INTERNET CLIENT-SIDE TECHNOLOGY IN EDUCATION

Elza Imnadze Doctoral student of the Faculty of Natural Sciences, Mathematics, Technology and Pharmacy of Sukhumi State University in the Computer Science Program; Ilia State University, Senior Specialist of Information Technologies Office
E-mail: elza.imnadze@iliauni.edu.ge

Nugzar Kereselidze Doctor of Informatics, Corresponding Member of the Tskhum-Abkhazian Academy of Sciences, Director of the Institute of Computer Science of the Academy of Sciences of Tskhum-Abkhazia, Sokhumi State University, Associate Professor
E-mail: nkereselidze@sou.edu.ge

Presented by the Institute of Computer Sciences of the Tskhum-Abkhazian Academy of Sciences

Abstract. An Internet-based application was developed using client-side technologies. The capabilities of modern browsers were used, which in addition to editing capabilities for the programming languages JavaScript, HTML, have built-in Storage for Data. Specifically, storage capabilities are considered LocalStorage. Using browser storages, a Comprehensive System of Testing (CST) has been built. With the help of the CST, the interested person can independently deepen knowledge of individual topics of any subject, or pass the exam in the form of a test, even in the absence of the Internet. CST allows the subject teacher to autonomously create a test on a separate topic, and each time, for example, the examinee displays different test questions in a different sequence. The program code was created using HTML, JavaScript and JSON.

Keywords: *Internet technologies, JavaScript, HTML, JSON, testing system, local storage.*

Introduction. The testing phase is widely used to effectively assimilate the curricula of higher and secondary schools. With the help of Testing training, an interested person can not only master, improve knowledge on the topic, but also pass exams in individual subjects. Today, there are technologies for creating tests and testing students' knowledge with their help, offered by leading organizations working in the digital world, such as Google and others. However, the products offered often have limited testing capabilities and are often expensive. Testing is usually done with server-hosted questions, but an interesting model is when testing is done entirely on the client side. An interesting question related to the introduction of questions for the test. In existing systems, only as many questions are entered for a given topic as determined during testing, which does not allow generating random questions. To expand the number of questions, they are added from the very beginning. Therefore, it would be better if a database of questions were created from the very beginning, and questions for testing were selected from it randomly. When testing, sometimes it is necessary to conduct unified testing on several topics, and the proposed systems do not have such an opportunity. Such a need arises when preparing for intermediate and final exams or when conducting such exams. During the testing training period, the complex testing software system should be able to show the user the correct answers along with the assessment if necessary. It is necessary to provide for a restriction on the conduct of tests in the form of an exam, which should exclude the possibility of re-passing the tests by the same person. CST includes the creation of a database of questions on various topics. The user will be able to pass various testing modes: to study a specific topic - in person; for examination; testing one or more topics.

I. STRUCTURE OF THE COMPREHENSIVE SYSTEM OF TESTING

Structure of the Comprehensive System of Testing (CST) must necessarily reflect the functional purpose of the CST. Thus, as components of a complex system, subsystems should be created that ensure the creation of a single database of test issues and the use of this database to create a specific test, provide a specific test to the user, evaluate the test performed, etc. The structure diagram of the comprehensive system of testing can be represented as follows:

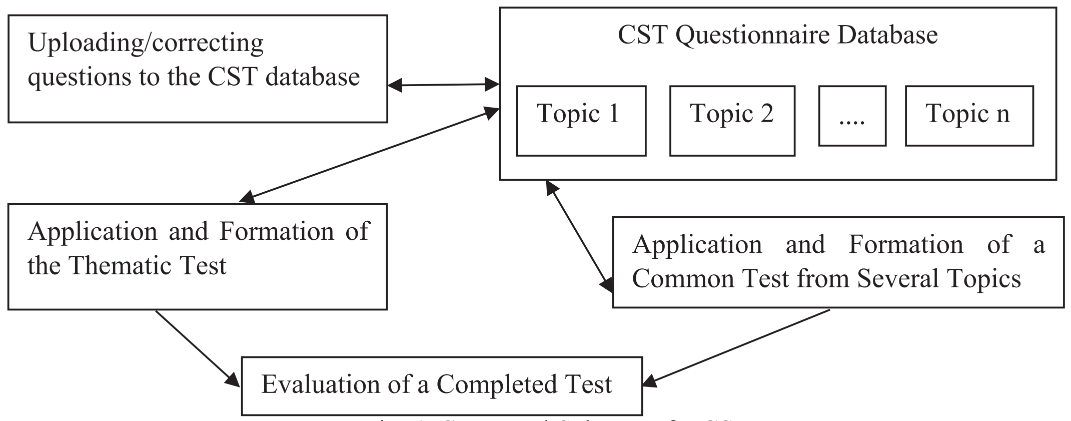

Uploading/correcting questions to the CST database
CST Questionnaire Database
Topic 1
Topic 2
....
Topic n
Application and Formation of the Thematic Test
Application and Formation of a Common Test from Several Topics
Evaluation of a Completed Test

