INNOVATIVE ALGORITHM OPTIMIZED FOR MULTIPLE ROUNDS AND STATELESS ASSIGNING OF TASKS NON-DUPLICATE FOR THE SAME SUBJECTS

[a]PETR VOBORNÍK, [b]RADEK NĚMEC

University of Hradec Králové, Rokitanského 62, 50003 Hradec Králové, Czech Republic
email: [a]petr.vobornik@uhk.cz, [b]radek.nemec@uhk.cz

Supported by Specific research project of University of Hradec Králové, Faculty of Science.

Abstract: Optimization of selection of tasks or questions assigned to different subjects from a larger set of tasks is a problematic topic, often solved not only in schools. The article presents the principles and original algorithmic procedures of the new tool that approaches the solution in a completely innovative way. This tool directs pseudorandom value generators to maximize the efficiency of the tasks that are entered in electronically form. The multiplatform open source implementation of this algorithm can be directly linked to LMS Moodle, it can be used alone or integrated into an own application, for multiple rounds and stateless assigning tasks or questions randomly selected from multiple options so as to avoid premature duplicate selection in different rounds for the same subjects. The principles used in this algorithm and presented in the article may also be an inspiration not only for the implementation of future learning applications but also for further development of the theory of selection functions, whether in this or an entirely different science branch.

Keywords: Tasks chooser, task assignment, random numbers, Moodle, External tool, eLearning tool, examination, LTI.

1 Introduction

Optimizing the selection of tasks or questions generated from a larger set of tasks for different subjects, e.g. students but not only for them, is a frequently solved problem not only at schools. It is not rare that a simple random selection is used, whether in the form of a draw or any automatically processed software. There are procedures for the teaching phase (see [1], [2], [3]) and tools (e.g. see *Universal Testing Environment* [4], *SuperMemo* [5], *Anki* [6], *Dril* [7] etc.) to select the questions in order to maximize the memorizing effect. However, a suitable and universal tool is missing for the final testing when it is necessary to specify objectively and sometimes assign repeatedly the task according to various parameters to a large group of people. We are going to present our solution to this problem.

1.1 Current Solutions and Their Disadvantages

Moodle is a great learning management system (LMS). Apart from other things, it enables assigning work or test tasks to students or collaborating groups of students, submitting the solutions and individual evaluation. This functionality is provided by the *Assignment activity* (module) which is part of the basic installation of Moodle. This activity can be used both for assigning homework and for an assignment of exam tasks (collection and evaluation), i.e. tasks assigned, elaborated and submitted within one block under the teacher's supervision. Homework, like any other modules, can also be made available or hidden to different groups of students, thus making possible to create multiple task assignments for the whole class. [8]

However, what is the procedure in case an original and unique assignment for each student is to be created? It is of course possible to create a separate *Assignment activity* with individual work for each student and make it visible only to them. Nevertheless, this solution has a number of disadvantages, such as laborious and difficult preparation and course arrangements (many copies of the activity will be created), loss of advantageous collective grading, confusion in grading records, degradation of computer facilities to a test paper, absence of random choice, also creating individual assignments of exam tasks for particular students may not always be objective, etc.

Another option is to use other Moodle activities. For example, the *Quiz activity* allows the teachers to work with random elements and select only a few of them from a large question bank for each student [9]. The *Quiz activity* is primarily designed as a quick exam and provides tasks of only certain types (e.g. multiple choice, short text answer question, numeral response, etc.). The advantage is that the quiz answers are scored automatically. However, the type of task "elaborate a project on the theme … in application …, upload and submit it here" is not available, and it would not even fit into the overall quiz concept.

However, there is one module (activity) that allows you to avoid the problem of the absence of some function. Its name is *External tool*. It enables to set up and configure external web services called via a URL, while it sends additional parameters via the LTI[1] protocol using the POST method[2]. Thanks to this it is possible for each potential request to either create your own or use an existing online service that can process it based on input data. These are secured by the OAuth[3] protocol and, in addition to their own defined parameters, they contain the identifiers of the course and the student who used the link [10]. Whereas it is a built-in Moodle component, time availability and current visibility for students can be easily managed right here [11, p. 82].

