running in a production environment).

TCExam is an open-source software (FOSS) web-based computer-based assessment (CBA) system that enables educators and trainers to author, schedule, deliver, and report on surveys, quizzes, tests, and exams [4]. In addition to the aforementioned ISO/ICE 9126 as well as its updated ISO/ICE 2501:2011 quality model and general CBA features, TCExam introduces other specific quality features such as platform-independent, no expensive hardware requirements, internationalization, accessibility and usability, data export, and import, rich content, and unique test per user. After its release, it has been used in several applications.

Shah et al. developed WriteSim TCExam, an open-source, web-based, textual simulation environment for teaching effective writing techniques to novice researchers [17]. Among other open-source applications, TCExam has been chosen to serve their needs for its simple, intuitive interface and its open-source architecture. TCExam has been used with some modifications, such as: (1) The system would give immediate feedback to the end user upon answering the question; (2) Blog and forums would be enabling mentoring relationship among participants and between participants and the administrator; and (3) Persistent bugs fix and user-interface modification for better user-friendly. It has been used to train 25 novice researchers. Ismail et al. developed a web-based homework system that can be embedded in teaching and learning by school teachers [18]. TCExam has been adopted in the system because of its cost-effectiveness, also as an alternative to the existing pen and paper-based homework. The system received a very high positive perception from the users even without any modification to the base system. There was no information about how many users are using the system concurrently. Ambiyar et al. used TCExam to study the test performance of CBT compared to paper-based test (PBT) [19]. From the questionnaires given to the respondents, the results showed that the CBT received a better response compared to the PBT. The paper did not discuss the technical issue of the TCExam.

The FOSS we have chosen is a web-based system. As we implement it as a whole system, we need to evaluate the running system in the same or similar condition of the production environment, where it will be used concurrently and simultaneously by many examinees. Shaw has conducted a case study of an online learning application to run performance testing on a

web application [20]. The author used COTS test tool: LoadRunner1, to measure the load on the system. The author found that late use of performance testing does not help to scale the system but did identify real problems and gave an indication of where the main problem lay. By knowing the location of the problem, the developer could find a solution to countermeasure the problem. The author also stated that the use of particular software architecture or particular hardware specification could not guarantee adequate performance. The important thing is how to understand what is impacting the performance and considering the development, usage and environment approaches to find the solution. Other studies also reported using load testing in web applications [21], [22]. The use of automated tools helps to reduce the resources for conducting the test and to increase the accuracy of the results when simulating the actual use of the system. Not only COTS but FOSS test tools could also be used. In our implementation, conducting a performance test did help us to indicate problems in the actual use where concurrent and simultaneous examinees use the system in the same period of test with the same start and end times.

And lastly, Hardiansyah et al., in their paper, proposed a new approach to control the Internet connection based on idle time using user behavior pattern analysis [23]. To see if the system can recognize patterns, they conducted experiments in two scenarios, one is aimed to determine the performance, and the other is aimed to determine the effectiveness if the method. We adopt their idea in designing our own experiment.

3. ORIGINALITY

As described above, as a FOSS, the freely-useable TCExam has the potential to be a good foundation of a computer-based test. However, based on our previous case study, it has a performance degradation problem in a case where there are a large number of concurrent users accessing the system. From the documentation of TCExam[1] and from its Github[2], there is no information regarding the condition or limitation of use. And, at the time this paper was written, there is no study indicating or addressing the problem. This paper describes our investigation and its result in addressing the problem. Our methodology is by first understanding the design and the architecture of the TCExam's system. And then by building a system for simulating the test with a large number of multiple concurrent users, we indicate the place in the system which needs to be addressed. And finally, we modify the system and simulate the test again to verify that the modification takes effect on the performance.

4. SYSTEM DESIGN

In this section, first, we describe how we implement it in our CBT system and summarize its issue. Second, we describe the architecture and the

[1] https://tcexam.org/docs/
[2] https://github.com/tecnickcom/tcexam

file structure of the base TCExam. And finally, we describe the outline of the investigation process and problem-solving we have conducted. Our system is based on TCExam version 14.0.3. The result and the evaluation will be described in the next section.

4.1 The Implementation of CBT Based on TCExam and Its Issue

The implementation of CBT in our use case is to conduct an academic test as part of the recruitment for the new student to our institution. In our case, the applicants from all around Indonesia take the test on the given schedule at the same time. In 2018 and 2019, there around 15,000 applicants are taking the test. As reported here [3], we found that there is a performance degradation problem when the number of concurrent users exceeds a specific number. The number depends on the specification of the server, but the maximum number of users for optimal use is between 100 and 200 for each server which is placed locally in each location of test around the country. In 2020 and 2021, due to the pandemic, we decided to conduct the test in full online environment, where all applicants access the same CBT server using Internet. Although the number of applicants is reduced to around 1,600 total due to the shift of the academic test to the second stage, the problem still needs to be addressed. In 2021, the number of concurrent users conducting the test is around 700 in one section of test.