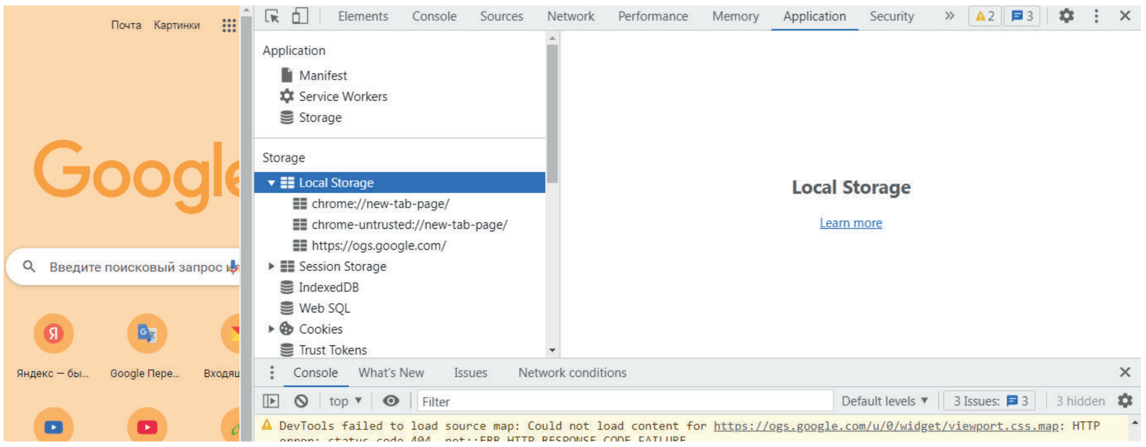
Fig. 1. Structural Scheme of a CST

Let's consider uploading/correcting the questionnaire of the CST and the unified database of questionnaires as a subsystem called: the Subsystem for Uploading Questions for Tests (SUQT). Let's imagine the substructure of thematic test application and formation, its connection with the database of questionnaires as a Subsystem for Forming Questions from the Thematic Test Database (SFQTTD). Let's imagine the substructure of evaluating the completed thematic and cross-thematic test as a Subsystem for Evaluating Test Answers (SETA). Let's imagine the substructure of applying and forming a common test from several topics as two subsystems: as a Subsystem for Midterm Exams from the General Test Database (SMEGTD) and as a Subsystem for Final Exams from the General Test Database (SFEGTD).

II. SUBSYSTEM FOR UPLOADING QUESTIONS FOR TESTS (SUQT)

The main cornerstone of a complex software system for testing is a single database of questions and possible answers necessary for testing. Since the CST should function without an Internet connection to the server, the database required for testing should be located on the client side, namely in the browser. Such a database can be Local Storage or LocalStorage. We will not consider other data storage in the browser - cookies and WebSQL. We will now discuss the possibility of using LocalStorage in CST.

A brief description of LocalStorage. LocalStorage is a non-relational database that is built into the browser. It can store data for a long time - permanently, represented as key:value pairs. An LocalStorage created in one browser will not appear in another browser. To see where LocalStorage is located in the browser, open the developer panel in the browser you are working in. Consider the developer panel for the Chrome browser, which is open in the Windows operating system - for this, use the keyboard shortcut Ctrl + Shift + I (or F12) to enter the panel, which will look like this:



```
DevTools failed to load source map: Could not load content for https://ogs.google.com/u/0/widget/viewport.css.map: HTTP
```

Figure 2. Developer panel, LocalStorage location

In the Application tab of the developer panel, LocalStorage is visible, and there is also a data storage - Local Storage. Here are also cookies and WebSQL. In LocalStorage and Local Storage, data is stored in the form of 'key': 'value', similar to a so-called indexed array.