This module could therefore be used as a solution, but no such online tool has been available so far that would meet the system requirements.

2 System Requirements

For the purpose of assigning individual, but randomly chosen and combined tasks, the following list of requirements of functions was compiled that the tool should support for their selection.

1) Select tasks randomly and evenly from the whole range of available tasks.
2) The task should be assigned to individual student permanently to see the same assignment when the page is reloaded, after repeated login or when reviewing the assessment even a few days later. The teacher must also have the possibility to reproduce this exact assignment.
3) Multiple round assignment:
    a) If the assignment has multiple rounds (e.g. several exam dates), the student should never be given the same assignment if there are some tasks that have not been assigned to him/her yet.
    b) The assignment (and evaluation) of previous rounds should still be available to the student.
4) Selection of multiple tasks:
    a) The selection could include not only one (1) of the potential tasks, but also a combination of multiple tasks of the given count (1 to N).
    b) For the selection of multiple tasks (2 to N), the same limitations (no recurrence, and back availability) should apply as for the selection of one single task.
    c) It should be possible to sort the tasks into groups and then specify in given assignment settings how many tasks are to be chosen from each group.
    d) It should be possible to display the selection of multiple tasks in the overall text of assignment in both random and predetermined order.
5) Text of the assignment should be able to include random elements (words, characters and numbers).
6) The access to a given set of questions should be password-protected. Without a password the tasks will not be selected or displayed.
7) The system should be in compliance with the GDPR[4] parameters, i.e. save all data about all its use (e.g. history of tasks assigned to one student) anonymously, encrypted, or preferably not to save them at all, but the requirements 2 and 3b should be met.

[1] LTI – Learning Tools Interoperability [10]
[2] POST method is used to send data to a server which is stored in the request body of the HTTP request [28]
[3] OAuth – Web Authorization Protocol [29]
[4] GDPR – General Data Protection Regulation [35]

3 Results

One of the currently most versatile technologies, namely *.NET* with the *C#* language was chosen to create a system that would meet all requirements. It can be connected to Moodle as an *External tool* and at the same time, it can work independently and provide an API[5] for use of other applications. The *.NET Standard*, i.e. a type of project that can be compiled into libraries and applications available for the widest range of target systems[6], was used as the basic class library because it manages the entire process of setting tasks. The *ASP.NET Core* technology, a highly cross-platform which provides a web interface, was chosen for the development of the application part. It can be used on web servers running operating system the Windows Server or various distributions of Linux [12].

3.1 Algorithm of Selection

The main goal is to randomly select tasks that will not be repeated for the given student in following rounds. If a quality generator of pseudorandom values is used instead of a purely random generator, an arbitrarily long line of numbers generated by it depends solely on the default value of the generator, the so-called *seed*. If we relate this value to the student in some way, then it would always be possible to reproduce a previously generated set of pseudorandom values for that student. His/her ID, login or e-mail address can be used, but only in combination with another hidden value (available only to the task setter), so that the student cannot see the task in advance.

The second essential parameter is the round number (*round*), which defines how many times the task has been selected for the student in order to choose a different work for the new round, and at the same time to see the same assignment in the already completed round, as when the page with the tasks for that round was first loaded.

The basic procedure is shown in diagram, see Fig. 1.

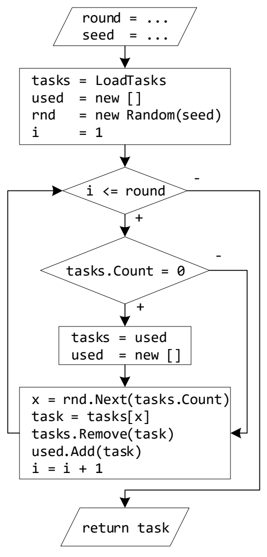

Fig. 1 – Basic schema of how the selection function works for the *round* and the initial value *seed*

The input parameters are the *seed* (a text is converted to an integer by the hash[7] function) and the *round* (an integer from 1 up). The algorithm then retrieves a list of available *tasks* from a file or a database, prepares the second empty list *used* to record the tasks used in the previous rounds, creates a new generator of pseudorandom values *rnd* with an initial *seed* value, and sets the round counter *i* to 1.