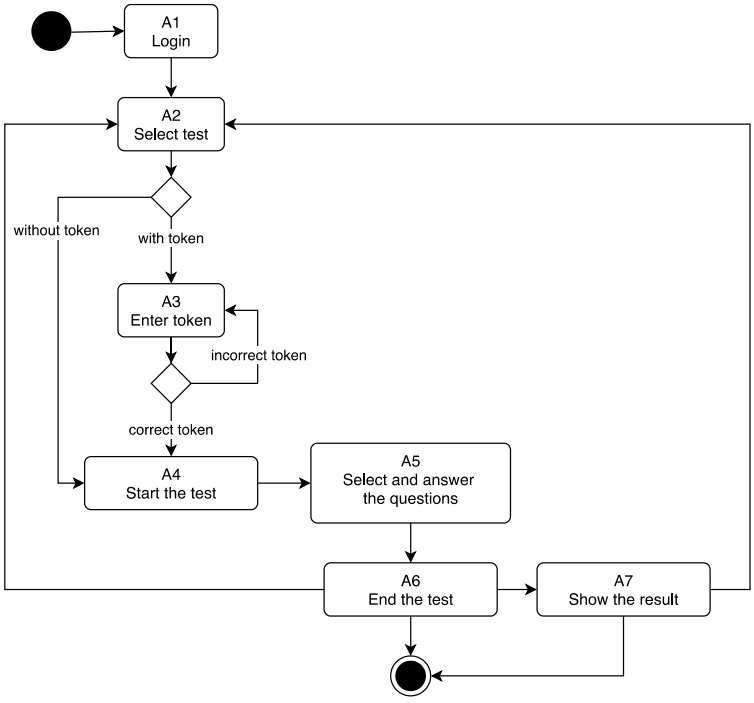

Figure 1. The flow of an applicant in TCExam based CBT

```
This algorithm shows how the TCExam processes the request from the
applicant.

In the landingPage controller (A2):
  Show all the test assigned to the applicant
  Waiting for the applicant to select a test
  If a test selected, checked for the token (tokenEntry controller)

In the tokenEntry controller (A3):
  Show the entry form for the token
  Check whether the entered token correct
    If yes, start the test (testPage controller)
  Back to the entry form

In the testPage controller (A4):
  Check the last viewed question
    If it is the first time, show the first question
    Else show the last viewed question
  Go to the questionsPage controller

In the questionsPage controller (A5):
  Loop:
    Wait for the action which can be either answer or go to
      the other question
    Check if the time is up
      If the time is up, go out the loop
    Check if the applicant terminate the test
      If the applicant end the test, go out the loop
  Go to the endPage controller

In the endPage controller (A6):
  Show an option to the applicant to show the result
    If the application select to show the result, go to resultPage
  End!

In the resultPage controller (A7):
  Show the result
  Back to the endPage controller

In the main controller (A1):
  Received the request from an applicant
  Check if the applicant have already login
    If yes, go the landing page (landingPage controller)
  Loop:
    Show the login form
    Check the login information
      If correct, go to the landingPage controller
```

Figure 2. The pseudo algorithm of the entire flow in TCExam based CBT

Figure 1 shows the flow of test for a particular applicant, with its pseudo algorithm shown in Figure 2. From our previous study [3], we found that there is a performance impact in activities A1, A3, A4, and A7, when a large number of users do each of those activities concurrently. As for the A4, where a unique combination of questions is generated on the fly for each applicant, we have solved the problem by generating the set of questions for each applicant prior to the test execution. However, it would sacrifice the guarantee that the applicants would not be knowing the questions until the time they started the test. And yet, the cause of the problem still needs to be addressed.

4.2 The Architecture of the TCExam

Figure 3 shows the architecture and the file structure of the TCExam. The figure has been remade for better visualization from the original shown in Asuni's report [4]. TCExam adopts a common three-tier architecture. As a web-based application, TCExam requires an application server such as the Apache HTTP server. In behind, it uses a RDBMS (relational database management system) such as PostgreSQL or MySQL or other well-known RDBMSs as the database server. The application server and the database server can be operated in one or separated hosts, whether physically or virtually. And in front, the end-user uses the system by utilizing a web browser. TCExam uses standard HTML, CSS, and minimal client-script for better browser compatibility.

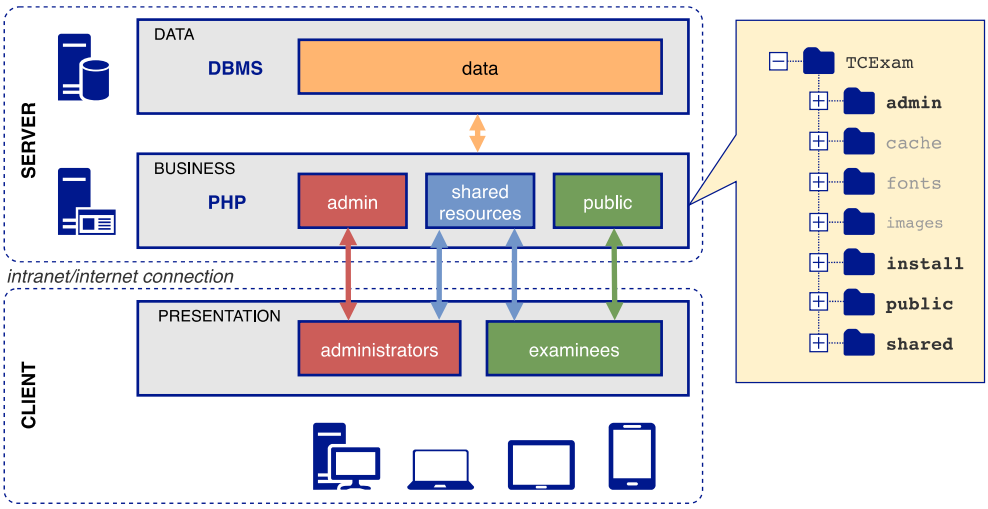

Figure 3. The architecture and file structure of TCExam

As for the file structure, there are several files stored in different folders as shown in Figure 3. The system is decomposed into two main functionalities: `admin` for administration and `public` for common users, each has its own user privilege management. The administration and the public areas are physically separated on the file system to improve security, while the shared codebase remains in the `shared` folder including those stored in `cache`, `fonts`, and `images` folders.