Based on the DOM principle, LocalStorage is a window element, this can be checked as follows: in the developer panel, we go to the console tab and type the command - console.log(window); - that is, we will display the window list log, where you will see LocalStorage in the list along with other objects. In LocalStorage, data can be written, deleted, and called. We do this using JavaScript. Since LocalStorage is a JavaScript object. It has methods that perform the above operations.

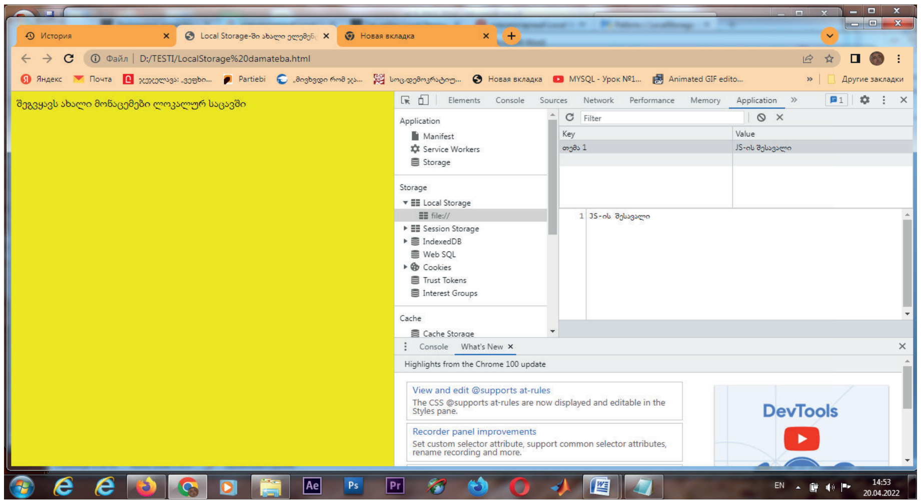
Method **setItem**. This method adds a new element to LocalStorage, which consists of a pair - key:value. For example

```
localStorage.setItem("თემა 1","JS-ის შესავალი");
```

Listing 1. Adding a new element to LocalStorage

```
<HTML>
<HEAD>
<TITLE>CTS-1</title>
<SCRIPT LANGUAGE="JavaScript">
localStorage.setItem("თემა 1","JS-ის შესავალი");
document.write("შეგვყავს ახალი მონაცემები ლოკალურ
საცავში");
</script>
</head>
<body bgcolor=yellow>
</body>
</html>
```

After implementing Listing 1, new data will be added to LocalStorage



შეგვყავს ახალი მონაცემები ლოკალურ საცავში

Key	Value
თემა 1	JS-ის შესავალი

Figure 3. Adding new data to LocalStorage

We can automate data entry, if there are, for example, 5 topics with their names to enter, we can put it in a loop:

Listing 2. Entering topic numbers and their names

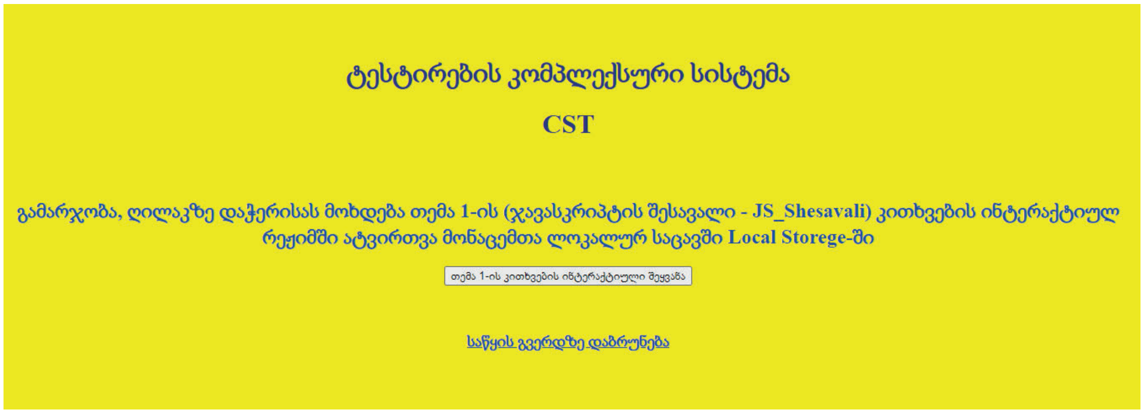
```
<HTML>
<HEAD>
<TITLE> Local Storage-ში ახალი ელემენტის დამატება </title>
<SCRIPT LANGUAGE="JavaScript">
for (i=1; i<6; i++){
document.write("შეგვყავს ახალი მონაცემები Local Storage
ლოკალურ საცავში - თემა "+i+" და მისი დასახელება"+"<br>");
a="თემა "+ i;
b=prompt("შეიყვანეთ "+i+"-ური თემის დასახელება");
localStorage.setItem(a,b);
}
</script>
</head>
<body bgcolor=yellow>
</body>
</html>
```