The next steps are repeated according to the required number of the rounds, i.e. if *round* is equal e.g. six, then the task selection will be done six times and the last selected one will be returned. In each round, one task is randomly chosen from the loaded set of tasks, it is removed from this set (*tasks*) and added to the set of already used tasks (*used*). If this round was the last, this task is returned as a result of the entire selection process. If it is not, the selection from the currently shortened list (*tasks*) of loaded tasks is repeated.

If the list of loaded tasks (*tasks*) is empty and the object task has not been selected yet, all tasks will be returned to the possible selection (each of them has been used once in previous rounds), and the list of used tasks (*used*) will be emptied (so far none of the tasks was chosen twice).

Of course, it is ideal if the number of possible tasks exceeds the number of rounds. However, this procedure can ensure the required functionality even if the opposite is true (the number of rounds is higher than the number of tasks, even several times).

3.2 Selection of Multiple Tasks

The given solution applies so far only to cases where the algorithm was supposed to select exactly one task from several possible. However, the System Requirements in point 4 also state the possibility of selecting multiple tasks, and according to 4c it should be possible to sort the tasks into categories and limit the number of selected tasks so that a specific number of tasks is chosen from each category.

To select more than one task without further limitations, it is enough to repeat the current procedure, as if the second and following tasks were selected in the next round. However, if there is still a selection restriction by different categories, it greatly expands the possibilities of task definition. These tasks may not always be completely independent, but together they can form a more complex assignment. The following mathematical problem demonstrates this possibility (Tab. 1).

Calculate required values for the triangle of known following data and draw it in the given way.

A) Category A (value declaration) – choose 1 of tasks
   1) $a = 5$, $b = 3$, $\gamma = 30°$
   2) $a = 10$, $\beta = 25°$, $\gamma = 40°$
   3) $a = 4$, $b = 2$, $c = 7$
B) Category B (what is to calculate) – choose 2 tasks
   1) area of the triangle S = ?
   2) perimeter of the triangle O = ?
   3) size of missing angles = ?
   4) length of the sagitta for the *a* side
C) Category A (drafting) – choose 1 of tasks
   1) draw this triangle with the inscribed circle
   2) draw this triangle with the circumscribed circle

Tab. 1 – Categorized set of tasks for the selection of possible assignment of 4 tasks (1+2+1)

Thus, up to 36 different assignments ($3 \cdot \binom{4}{2} \cdot 2$) with 4 tasks (1x from 3 in A, 2x from 4 in B, 1x from 2 in C) can be generated from the set of tasks in Tab. 1.

Modifying the algorithm to support tasks selection from categories is not complicated at all. Repeat the selection in the required number for each desired category separately. So, before we select a random task, we filter the list *tasks* into an auxiliary list, e.g. with the help of method *Where* of the LINQ[8] tool.

The next step was to establish a log format, i.e. a mini-language (such as [13] or [14]) that would allow you to easily, clearly and

[5] API – Application Programming Interface
[6] .NET Standard libraries can be added to project running on OS Windows, Android, iOS, MAC OSx, Linux… [31]
[7] *hash is one-way (irreversible) computationally efficient function mapping binary strings of arbitrary length to strings of fixed length, it is called the hash-value* [19]

[8] *LINQ (Language Integrated Query) is the part of the .NET Framework that provides a generic approach to querying data from different data sources* [32]

intuitively, but also comprehensively and uniquely define the number of tasks to be selected from which category. At the same time, this format should be easily programmable. [4, pp. 148-168]

If it were not for categories, a simple number would be sufficient, which is also the first possibility to define the number of selected tasks. With categories, however, there are situations where it might be important to have the possibility to define the following cases.