TCExam is written mainly in the PHP programming language. The main codes are placeholder for administration, public, and shared, each contains folders as shown in Figure 4. All main PHP files that are accessible by the end user are stored in `admin` and `public` folders, specifically within `code` folder in each folder. Other folders only contain constant and function definitions, style sheets, and images, which are included or referred from the main PHP files. Basically, there is no static HTML file. All HTML contents are rendered from the PHP files. It does not use any specific framework such as Laravel,

Symfony, or CodeIgniter, instead, the entire codebase follows a particular design pattern.

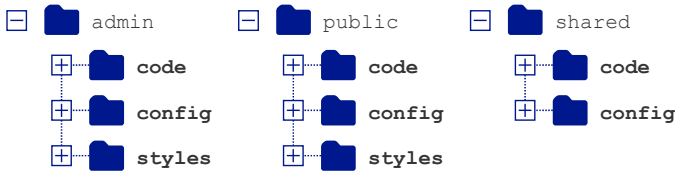

Figure 4. The 2nd level of the file structure in TCExam

Codes in shared folder consist of a declaration of constant and global variables and a definition of library functions. They are included in and called from the main codes. Except for the declaration of constants and global variables, there is no output rendered from these codes. Direct call to these PHP files from the browser only returns a blank page.

Figure 5 shows the structure of a PHP file in `admin` and `public` folders. Each 'layer' refers to a separate PHP file which is included in the main PHP file, except for the green layer which contains the business logic and functions calls inside the main PHP file. The structure is quite complex and uses the old-fashion structured programming rather than the modern object-oriented paradigm. However, they follow a common rule so the debugging of the execution is relatively easy to conduct.

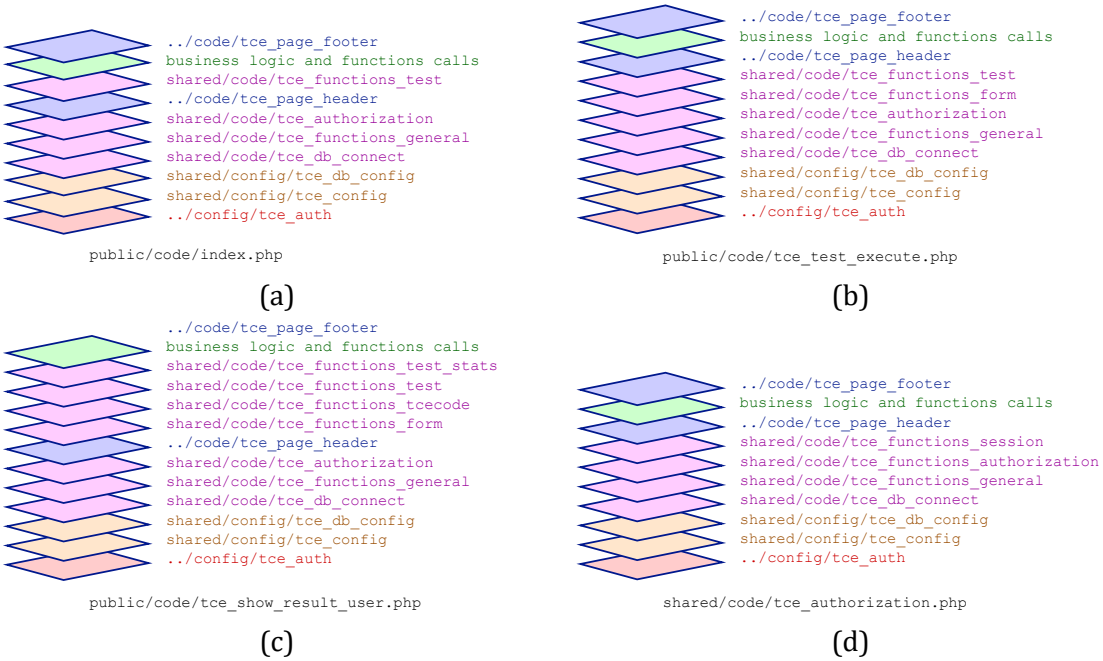

Figure 5. The structure of PHP files in TCExam

4.3 Investigation Process and Problem Solving

In this section, we describe the process of our investigation to find the problem. First, we search for the problem of codes (the PHP files) from the `public` folder that handles the action A1, A3, A4, and A7 from Figure 1. We found that the codes handling the actions are `tce_authorization.php`, `tce_test_execute.php`, and `tce_show_result_user.php` as shown in (d), (b), and (c) in Figure 5. We then inspected the content of those files and searched for suspected codes. In this study, we focused on the actions A1, A3, and A7. What we were looking for is the inappropriate programming logic, unoptimized SQL query and its execution, and high-processing demand of function. When we found a suspected code, then we modified the code and measured the execution time. We describe the procedure of our experiment and the analysis in the next section.

The code for the actions A1 and A3 uses the `tce_authorization.php` where exists in `shared/code` folder. This code consists of functions for user authorization purposes. In the original code, it uses the PHP function `password_hash` to encrypt (and match) the user's password and token used in a test. The function uses the Bcrypt algorithm. The Bcrypt is known as being a strong one-way hashing algorithm at the cost of high demand of CPU processing. We suspected it causing a bottleneck when a large number try to log in or enter the test token. We try to modify it using other lower-demand of CPU processing algorithms such as SHA256 or SHA1 as well as try to scale up the CPU by adding the number of cores to see the differences.