In the data store, we need to record the test questions, the possible answers and highlight the correct answer. This information will be recorded for a long time and will be stored on the client side/browser, even after the computer is turned off from the power source.

The operator should enter the test information and write it to localStorage, he can also add or change questions, answers, topics if necessary.

It is logical to encode the test information in JSON format in the SCRIPT section of the HTML code. Obviously, a separate JSON should be encoded for each topic. Since the information in JSON format needs to be transferred to localStorage, it should be converted to string format accordingly - for which we should use the appropriate method - JSON.stringify(). In the future, when we need to download the test information from localStorage, which is stored in string format, we should convert it to JSON format - for which we should use the appropriate method JSON.parse();

Writing data in JSON format is a time-consuming task and it is desirable to automate it so that anyone interested can use it. The relevant code for uploading the test questions and answers for Topic 1 can be found here [1]. As a result of implementing this code, the window shown in Figure 4 will appear and Test 1 will be uploaded interactively.


ტესტირების კომპლექსური სისტემა

CST

გამარჯობა, ღილაკზე დაჭერისას მოხდება თემა 1-ის (ჯავასკრიპტის შესავალი - JS_Shesavali) კითხვების ინტერაქტიულ რეჟიმში ატვირთვა მონაცემთა ლოკალურ საცავში Local Storege-ში

Figure 4. Adding new data to LocalStorage

For example, in CST we will have 5 topics and the interactive completion code for each topic is given in [2-5] respectively. Thus, in principle, a subsystem for uploading questions for tests has been created. In particular, in the client-side data storage - localStorage, it is possible to enter the name of the topic and the questions corresponding to this topic, the estimated and correct answers. The questions, estimated and correct answers are written to the localStorage storage in the form of a string type, but its formation is carried out in JSON format. Then the questions, estimated and correct answers written in JSON format are converted into a string - a string variable and only then are they loaded into localStorage in this form.

III. SUBSYSTEMS FOR GENERATING QUESTIONS FROM THEMATIC TEST DATABASE (SFQTTD AND EVALUATING TEST ANSWERS (SETA

After we have recorded several topics for testing and their corresponding questions, probable and correct answers in localStorage, it is necessary to fill out a test/quiz within a specific topic. But before that, we need to select questions for this test. As we remember, our goal is to create a specific test from a certain number of questions at random each time of testing. Let's set a goal to select several different questions each time from the set of questions, probable and correct answers corresponding to a specific topic. For example, we have recorded 15 questions in Topic1, with their answers, to prepare a test consisting of five questions for the interested person to check, and in addition, in each created test these five questions should be different, in a different order.

Let's define the solution to this task as follows: first - download the questions and answers of the topic of interest, in our case Topic1, from localStorage; Second - let's create a five-element random array of integers, where each number is from 0 to 15 (inclusive). Third - with the help of this random array, we will extract five questions from the downloaded topic 1 with their answers.