- A certain number of tasks from a specific category, a different number of tasks from a different category
- A selection of a specific number of tasks from several different categories
- A selection of a specific number of tasks from each category

Similarly to the SQL [15] language, an effort was made here to design this mini-language so that its notation would be an analogy of a classic sentence expressing the exact tasks number requirements. Instead of words, separators have been used that are commonly used in such cases and whose choice should therefore be intuitive. These characters, instead of whole words, shorten the total length of the notation and also facilitate programming of such a definition. These are the following characters (sorted by their priority):

- **:** – separator of total number of questions from closer specification
- **,** – separator of uniform requirements on different categories
- **/** – symbolizes “from” to specify entered number of tasks from a particular category
- **|** – the phrase “or” to select from several possible categories
- ***** – wildcard character “any previously not mentioned category”

Several particular cases of defining the number of selected tasks using the above-described characters are shown in Tab. 2.

Notation	Meaning
1	1 task from all possible (default choice for selection)
3	3 tasks from all possible regardless of cat.
1/A	1 task from category A
1/A,2/B,1/C	1 task from category A, 2 from B, 1 from C (see Tab. 1)
4/A|B,2/C	4 tasks from categories A or B, 2 tasks from category C
1/*	One task from each category
1/A,1/D,2/*	1 task from category A, 1 from D, and 2 from all other categories
5:1/*	5 tasks in total, maximum one (or none) from each category
6:1/A,1/B,1/*	6 tasks in total, 1 from A, 1 from B and remaining 4 tasks, one from other categories

Tab. 2 – Demonstration of notification of various requirements for the number of tasks, total and categorized in its own intuitive mini-language

Each task can then be labelled with an attribute to determine which category or categories it belongs to (e.g. “a,b,c“). Thanks to categories and number definition, some tasks can be omitted (filtered) from the selection. If random change of tasks order is off, some categories can also be used as headings of the following tasks section if necessary.

3.3 Random Elements

A significant number of tasks is or can be defined by just changing one word or parameter value to create a completely new tasks, even of the same type.

For example, an assignment “*Create a presentation in [Sway,PowerPoint] on topic [computers,animals,your favourite sport,your favourite serial story]*.“, where only one of the comma-separated values in each brackets is always selected and inserted in a sentence. Thus, a single task equals 2*4=8 in a chosen environment and on a given topic, and there would be no problem to add more of them.

The situation is even easier with numerical values. If in Tab. 1, for example in category A, at least one of the values (side length or angle size) of each variant was randomly generated in the range of just 10 numbers (e.g. for A1: γ=[30-39]), the number of variants (of different assignments) will increase from 3 to 30 for category A, and the total number of possible assignment variants will increase from the original 36 to 360.[9]

A similar possibility of entering random values into test questions has already been implemented e.g. in the *Universal Testing Environment* (see [4, pp. 48-53], [16] and [10]). To choose tasks, support for selecting random words (parts of the text from the list of options), integers (defined from-to), characters (determined by the first and last possible characters in ASCII code[10]), and XML elements have been used for the system. In the future, it would be possible to add also support for further use of already selected values (repeated list of values, or support for basic calculations with chosen random numbers[11]).

For the selection of random values, a new pseudorandom number generator is created in the system with an initial value derived from the *seed*, so that any changes (adding, deleting or changes in specification or range of values) would not affect the main generator designated for the tasks selection. Whereas the primary task of the generator is to choose tasks and random values are just a supplementary option, the non-recurrence of the assignment cannot be guaranteed in their case and everything is left to random choice. However, the principle of reproducible assignment is applied here (see the requirement No. 2).

Thanks to random values, the student may be assigned a task that he/she or classmates have previously solved, but the finished solution or correct result known from the previous assignment cannot be used as some of the parameters of the task may differ. At the same time, this is a good way how to prevent the students from cheating and copying the solutions.

3.4 Generator of Pseudorandom Values

The functionality of the entire system is absolutely dependent on the generator of pseudorandom values. The selection algorithm is built in such a way that in fact it is possible to use any pseudorandom value generator, either as an implementation of some classical ones (see [17, pp. 10-40]), a library of generator from third parties (e.g. [18]), or more advanced cryptography techniques can be used (e.g. such as a multilevel hash series used for encrypting by *perfect cipher* in [19]).

For standard use, it seemed logical to use the integrated *.NET Framework* generator – class *Random* (see [20, pp. 52-60] and source code [21]). Its properties have been tested independently many times (e.g. see [20, pp. 54-57], [22], [23]) and its suitability for the use in this project was then tested for the following aspects.