The code for the action A7 uses `tce_show_result_user.php` where exists in `public/code` folder. After an applicant finished the test, he/she can show the result of the test including the statistics. The code executes some queries (SQL) from the database to gather the test data and calculate the statistics. The original code generates a query (SQL) to be executed dynamically by injecting values of a specific user and the current test to the query string, then executes it to get the result. It does not use the parameterized query. We first suspected that was the problem.

```
$question_max_score = ($testdata['test_score_right'] * 1);
$question_half_score = ($question_max_score / 2);
$qright = F_count_rows($sqltot, $sqlw.' AND
    testlog_question_id='.$m['question_id'].' AND
    testlog_score>'.$question_half_score.'');
$qwrong = F_count_rows($sqltot, $sqlw.' AND
    testlog_question_id='.$m['question_id'].' AND
    testlog_score<='.$question_half_score.'');
$qunanswered = F_count_rows($sqltot, $sqlw.' AND
    testlog_question_id='.$m['question_id'].' AND
    testlog_change_time IS NULL');
$qundisplayed = F_count_rows($sqltot, $sqlw.' AND
    testlog_question_id='.$m['question_id'].' AND
    testlog_display_time IS NULL');
$qunrated = F_count_rows($sqltot, $sqlw.' AND
    testlog_question_id='.$m['question_id'].' AND
    testlog_score IS NULL');
```

Figure 6. The non-parameterized and unoptimized original query

```
This algorithm shows how the program calculate the result of the test.

For the given test data:
  Calculate the max score of the right answer
  Calculate the half-max score of the right answer; used to determine
    whether the score of the question was right or not
  Generate the query to count the number of rows in the database of which
    the answer was right using its score compared to the half-max score
  Generate the query to count the number of rows in the database of which
    the answer was wrong using its score compared to the half-max score
  Generate the query to count the number of rows in the database of which
    there was no answer by compared it to NULL
  Generate the query to count the number of rows in the database of which
    the question has not been displayed
  Generate the query to count the number of rows in the database of which
    the question has not been rated
  (Not shown here) Each query is executed one after another
```

Figure 7. The pseudo algorithm of non-parameterized and unoptimized query

We also find that there is a number of small queries executed iteratively which we thought also that this is the reason why the performance is degraded especially when accessed by multiple users concurrently because of the excessive query round-trips have happened. As shown in Figure 6, there are five queries being executed to get the value of five values related to the number of right and false answers and other statistics from the database. The pseudo algorithm of the code is shown in Figure 7. We modified those queries by combined them into one query so that the round-trip can be avoided. The modified codes are shown in Figure 8. These codes are written in `tce_function_test_stats.php` where exists in `shared/code` folder, which is included in the `tce_show_result_user.php`. The pseudo algorithm of the modified codes is shown in Figure 9.

```
$sqlq_count = 'SELECT 1';
$sqlq_count .= ' , SUM(CASE WHEN testlog_score>'.
    $question_half_score.' THEN 1 ELSE 0 END) AS qright';
$sqlq_count .= ' , SUM(CASE WHEN testlog_score<='.
    $question_half_score.' THEN 1 ELSE 0 END) AS qwrong';
$sqlq_count .= ' , SUM(CASE WHEN testlog_change_time IS NULL THEN 1
    ELSE 0 END) AS qunanswered';
$sqlq_count .= ' , SUM(CASE WHEN testlog_display_time IS NULL THEN
    1 ELSE 0 END) AS qundisplayed';
$sqlq_count .= ' , SUM(CASE WHEN testlog_score IS NULL THEN 1 ELSE
    0 END) AS qunrated';
$sqlq_count .= ' FROM '.$sqltot;
$sqlq_count .= ' '.$sqlw;
$sqlq_count .= ' AND testlog_question_id=?';
$sqlq_count_stmt = mysqli_prepare($db, $sqlq_count);
mysqli_stmt_bind_param($sqlq_count_stmt, 'i', $p_question_id);
```

Figure 8. The parameterized and optimized query

```
This algorithm shows how the program calculate the result of the test.

For the given test data:
  (not shown here) Precalculate the max score and the half-max score
  Compose in one string..
    Add a query field to calculate the number of test log data where the
      score is higher than the half-max score (for the correct answer)
    Add a query field to calculate the number of test log data where the
      score is lower than or equal to the half-max score (for the
      wrong answer)
    Add a query field to calculate the sum of scores for the entire test
      log where there is no change to the update time (for the unanswered
      question)
    Add a query field to calculate the sum of scores for the entire test
      log where there is display time (for the undisplayed question)
    Add a query field to calculate the sum of scores for the entire test
      log where the score is NULL (for the unrated question)
  (not showen here) The query is executed once
```

Figure 9. The pseudo algorithm of the parameterized and optimized query

5. EXPERIMENT AND ANALYSIS

In this section, we describe the procedure of our experiment and the analysis of the data collected. There are two groups of parameters we used in the experiment. The first one is from the application which includes the programming logic and the query (SQL) including its execution codes. The second one is the hardware configuration which includes the number of cores and the architecture whether it is a single or separated host. For each configuration, we measured the execution time to see the effect. We simulated the usage of multiple concurrent users using parallelized HTTP requests executed with Linux background command (`&`), `xargs`, and GNU `parallel`. The design of the experiment is shown in Table 1, and the sample data collected from the experiment are shown in Table 2 and 3.