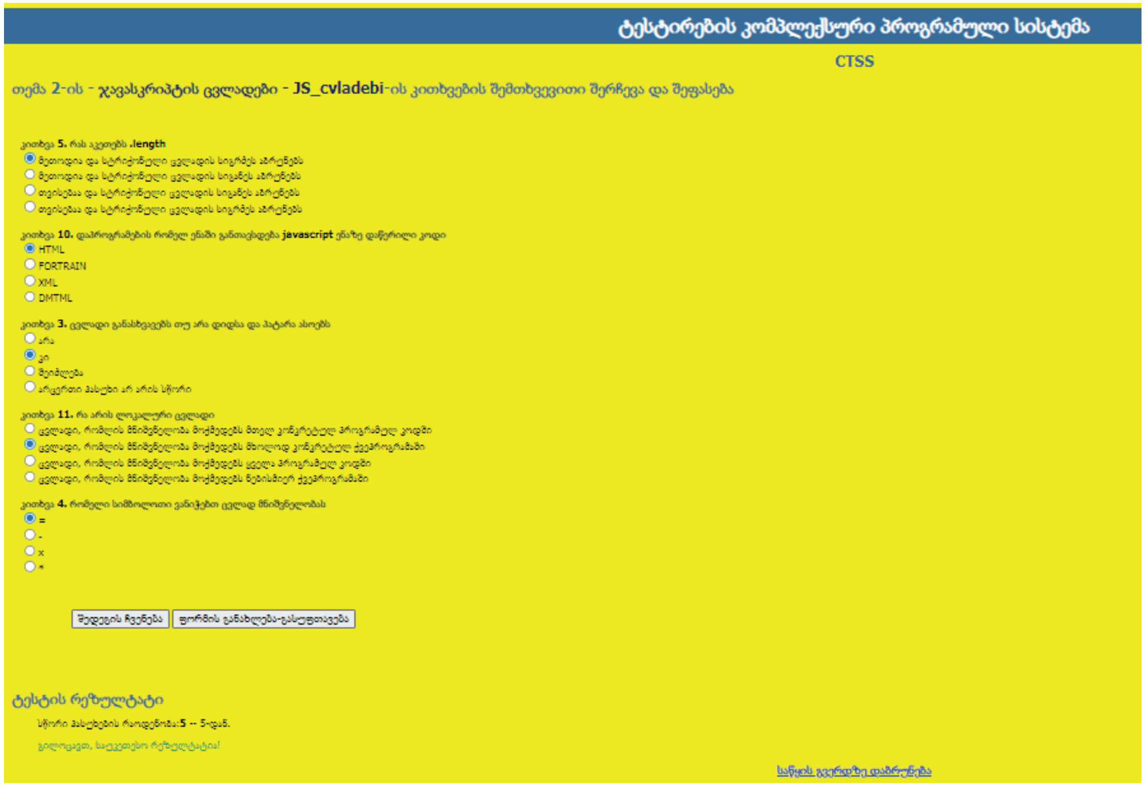
The first issue is solved with one command - get the value corresponding to the key of topic 1 and write it in JSON format to some variable, say - gadmowerili. In code, this will be the following command:

var gadmowerili= JSON.parse(localStorage.getItem("Topic 1"));

In this command, first the value corresponding to the key of topic 1 is retrieved from the local storage using localStorage.getItem, then this value is converted to JSON format and given the identifier gadmowerili. After that, we can access the gadmowerili JSON object, for example, if we want to display the third question of topic 1 on the monitor, then we get the following statement:

document.write("Third question i=", gadmowerili.JS_Shesavali[2].shekiTxva);

The codes for randomly selecting questions and evaluating the answers given in the test for all five topics are given accordingly [6-10]. Figure 5 illustrates the randomization of questions for testing Topic 2 and the scoring of the answers provided.



ტესტირების კომპლექსური პროგრამული სისტემა

CTSS

თემა 2-ის - ჯავასკრიპტის ცვლადები - JS_cvladebi-ის კითხვების შემთხვევითი შერჩევა და შეფასება

კითხვა **5.** რას აკეთებს **.length**
- [✓ მეთოდია და სტრიქონული ცვლადის სიგრძეს აბრუნებს]
- მეთოდია და სტრიქონული ცვლადის სიგანეს აბრუნებს
- თვისებაა და სტრიქონული ცვლადის სიგანეს აბრუნებს
- თვისებაა და სტრიქონული ცვლადის სიგრძეს აბრუნებს

კითხვა **10.** დაპროგრამების რომელ ენაში განთავსდება **javascript** ენაზე დაწერილი კოდი
- [✓ HTML]
- FORTRAIN
- XML
- DMTML

კითხვა **3.** ცვლადი განასხვავებს თუ არა დიდსა და პატარა ასოებს
- არა
- [✓ კი]
- შეიძლება
- არცერთი პასუხი არ არის სწორი

კითხვა **11.** რა არის ლოკალური ცვლადი
- ცვლადი, რომლის მნიშვნელობა მოქმედებს მთელ კონკრეტულ პროგრამულ კოდში
- [✓ ცვლადი, რომლის მნიშვნელობა მოქმედებს მხოლოდ კონკრეტულ ქვეპროგრამაში]
- ცვლადი, რომლის მნიშვნელობა მოქმედებს ყველა პროგრამულ კოდში
- ცვლადი, რომლის მნიშვნელობა მოქმედებს ნებისმიერ ქვეპროგრამაში

კითხვა **4.** რომელი სიმბოლოთი ვანიჭებთ ცვლადს მნიშვნელობას
- [✓ =]
- -
- x
- *

ტესტის რეზულტატი

სწორი პასუხების რაოდენობა:**5** -- 5-დან.