- The generated series of numbers must be theoretically infinite.
- The generated series of numbers must be statistically uniform.

[9] In task A-3 it is of course not possible to select numbers completely randomly, one of the 10 pre-set triplets of values have to be selected
[10] ASCII – American Standard Code for Information Interchange
[11] E.g. [a=1-10] * [b=1-10] = {a*b}

- The generated series of numbers must be fully dependent only on the default *seed* value, and if it is the same, the entire series of numbers and its each member must also be the same independently of the hardware, operating system and version of the development environment on which is the application compiled or run.

3.5 Implementation

Class libraries handling the entire algorithm of selection have been developed separately (.NET Standard) so that they can eventually be linked to any type of project or application. For testing the generator, three "one-line" applications were then created: console (.NET Framework) for Windows, console (.NET Core) for Linux and mobile (Xamarin [24]) for OS Android. A web application (ASP.NET Core) was created for use via web interface and to link to the LMS Moodle via the *External tool* module.

The web application supports three interfaces to query tasks that can be used: via URL parameters, via POST (hidden), and via POST parameters from Moodle *External tool* (LTI). However, this last variant works in such a way that the parameters are read, translated into their POST form and the request is redirected to the second version of the interface (similar to [10]). The Moodle user login combined with a secret part of the password which can be set in the *External tool* module, is used as a *seed* here.

Tasks sets can currently be retrieved from separate XML files with a structure that enables both writing and processing of all required properties. In comparison with the original requirements, the option of basic multilevel settings of conditions under which the tasks are to be selected, has also been added. The same format for defining the tasks set is also planned for the eventual extension of the application to support their database management via the web administration interface, where XML sets will be saved as DB type XML[12].

The web application is stateless and it does not save anything itself (thus fulfilling the GDPR conditions). It only processes incoming requests and hands back the generated task assignment either in HTML form or as plain text. However, it supports the possibility of logging activation (e.g. when debugging the set of tasks or setting a communication), when into the text file, individual lines are recorded with dates and times of the incoming request and its form is transformed into the final URL version.

All source codes of all libraries, projects and applications were placed on GitHub[13] under the MIT[14] license, where it can be studied in detail, downloaded and tested, or further developed or included into your own application.

4 Use in Practice

This version as well as the previous one of the algorithm and web application has already been successfully used in practice several times. In the first version, the system connected to Moodle via the *External tool* was first used in December 2016 for random selection of one task from more possible ones for the exam in *Database Systems* course for a total of 26 students. 13 tasks were added in turn: 1st round/group 5, 2nd round/group different 5, 3rd and next rounds (replacement/correction of the exam) previous 10 + 3 new tasks.

The Moodle *External tool* was not used directly, but only as an interface that mediated communication with the web application for the tasks selection. The *External tool* was set up to get the text with the task in the HTML format based on the user login name. Although it would work like that independently, it was hidden from the students, and the task with a wider description was displayed to students through the *Assignment activity* within its text, as a nested *iFrame*[15] referring to the URL of the *External tool*. It read the task text and included it into the text of assignment, and it was not noticeable without examining the source code of the page. In addition, the task text was protected by a special HTML element against marking and copying.

Since the following school year (2017/18), the system was further developed, and new features described in this article were added. At the same time, it has been used in more subjects (Programming 1, Application Software etc.) at the *Faculty of Science* on *University of Hradec Králové*, but also at the *Secondary school "Podorlické vzdělávací centrum Dobruška"*.

By the end of summer term of the school year 2018/19, the system was also used for the collective assignment of question for the final state examinations at the Department of Informatics.

5 Conclusion

The article introduced innovative principles and original algorithmic procedures of a new cross-platform open source tool for multiple round and stateless selection of tasks or questions randomly chosen from multiple options, avoiding premature duplicate selection in different rounds for the same subjects. This tool works with directing pseudorandom value generators to maximize the efficiency of tasks entered electronically. The result thus meets all the requirements and fulfils all the objectives stated in the introduction.

The tool, in its current versions, has been successfully used in practice for several years for exams at secondary school and university. It was also newly tested for assigning questions during state final examinations. Its potential does not end there, it could equally well be used also for assigning questions for secondary school leaving exam or driving tests. However, it can also help during preparation for all these exams.