Table 1. The design of the experiment

Parameter	Criteria	Configurations
Application	Programming logic	Bcrypt, SHA256, SHA1
	Query (SQL)	Not-parametrized vs Parameterized vs Parameterized and Optimized
Hardware	Number of Cores	2, 4, 8, 16, 20, 24
	Architecture	Single vs Separated host
Users	Number of concurrent users	1000 users which are simulated in 200 parallel processes

Table 2. The data collected from the experiment (excerpt)

original 4-core		time	prepared stmt 4-core		time	optimized query 4-core		time	original 12-core		time	prepared stmt 12-core		time
[	2020-05-20 23:24:56	0.00	[	2020-05-20 23:43:53	0.00	[	2020-05-20 23:53:51	0.00	[	2020-05-20 22:45:31	0.00	[	2020-05-20 22:50:31	0.00
[	2020-05-20 23:24:56	0.00	[	2020-05-20 23:43:53	0.00	[	2020-05-20 23:53:51	0.00	[	2020-05-20 22:45:31	0.00	[	2020-05-20 22:50:31	0.00
[	2020-05-20 23:24:56	0.00	[	2020-05-20 23:43:53	0.00	[	2020-05-20 23:53:51	0.00	[	2020-05-20 22:45:31	0.00	[	2020-05-20 22:50:31	0.00
[	2020-05-20 23:24:56	0.00	[	2020-05-20 23:43:53	0.00	[	2020-05-20 23:53:51	0.00	[	2020-05-20 22:45:31	0.00	[	2020-05-20 22:50:31	0.00
[	2020-05-20 23:24:56	0.00	[	2020-05-20 23:43:53	0.00	[	2020-05-20 23:53:51	0.00	[	2020-05-20 22:45:31	0.00	[	2020-05-20 22:50:31	0.00
[	2020-05-20 23:24:57	1.00	[	2020-05-20 23:43:53	0.00	[	2020-05-20 23:53:51	0.00	[	2020-05-20 22:45:31	0.00	[	2020-05-20 22:50:31	0.00
[	2020-05-20 23:24:57	1.00	[	2020-05-20 23:43:53	0.00	[	2020-05-20 23:53:51	0.00	[	2020-05-20 22:45:31	0.00	[	2020-05-20 22:50:31	0.00
[	2020-05-20 23:24:57	1.00	[	2020-05-20 23:43:53	0.00	[	2020-05-20 23:53:51	0.00	[	2020-05-20 22:45:31	0.00	[	2020-05-20 22:50:31	0.00
[	2020-05-20 23:24:59	3.00	[	2020-05-20 23:43:53	0.00	[	2020-05-20 23:53:51	0.00	[	2020-05-20 22:45:31	0.00	[	2020-05-20 22:50:31	0.00
[	2020-05-20 23:24:59	3.00	[	2020-05-20 23:43:54	1.00	[	2020-05-20 23:53:52	1.00	[	2020-05-20 22:45:31	0.00	[	2020-05-20 22:50:31	0.00
[	2020-05-20 23:24:59	3.00	[	2020-05-20 23:43:54	1.00	[	2020-05-20 23:53:52	1.00	[	2020-05-20 22:45:31	0.00	[	2020-05-20 22:50:32	1.00
[	2020-05-20 23:24:59	3.00	[	2020-05-20 23:43:56	3.00	[	2020-05-20 23:53:52	1.00	[	2020-05-20 22:45:33	2.00	[	2020-05-20 22:50:32	1.00
[	2020-05-20 23:24:59	3.00	[	2020-05-20 23:43:56	3.00	[	2020-05-20 23:53:52	1.00	[	2020-05-20 22:45:33	2.00	[	2020-05-20 22:50:32	1.00
[	2020-05-20 23:24:59	3.00	[	2020-05-20 23:43:56	3.00	[	2020-05-20 23:53:52	1.00	[	2020-05-20 22:45:33	2.00	[	2020-05-20 22:50:32	1.00
[	2020-05-20 23:24:59	3.00	[	2020-05-20 23:43:56	3.00	[	2020-05-20 23:53:52	1.00	[	2020-05-20 22:45:33	2.00	[	2020-05-20 22:50:32	1.00
[	2020-05-20 23:25:00	4.00	[	2020-05-20 23:43:56	3.00	[	2020-05-20 23:53:52	1.00	[	2020-05-20 22:45:33	2.00	[	2020-05-20 22:50:32	1.00
[	2020-05-20 23:25:01	5.00	[	2020-05-20 23:43:57	4.00	[	2020-05-20 23:53:52	1.00	[	2020-05-20 22:45:33	2.00	[	2020-05-20 22:50:32	1.00
[	2020-05-20 23:25:02	6.00	[	2020-05-20 23:43:57	4.00	[	2020-05-20 23:53:52	1.00	[	2020-05-20 22:45:33	2.00	[	2020-05-20 22:50:32	1.00

Table 3. The data collected from the experiment (excerpt)