გილოცავთ, საუკეთესო რეზულტატია!

Figure 5. Randomly generating questions for testing topic 2 and evaluating the answers given

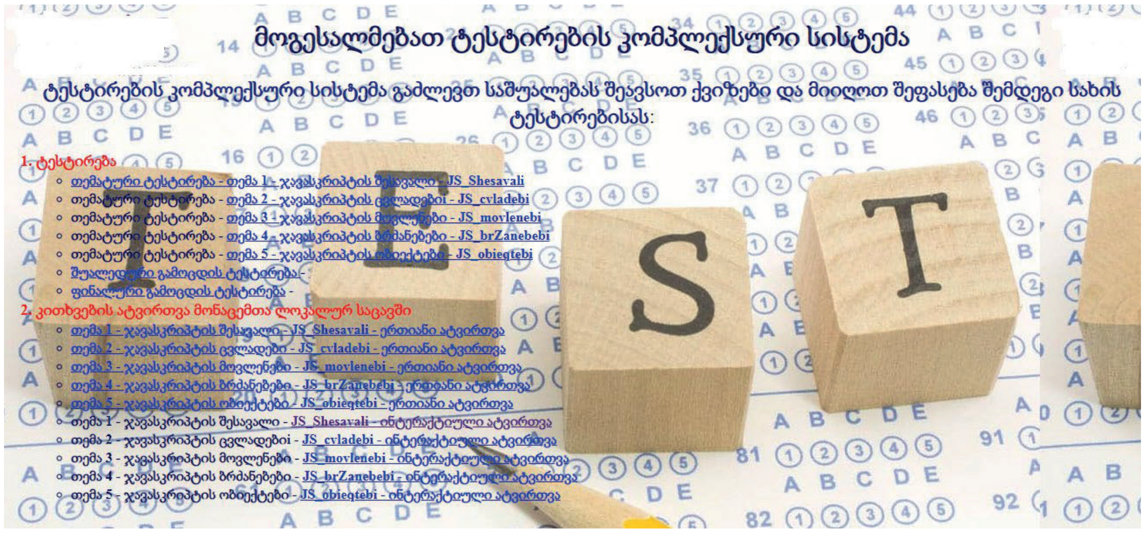
IV. SUBSYSTEM FOR FORMING AND EVALUATING MIDTERM (SMEGTD AND FINAL EXAM QUESTIONS FROM THE GENERAL TEST DATABASE (SFEGTD)

Before building a subsystem for forming and evaluating midterm exam questions from the general test base, let's consider the principles of selecting questions for the midterm exam. As a rule, the midterm exam takes place after the eighth week of study, that is, when, say, 8 lectures out of 15 lectures have been taught, that is, more than half of the teaching material. Taking this principle into account, in our case, since we have chosen the number of topics equal to five, then let's assume that the midterm exam in the TKP includes an assessment of the mastery of the first three topics in the form of a test. If the number of midterm exam questions is 30, then we select ten questions for the assessment of each topic, naturally by a random principle. Thus, we can, taking into account the results of Chapters 2 and 3, present the principles of building a subsystem for forming and evaluating midterm exam questions. First of all, we should randomly select ten questions from each of the first three topics. Therefore, for the first topic, we will create a ten-element array of output question numbers - massTem_1[], similarly, for the second topic, we will create a ten-element array of output question numbers - massTem_2[], and for the third topic, we will create a ten-element array of output question numbers - massTem_3[]. As a result, we will get three arrays: massTem_1[], massTem_2[], massTem_3[], which contain 10 non-repeating random numbers from 0 to 14. After that, we need to select the corresponding questions from each topic in this array. Thus, for the midterm exam, the specific options for questions, probable and correct answers are in six combined arrays, and from these, we need to extract the first five in the form of a form. To do this, we will use the experience that we accumulated in Chapter 3 - when forming quizzes on individual topics. The code for displaying the midterm exam quiz and evaluating the answers is here [11].