Thanks to the transparency of open source codes placed on GitHub [25] and possibility of reconstruction of the entire draw process at any time and by anyone, this project could replace not always transparent systems for the draws public contracts, or may find use in lottery. Random values are also needed in modelling and simulation, cryptography, evolutionary algorithms or games.

The development of the system, both of the selection algorithm itself and other functions for tasks sets, as well as applications providing this library with a user or administration interface, has not finished and will continue in line with the findings from its use in practice. GitHub also provides the opportunity to participate in development to any other developers who can, based on the current version, create a new development branch, and either use their improvements themselves, provide them as a basis for further development in their independent branch, or offer them for reconnection with the main development branch of the project. [26]

Principles used in the main algorithm and presented in this article can also be an inspiration not only for the implementation of future educational applications (e.g. [27]), but also for the further development of the selection function theories whether in this or a completely different field.

Literature

1. Kintsch, W.: *Memory and Cognition*. Wiley, 1977. ISBN 978-0471480723.
2. Wozniak, P.A., Gorzelanczyk, E.J.: *Optimization of repetition spacing in the practice of learning*. Acta Neurobiologiae Experimentalis. 1994, pp. 59–62.

[12] *Microsoft SQL Server* offers for table attributes as one of the possible XML data types, with advanced content processing capabilities [33]
[13] all source codes of this project are available at https://github.com/PetrVobornik/TasksChooser [25]
[14] MIT is *a short and simple permissive license with conditions only requiring preservation of copyright and license notices. Licensed works, modifications, and larger works may be distributed under different terms and without source code.* [34]

[15] iFrame – *an inline frame is used to embed another document within the current HTML document* [35]