optimized query 12-core		time	optimized query 12-core		time		original 12-core dist		time	prepared stmt 12-core c		time	optimized query 12-core		time
[	2020-05-20 22:54:55	0.00	[	2020-05-21 21:45:39	0.00	1	[	2020-05-20 22:11:49	0.00	[	2020-05-20 22:19:55	0.00	[	2020-05-20 22:35:46	0.00
[	2020-05-20 22:54:55	0.00	[	2020-05-21 21:45:39	0.00	1	[	2020-05-20 22:11:49	0.00	[	2020-05-20 22:19:55	0.00	[	2020-05-20 22:35:46	0.00
[	2020-05-20 22:54:55	0.00	[	2020-05-21 21:45:39	0.00	1	[	2020-05-20 22:11:49	0.00	[	2020-05-20 22:19:55	0.00	[	2020-05-20 22:35:46	0.00
[	2020-05-20 22:54:55	0.00	[	2020-05-21 21:45:39	0.00	1	[	2020-05-20 22:11:50	1.00	[	2020-05-20 22:19:55	0.00	[	2020-05-20 22:35:46	0.00
[	2020-05-20 22:54:55	0.00	[	2020-05-21 21:45:39	0.00	1	[	2020-05-20 22:11:50	1.00	[	2020-05-20 22:19:55	0.00	[	2020-05-20 22:35:46	0.00
[	2020-05-20 22:54:55	0.00	[	2020-05-21 21:45:39	0.00	1	[	2020-05-20 22:11:50	1.00	[	2020-05-20 22:19:55	0.00	[	2020-05-20 22:35:46	0.00
[	2020-05-20 22:54:56	1.00	[	2020-05-21 21:45:39	0.00	1	[	2020-05-20 22:11:50	1.00	[	2020-05-20 22:19:55	0.00	[	2020-05-20 22:35:46	0.00
[	2020-05-20 22:54:56	1.00	[	2020-05-21 21:45:39	0.00	1	[	2020-05-20 22:11:51	2.00	[	2020-05-20 22:19:55	0.00	[	2020-05-20 22:35:46	0.00
[	2020-05-20 22:54:56	1.00	[	2020-05-21 21:45:39	0.00	1	[	2020-05-20 22:11:51	2.00	[	2020-05-20 22:19:55	0.00	[	2020-05-20 22:35:47	1.00
[	2020-05-20 22:54:56	1.00	[	2020-05-21 21:45:39	0.00	0	[	2020-05-20 22:11:51	2.00	[	2020-05-20 22:19:55	0.00	[	2020-05-20 22:35:47	1.00
[	2020-05-20 22:54:56	1.00	[	2020-05-21 21:45:40	1.00	1	[	2020-05-20 22:11:51	2.00	[	2020-05-20 22:19:56	1.00	[	2020-05-20 22:35:47	1.00
[	2020-05-20 22:54:57	2.00	[	2020-05-21 21:45:40	1.00	1	[	2020-05-20 22:11:51	2.00	[	2020-05-20 22:19:57	2.00	[	2020-05-20 22:35:47	1.00
[	2020-05-20 22:54:57	2.00	[	2020-05-21 21:45:40	1.00	1	[	2020-05-20 22:11:51	2.00	[	2020-05-20 22:19:57	2.00	[	2020-05-20 22:35:47	1.00
[	2020-05-20 22:54:58	3.00	[	2020-05-21 21:45:40	1.00	1	[	2020-05-20 22:11:51	2.00	[	2020-05-20 22:19:57	2.00	[	2020-05-20 22:35:47	1.00
[	2020-05-20 22:54:58	3.00	[	2020-05-21 21:45:40	1.00	1	[	2020-05-20 22:11:51	2.00	[	2020-05-20 22:19:57	2.00	[	2020-05-20 22:35:47	1.00
[	2020-05-20 22:54:58	3.00	[	2020-05-21 21:45:40	1.00	1	[	2020-05-20 22:11:51	2.00	[	2020-05-20 22:19:57	2.00	[	2020-05-20 22:35:47	1.00
[	2020-05-20 22:54:59	4.00	[	2020-05-21 21:45:40	1.00	1	[	2020-05-20 22:11:51	2.00	[	2020-05-20 22:19:57	2.00	[	2020-05-20 22:35:47	1.00
[	2020-05-20 22:54:59	4.00	[	2020-05-21 21:45:40	1.00	1	[	2020-05-20 22:11:52	3.00	[	2020-05-20 22:19:57	2.00	[	2020-05-20 22:35:47	1.00

The result of the experiment for the user authorization code is shown in Figure 10. We can see here that the original code, which of the password hashing function is using Bcrypt algorithm with 4-core CPU as the host has the lowest response time in peak usage. While that is expected, it does not fit in our usage. When we assign more cores (12-core in this experiment), we can see an improvement by 58%. Even more improvement can be gained by separating the host/server for application from the host/server for database. Furthermore, when we change the algorithm to SHA256 and SHA1, we can gain more speed. Response time of using SHA256 (or SHA1) was 55% better than using Bcrypt with the same 4-core CPU and 30% better with a 12-core CPU. In this experiment, the best speed was with the SHA256 algorithm and 12-core distributed architecture, which is as expected.

As for the code for displaying the result of a test, the result of the experiment is shown in Figures 11 and 12. From Figure 11, we can see that the original code has the worst performance when accessed by a large number of users concurrently. The prepared (or parameterized) query slightly gives better performance but is not so significant compared to the original one which is not prepared. The optimized (which also uses the prepared query) gives a better response time, which is around 50% better compared to the original code, with the same number of 4-core CPU. There was a slightly better response time when using more CPU core as shown in Figure 12. The best performance in this experiment was with optimized

query and 12-core and separated hosts (for application and database) which give around 56% of better response time.