For the construction of a subsystem for the formation and evaluation of questions for the final exam from the general test base, it is essential to clarify the following issue - by what principle will the questions for the final exam be selected? Should we select the same number of questions from all topics? If, taking into account that the first topics (in our case, the first three topics) were already used in the midterm exam, we select a smaller number of questions from them, and the share of more questions should be distributed to the remaining topics. Since this is a more methodical type of question, we will consider the case when we select an equal number of questions from all topics for the final exam - that is, eight questions from five topics.

To solve this problem, we will use the experience of the subsystem for the formation and evaluation of questions for the midterm exam from the general test base and adapt these subsystems. In particular, we should use questions from all topics. The code for the subsystem for forming and evaluating final exam questions is located here [12].

Let's integrate the subsystems into a complex testing system and build a local website. The first page of which has the code that can be found here [13]. From the home page of the website, we can navigate to the pages that actually represent a specific implementation of a certain subsystem of the project. The interface of the home page is given in Fig. 6.


მოგესალმებათ ტესტირების კომპლექსური სისტემა

ტესტირების კომპლექსური სისტემა გაძლევთ საშუალებას შეავსოთ ქვიზები და მიიღოთ შეფასება შემდეგი სახის ტესტირებისას:

1. ტესტირება
   - თემატური ტესტირება - თემა 1 - ჯავასკრიპტის შესავალი - JS_Shesavali
   - თემატური ტესტირება - თემა 2 - ჯავასკრიპტის ცვლადებi - JS_cvladebi
   - თემატური ტესტირება - თემა 3 - ჯავასკრიპტის მოვლენები - JS_movlenebi
   - თემატური ტესტირება - თემა 4 - ჯავასკრიპტის ბრძანებები - JS_brZanebebi
   - თემატური ტესტირება - თემა 5 - ჯავასკრიპტის ობიექტები - JS_obieqtebi
   - შუალედური გამოცდის ტესტირება -
   - ფინალური გამოცდის ტესტირება -
2. კითხვების ატვირთვა მონაცემთა ლოკალურ საცავში
   - თემა 1 - ჯავასკრიპტის შესავალი - JS_Shesavali - ერთიანი ატვირთვა
   - თემა 2 - ჯავასკრიპტის ცვლადები - JS_cvladebi - ერთიანი ატვირთვა
   - თემა 3 - ჯავასკრიპტის მოვლენები - JS_movlenebi - ერთიანი ატვირთვა
   - თემა 4 - ჯავასკრიპტის ბრძანებები - JS_brZanebebi - ერთიანი ატვირთვა
   - თემა 5 - ჯავასკრიპტის ობიექტები - JS_obieqtebi - ერთიანი ატვირთვა
   - თემა 1 - ჯავასკრიპტის შესავალი - JS_Shesavali - ინტერაქტიული ატვირთვა
   - თემა 2 - ჯავასკრიპტის ცვლადებi - JS_cvladebi - ინტერაქტიული ატვირთვა
   - თემა 3 - ჯავასკრიპტის მოვლენები - JS_movlenebi - ინტერაქტიული ატვირთვა
   - თემა 4 - ჯავასკრიპტის ბრძანებები - JS_brZanebebi - ინტერაქტიული ატვირთვა
   - თემა 5 - ჯავასკრიპტის ობიექტები - JS_obieqtebi - ინტერაქტიული ატვირთვა

Fig. 6. Complex testing system home page interface

Conclusion. Thus, a "complex testing system" (**CST**) has been built. As a result, a local site has been obtained, through which various stages of testing are presented. At the initial stage of creating a "complex testing system", the task was presented as a set of several subsystems.

Client programming languages - **HTML, Javascript, CSS** were used when creating the CST software code. The questions intended for testing, their probable answers, including the correct answer, are placed in the local data storage - Local Storage. Testing is carried out with a radio-type switch, that is, only one is selected from several probable answers given to the question. It is possible to name topics and place their corresponding questions and probable answers in the local storage. Their placement is done in two ways: a) the user personally enters data in an interactive mode and b) the data is placed in the software code in **JSON** format and then, by pressing one button, this data is uploaded to the local data storage. The number of questions stored in the local storage is much larger than the number of questions offered during testing. Each time during thematic testing, the questions offered are randomly generated; after the user selects the answers and uploads them, the work is automatically and instantly evaluated and the result is displayed on the testing website. In addition to thematic testing, it is possible to conduct testing for midterm and final exams with the same random questions and automatic evaluation. A unified interface has been created, which makes it easy to switch to a specific form of testing.