3. Hubálovský, Š., Hubálovská, M., Musílek, M.: *Assessment of the influence of adaptive E-learning on learning effectiveness of primary school pupils*. Computers in Human Behavior. Vol. 92, March 2019, pp. 691–705.
4. Voborník, P.: *Universal Testing Environment*. Ph.D. thesis, Hradec Králové: University of Hradec Králové, 2012.
5. Wozniak, P.A. *Optimization of learning*. Master's thesis, Poznan, Poland: University of Technology in Poznan, 1990. https://www.supermemo.com/en/archives1990-2015/english/ol
6. Hanson, A.E.S., Brown, Ch.M.: *Enhancing L2 learning through a mobile assisted spaced-repetition tool: an effective but bitter pill?* Computer Assisted Language Learning. February 2019, pp. 1–23.
7. Brandejs, M., Brandejsová, J., Misáková, M., Kasprzak, J., Lunter, Ľ.: *Inteligentní dril: studenti méně opakují a více si pamatují*. 7. ročník konference Alternativní metody výuky 2009. Prague, 2009. ISBN 978-80-7041-515-3.
8. Büchner, A.: *Moodle 3 Administration*. Packt Publishing, 2016. ISBN 9781783289721.
9. Gamage, S.H.P.W., Ayres, J.R., Behrend, M.B., Smith, E.J.: *Optimising Moodle quizzes for online assessments*. International Journal of STEM Education. 2019, 6:27.
10. Voborník, P.: *Universal Testing Environment as an External Tool of Moodle*. 10th International Scientific Conference on Distance Learning in Applied Informatics (DiVAI 2014). Štúrovo, Slovakia: Wolters Kluwer, 2014, pp. 215–225. ISBN 978-80-7478-497-2.
11. Voborník, P. *Základní moduly činností v Moodle*. Hradec Králové, 2014.
12. Price, M.J., Khan, O.M.A.: *C# 7 and .NET: Designing Modern Cross-platform Applications: The Open Source revolution of .NET Core*. Packt Publishing, 2018. ISBN 9781789957877.
13. Voborník, P.: *Mini-Language for Effective Definition of the Color Gradients*. Advanced Materials Research. Vols. 1030–1032, September 2014, pp. 1882–1885.
14. Voborník, P.: *Mini-language for efficient and comprehensive definition of time intervals with the possibility of recurrence*. Computing, Control, Information and Education Engineering: Proceedings of the 2015 Second International Conference on Computer, Intelligent and Education Technology (CICET 2015). Guilin, P.R. China: CRC Press, 2015, pp. 737–740. ISBN 978-1-138-02800-5.
15. Hursch, C.J., Hursch, J.L.: *SQL: Structured Query Language*. Subsequent Edition. Windcrest, 1991. ISBN 978-0830688036.
16. Voborník, P.: *Univerzální testovací prostředí*. Sborník příspěvků z konference eLearning 2011. Hradec Králové: Gaudeamus UHK, 2011, pp. 80–85. ISBN 978-80-7435-153-2.
17. Kunth, D.E.: *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms*. 3rd ed., Addison-Wesley, 1998. ISBN 0-201-89684-2.
18. Anger, F.: *Chaotic random number generators with random cycle lengths*. [Online] 25th November 2001. https://www.agner.org/random/theory/chaosran.pdf
19. Voborník, P.: *Migration of the Perfect Cipher to the Current Computing Environment*. WSEAS Transactions on Information Science and Applications. Vol. 11, 2014, pp. 196–203.
20. Sinai, A.: *Pseudo Random Number Generators in Programming Languages*. M.Sc dissertation, Herzlia, Israel: The Interdisciplinary Center, Efi Arazi School of Computer Science, 2011.
21. Microsoft. *Class Random*. Source code .NET Framework 4.6. [Online] GitHub, 15 October 2015 https://github.com/microsoft/referencesource/blob/master/mscorlib/system/random.cs
22. Tezuka, S.: *Uniform Random Numbers: Theory and Practice*. Springer Science & Business Media, 2012. ISBN 978-1461523178.
23. CAcert Research Lab: *Random Number Generator Results*. [Online] http://www.cacert.at/cgi-bin/rngresults
24. Goetz, J., Li, Y.: *Evaluation of Cross-Platform Frameworks for Mobile Applications*. International Journal of Engineering and Innovative Technology (IJEIT). Vol. 8, Issue 3, September 2018, pp. 10–17.
25. Voborník, P.: *TasksChooser project*. [Online] GitHub, 2019. https://github.com/PetrVobornik/TasksChooser
26. Preethi, M.B., Krishnan, D.G., Sivarnjani, G.: *An Overview on GITHUB*. International Journal for Research in Applied Science & Engineering Technology (IJRASET). Vol. 7, January 2019, pp. 132–134.
27. Maněna, V., Milková, E., Pekárková, S., Dostál, R.: *Integration of mobile technologies and social networks into activation methods in education*. International journal of education and information technologies. Vol. 11, 2017, pp. 31-36.
28. Alam, S., Cartledge, C.L., Nelson, M.L.: *Support for Various HTTP Methods on the Web*. Norfolk, VA: Old Dominion University, Computer Science Department, 2014.
29. Leiba, B.: *OAuth Web Authorization Protocol*. IEEE Internet Computing. Vol. 16, January 2012, pp. 74–77.
30. Baxevani, T.: *GDPR Overview*. Thessaloniki, Greece: Alexander Technological Educational Institute of Thessaloniki 2019.
31. Landwerth, I.: *Introducing .NET Standard*. [Online] 26th September 2016. https://devblogs.microsoft.com/dotnet/introducing-net-standard/
32. Freeman, A., Rattz, J.: *Pro LINQ - Language Integrated Query in C# 2010*. Apress, 2010. ISBN 978-1430226536.
33. Pal, S., Cseri, I., Seeliger, O., Schaller, G., Giakoumakis, L., Zolotov, V.: *Indexing XML Data Stored in a Relational Database*. Proceedings of the Thirtieth international conference on Very large data bases. Toronto, Canada, 2004, pp. 1146–1157.
34. GitHub: *MIT License*. [Online] https://choosealicense.com/licenses/mit/
35. AL-Amro, H., El-Qawasmeh, E.: *Discovering security vulnerabilities and leaks in ASP.NET websites*. International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec). Kuala Lumpur, Malaysia: IEEE, 2012. ISBN 978-1-4673-1426-8.

Primary Paper Section: I

Secondary Paper Section: IN