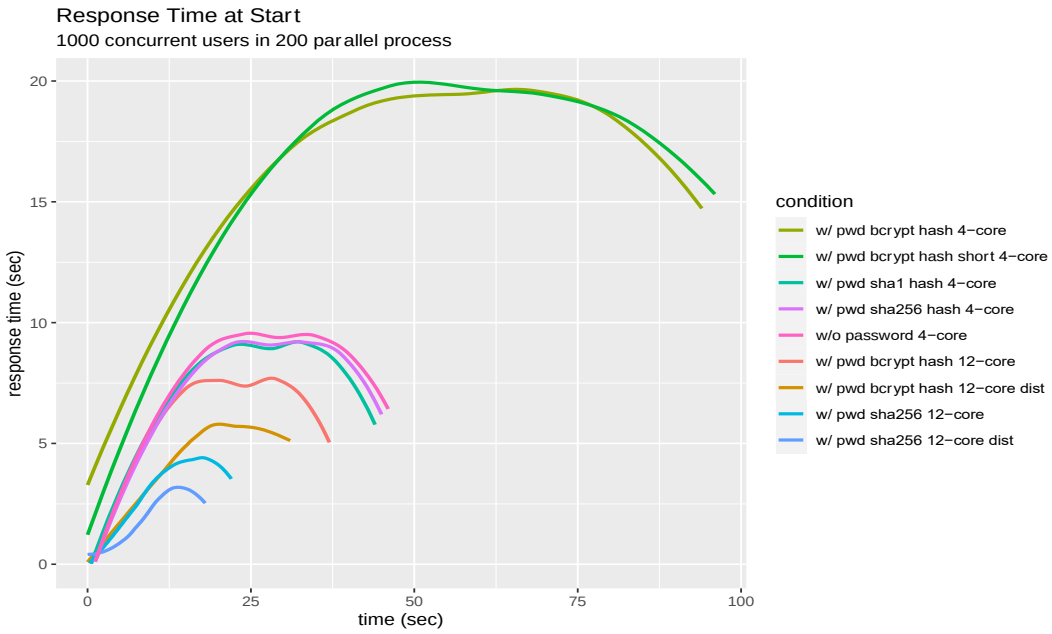

Figure 10. Result of the experiment for the user authorization code

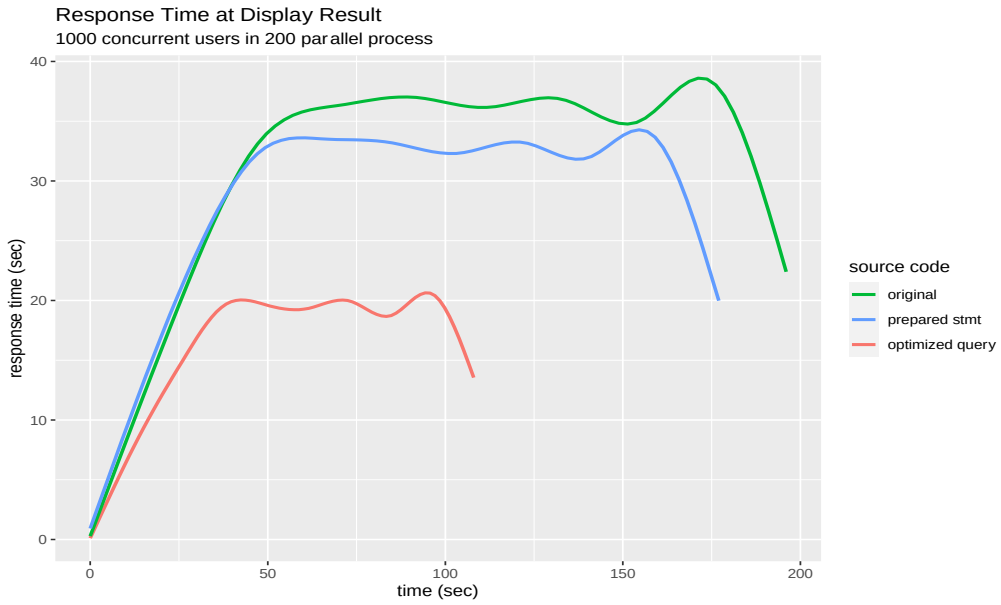

Figure 11. Result of the experiment for the test result display code (1)

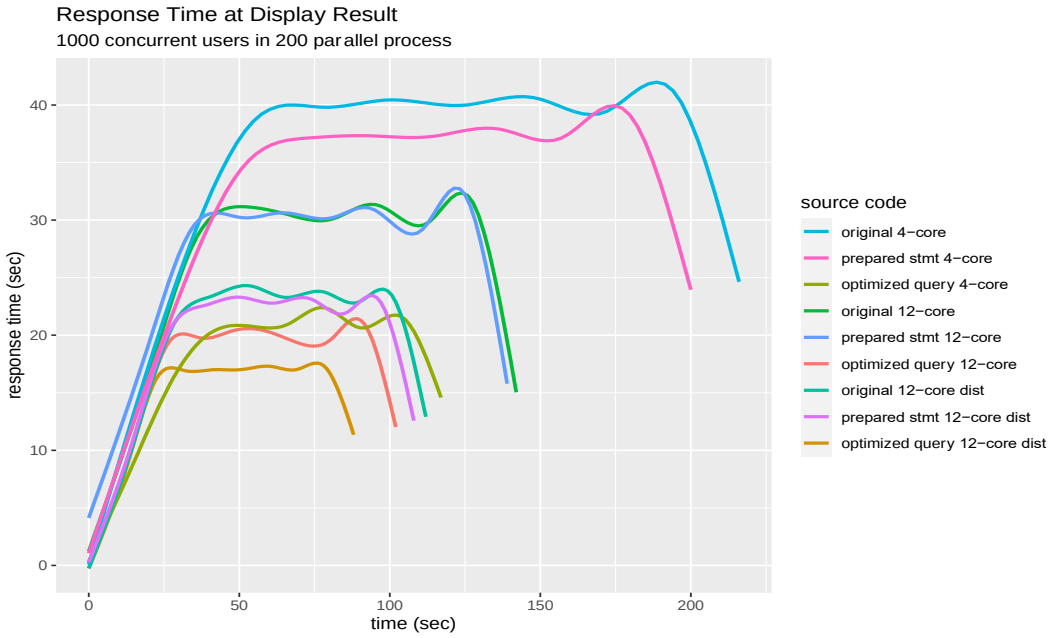

Figure 12. Result of the experiment for the test result display code (2)

6. CONCLUSION

As a free and open-source software (FOSS), TCExam is one solution that can be chosen to implement a computer-based test (CBT). However, from our previous use case, we found a performance degradation problem when used in a large number of applicants concurrently.

This paper reports the result of our investigation in searching the problem and provides the solution by modifying part of the codes. We evaluated the impact of the solution by conducting an experiment in a simulated environment. Including the recommendation to scale up the processor and separate the hosts for the application server, the best improvement can be gained regarding the performance was up to 56%.

The future work includes more investigation on another important part of the codes of TCExam where the test instrument (set of questions) is generated on the fly, which also has a performance degradation in a concurrent large number of users.