References:

1. http://kereseli.sou.edu.ge/CST/tema%201-is%20kiTxvebis%20interaqtiuli%20damateba.txt
2. http://kereseli.sou.edu.ge/CST/tema%202-is%20kiTxvebis%20interaqtiuli%20damateba.txt
3. http://kereseli.sou.edu.ge/CST/tema%203-is%20kiTxvebis%20interaqtiuli%20damateba.txt
4. http://kereseli.sou.edu.ge/CST/tema%204-is%20kiTxvebis%20interaqtiuli%20damateba.txt
5. http://kereseli.sou.edu.ge/CST/tema%205-is%20kiTxvebis%20interaqtiuli%20damateba.txt
6. https://kereseli.sou.edu.ge/CST/Tema_1_is%20testi.txt
7. https://kereseli.sou.edu.ge/CST/Tema_2_is%20testi.txt
8. https://kereseli.sou.edu.ge/CST/Tema_3_is%20testi.txt
9. https://kereseli.sou.edu.ge/CST/Tema_4_is%20testi.txt
10. https://kereseli.sou.edu.ge/CST/Tema_5_is%20testi.txt

11. https://kereseli.sou.edu.ge/CST/shualeduri%20gamocdis%20testi.txt
12. https://kereseli.sou.edu.ge/CST/finaluri%20gamocdis%20testi.txt
13. https://kereseli.sou.edu.ge/CST/finaluri%20gamocdis%20testi.txt

კლიენტის მხარის ინტერნეტ ტექნოლოგიები განათლებაში

ელზა იმნაძე სოხუმის სახელმწიფო უნივერსიტეტის საბუნებისმეტყველო მეცნიერებების, მათემატიკის, ტექნოლოგიისა და ფარმაციის ფაკულტეტის დოქტორანტი, კომპიუტერული მეცნიერებების პროგრამა; ილიას სახელმწიფო უნივერსიტეტი, საინფორმაციო ტექნოლოგიების ოფისის უფროსი სპეციალისტი

E-mail: elza.imnadze@iliauni.edu.ge

ნუგზარ კერესელიძე ინფორმატიკის დოქტორი, ცხუმ-აფხაზეთის მეცნიერებათა აკადემიის წევრ-კორესპონდენტი, ცხუმ-აფხაზეთის მეცნიერებათა აკადემიის კომპიუტერულ მეცნიერებათა ინსტიტუტის დირექტორი, სოხუმის სახელმწიფო უნივერსიტეტი, ასოცირებული პროფესორი

E-mail: nkereselidze@sou.edu.ge

წარმოადგინა ცხუმ-აფხაზეთის მეცნიერებათა აკადემიის კომპიუტერულ მეცნიერებათა ინსტიტუტმა

აბსტრაქტი. შემოთავაზებული ინტერნეტ ტექნოლოგიების აპლიკაცია შეიქმნა კლიენტის მხარის შესაძლებლობების საფუძველზე. გამოყენებული იქნა თანამედროვე ბრაუზერების შესაძლებლობები, რომლებსაც JavaScript-ის, HTML-ის პროგრამირების ენების რედაქტირების შესაძლებლობების გარდა, აქვთ ჩაშენებული მონაცემთა შენახვის ფუნქცია. კერძოდ, შენახვის შესაძლებლობები განიხილება როგორც Local Storage . ბრაუზერის საცავების გამოყენებით, შექმნილია ტესტირების ყოვლისმომცველი სისტემა (CST). CST-ის დახმარებით, დაინტერესებულ პირს შეუძლია დამოუკიდებლად გაიღრმავოს ცოდნა ნებისმიერი საგნის ცალკეულ თემებზე, ან ჩააბაროს გამოცდა ტესტის სახით, ინტერნეტის

არარსებობის შემთხვევაშიც კი. CST საშუალებას აძლევს საგნის მასწავლებელს დამოუკიდებლად შექმნას ტესტი ცალკეულ თემაზე და ყოველ ჯერზე, მაგალითად, გამოსაცდელს აჩვენოს სხვადასხვა ტესტის კითხვები განსხვავებული თანმიმდევრობით. შექმნილია პროგრამული კოდი HTML, JavaScript და JSON-ზე დაყრდნობით.

საკვანძო სიტყვები: *ინტერნეტ ტექნოლოგიები, JavaScript, HTML, JSON, ტესტირების სისტემა, ლოკალური საცავი.*