REFERENCES

[1] Carole J. Gallagher, **Reconciling a Tradition of Testing with a New Learning Paradigm**, *Educational Psychology Review*, Vol. 15, No. 1, pp. 83-99, 2003.

[2] Cynthia G. Parshall, Judith A. Spray, John C. Kalohn, and Tim Davey, **Practical Considerations in Computer-Based Testing**, *Practical Considerations in Computer-Based Testing*, 2002.

[3] Y. Anang, Takdir, F. Ridho, I. Santoso, L. R. Maghfiroh, S. Mariyah, M. Takahashi, and Y. Watanabe, **Implementation of Computer-Based Test in a Countrywide New Student Recruitment Process**,

Proceedings of the 4th International Conference on Information Technology (InCIT), Bangkok, pp. 268-273, 2019.

[4] Nicola Asuni, **Quality Features of TCExam, an Open-Source Computer-Based Assessment Software**, *JRC Scientific and Technical Reports EUR 23306 EN, Institute for the Protection and Security of the Citizen, Joint Research Centre, European Commission*, Ispra (VA), Italy, 2008.

[5] ISO/IEC, **ISO/IEC 9126*:* Software Engineering – Product Quality**. *ISO/IEC*, 2001.

[6] ISO/IEC, **ISO/IEC 25010:2011: Systems and Software Engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) — System and Software Quality Models**. *ISO/IEC*, 2011.

[7] Michael Russell, Amie Goldberg, and Kathleen O'connor, **Computer-based Testing and Validity: a look back into the future**, *Assessment in Education: Principles, Policy & Practice*, Vol. 10 No. 3, pp. 279–293, 2003.

[8] Chua Yan Piaw, **Replacing Paper-based Testing with Computer-based Testing in Assessment: Are We Doing Wrong?**, *Procedia - Social and Behavioral Sciences*, *Proceedings of the 12 th International Educational Technology Conference - IETC*, pp. 655–664, 2012.

[9] Maria M. Llabre, Nancy E. Clements, Katharine B. Fitzhugh, Gary Lancelotta, Roy D. Mazzagatti, and Nancy Quinones, **The Effect of Computer-Administered Testing on Test Anxiety and Performance**, *Journal of Educational Computing Research*, Vol. 3 No. 4, pp. 429–433, 1987.

[10] Jr. Thomas J. Ward, Simon R. Hooper, and Kathleen M. Hannafin, **The Effect of Computerized Tests on the Performance and Attitudes of College Students**, *Journal of Educational Computing Research*, Vol. 5 No. 3, pp. 327–333, 1989.

[11] Joseph Hardcastle, Cari F. Herrmann-Abell, DeBoer, and E. George, **Comparing Student Performance on Paper-and-Pencil and Computer-Based-Tests**, *In Annual Meeting of the American Educational Research Association*, April 2017.

[12] Dave Bartram, **The International Test Commission Guidelines on Computer-Based and Internet-Delivered Testing**, *Industrial and Organizational Psychology*, Vol, 2 No. 1, pp. 11–13, 2009.

[13] Martha Thurlow, Sheryl S. Lazarus, Debra Albus, and Jennifer Hodgson, **Computer-based Testing: Practices and Considerations. Synthesis report**, *National Center on Educational Outcomes*, 2010.

[14] Pierre Bourque, EÉcole de Technologie Supeérieure (EÉTS), Richard E. (Dick) Fairley, and Software and Systems Engineering Associates (S2EA), **Guide to the Software Engineering Body of Knowledge (SWEBOK®): Version 3.0**, *IEEE Computer Society Press*, 2014.

[15] IEEE Computer Society, **Software and Systems Engineering Vocabulary (SEVOCAB)**, *http://www.computer.org/sevocab/*, 2021. Accessed: Feb 12, 2021.

[16] Qingping He and Peter Tymms, **A computer-assisted test design and diagnosis system for use by classroom teachers**, *Journal of Computer Assisted Learning*, Vol. 21 No. 6, pp. 419–429, 2005.

[17] Jatin Shah, Dimple Rajgor, Meenakshi Vaghasia, Amruta Phadtare, Shreyasee Pradhan, Elias Carvalho, and Ricardo Pietrobon, **WriteSim TCExam - An open source text simulation environment for training novice researchers in scientific writing**, BMC Medical Education 2010, 10:39, pp. 1-14, 2010.

[18] M. Ismail, W. Z. A. Mokhtar, N. N. M. Nasir, N. R. L. Rashid, and A. K. Ariffin. **The development of a web-based homework system (wbh) via tcexam**, *Mediterranean Journal of Social Sciences*, Vol. 5 No. 15, 2014.

[19] Ambiyar, Muhammad Luthfi Hamzah, Astri Ayu Purwati, and Eki Saputra, **Computer Based Test Using Tcexam as An Instrument Learning Evaluation**, *International Journal of Scientific & Technology Research*, Vol. 8, pp. 1066–1069, 2019.

[20] James Shaw, **Web Application Performance Testing—a Case Study of an On-line Learning Application**, *BT Technology Journal*, Vol. 18 No. 2, pp. 79–86, 2000.

[21] Eljona Proko and Ilia Ninka, **Analyzing and Testing Web Application Performance**, *International Journal of Engineering and Science*, Vol. 3 No. 10, pp. 47–50, 2013.

[22] Rijwan Khan and Mohd Amjad, **Performance testing (load) of web applications based on test case management**, *Perspectives in Science*, Vol. 8, pp. 355–357, 2016.

[23] F. F. Hardiansyah, J. L. Buliali, and W. Wibisono, **Internet Connection Control based on Idle Time Using User Behavior Pattern Analysis**, *EMITTER International Journal of Engineering Technology*., Vol. 2, No. 2, pp. 49-61, 2014.