TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Alexander Dan Håkan Fridlund 155223IAPB

SECURE LIGHTWEIGHT BROWSER-BASED TEST SYSTEM

Bachelor's thesis

Supervisor: Jaak Henno
PhD.math

Tallinn 2020

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Alexander Dan Håkan Fridlund 155223IAPB

TURVALINE KERGEKAALULINE BRAUSERIPÕHINE TESTIDE TEGEMISE SÜSTEEM

Bakalaureusetöö

Juhendaja: Jaak Henno
PhD.math

Tallinn 2020

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Alexander Dan Håkan Fridlund

02.08.2020

Abstract

The objective of this thesis is to create testing system, which helps teacher in quick assessment of students. As an extension to increase user's involvement is created encryption-based chat system.

The web applications functionality is implemented using Hypertext Preprocessor (PHP) and JavaScript (JS) scripts. Web page itself is written in HTML (HyperText Markup Language) using CSS (Cascading Style Sheets). System is called *lightweight* because

1. teacher can upload, delete tests and get student's results directly from browser and does not need to handle server directly;

2. implementation does not use libraries for design or JavaScript and HTML interaction, there is no 'dead code' (code from libraries which is not used) thus system loads very quickly (Figure 16).

The result of this work is a working web application, where teacher can add/delete tests and get student's test results from browser. Students can do the tests and play a Rock-Paper-Scissors game with opportunity of chatting with opponent.

This thesis is written in English and is 43 pages long, including 6 chapters, 46 figures and 1 table.

Annotatsioon

Brauseripõhine testsüsteem, mis aitab õpetajat õpilaste kiire hindamisega

Selle töö eesmärgiks on luua testsüsteemi, mis aitab õpetajat õpilaste kiire hindamisega. Süsteemi teeb õpilastele huvitavamaks võimalus mängida mänge ja kasutada krüpteerimispõhist vestlussüsteemi.

Veebirakendus on kirjutatud kasutades Hypertext Preprocessor (PHP) and JavaScript (JS). Veebileht ise on kirjutatud HTML (HyperText Markup Language) abil, kasutades CSS (Cascading Style Sheets). Süsteemi nimetatakse *kergekaaluliseks* sellepärast, et

1. õppejõud saab üles laadida, eemaldada teste ja näha tudengite tulemusi otse brauserist ja ta ei pea tegelema serveriga otse;
2. süsteemi realisatsioonis ei kasutata teeke (libraries) kujundamiseks või JavaScripti ja HTML interaktsiooni jaoks, seega täielikult puudub ’surnud kood’ (teekide kood, mida tegelikult ei kasutata) seega süsteemi käivitamine on väga kiire (Figure 16).

Töö tulemuseks on veebirakendus, kus õpetajad saavad lisada teste ja saada õpilaste tulemusi. Õpilased saavad teha teste ja mängida Kivi-Paber-Käärid mängu ja võib vestelda oponentidega.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 43 leheküljel, 6 peatükki, 46 joonist, 1 tabelit.

List of abbreviations and terms

AES-128	Encryption algorithm
AJAX	Asynchronous Javascript and XML
Bootstrap	Framework for building sites
CSS	Cascading Style Sheets
Firebase	Development platform https://firebase.google.com/
Hot Potatoes	Free application for creating web exercises https://hotpot.uvic.ca/
HTML	HyperText Markup Language
JS	JavaScript
jQuery	JavaScript library
MySQL	Relational database
PHP	Hypertext Preprocessor
SJCL	Stanford Javascript Crypto Library http://bitwiseshiftleft.github.io/sjcl/
SQL injection	Placement of malicious code in SQL statements, via web page input
Unsplash	Website with stock photographs https://unsplash.com/

Table of contents

List of figures

List of tables

1 Introduction

Assessment is an important part of learning process. Mostly teachers check student work manually. Automation of this process can save time for teachers. Also, it would save time for students, who are waiting for their results. System, which helps teacher with assessment can make learning process more comfortable for those who study and those who teach as well.

The objective of this thesis is to simplify student assessment for teacher. After teacher created test, system should provide possibility to show this test in web browser and receive results from it. To increase attractiveness of the system for students is implemented possibility to play multiplayer game and use the chat system, which provides opportunity of communication for users during the game. Communication between users are encrypted using a symmetric encryption system and the encryption key would be generated locally, using randomness generated by game players.

2 Why create a testing system?

Nowadays, when many governments have closed schools and universities, studying process must go on [1]. There are a lot of applications and systems to create tests [2] [16]. Unfortunately, some of them are giving only free trial or not giving opportunity to embed created test and result of it into your own website. If test is not embedded, teacher should somehow provide it to students. If teacher don't have own website, gathering this information after school closes can be difficult. Also, it is comfortable for teacher if test and test result are on same webpage. That is the reason why tests should be embedded to webpage.

System can be called *secure*, because it uses cloud to store results [17].

Most online games include communication between players [3]. Messaging for players should be encrypted [4]. Chat with symmetric encryption is a solution. Since encryption keys for messaging could be created locally using the randomness created by players in gameplay. *Bring Your Own Key* model is very suitable. This model allows users to manage keys for encryption [5].

Goal of this thesis is to create a system, which provide:

- Uploading tests
- Participating in tests
- Saving the results of the test participants
- Ordering the results of the test participants in scoreboard (users, who receive better grades are first in the result table)
- Playing rock-paper-scissors game with another player with possibility of secure communication

2.1 Differences from other popular web testing systems

If many systems are already existing, why should new system be created? One popular game-based learning platform is “Kahoot!” [18]. This platform allows creating quizzes and sharing them. It seems that “Kahoot!” achieve needed goals: teacher have test which is not needed to be checked manually. Unfortunately, it has a problem. After quiz is created, game pin or link is generated for it. So, this pin must be somehow be distributed to students. Also, question for quiz can have maximum 4 answers. Moreover, quiz and results of this quiz cannot be embedded to another website. This platform is not multipurpose. For example, it is impossible to implement chat system mentioned in 1.

“Strawpoll” is another system for creating quizzes [19]. It also provides creating quizzes and sharing them with participants. Unfortunately, each *poll* has only 1 question, which makes this system very uncomfortable for test creation. Like a “Kahoot!”, link to the quiz must be distributed to students and chat system or game is impossible to implement.

Created system allows uploading test directly from browser. Test participants do not need a pin or link to start a test, they would see all available tests. Results of the test can also be seen in browser. System also provide Rock-Paper-Scissors game with encrypted chat with opponent.

3 System

System provide registration, logging in with authentication and use without authentication. System is written in HTML5, Hypertext Preprocessor (PHP) and JavaScript (JS). Asynchronous JavaScript and XML(AJAX) is used for dynamic change of content on the webpage without reloading the whole page.

3.1 Logging without authentication

If user do not want to log in with created account or user do not have account and do not want to create it, user can enter system as “Guest”. “Guest” user can see welcoming message (Figure 1).

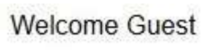

Welcome Guest

Figure 1. Guest
user screen

Guest can also see bar with available functionality (Figure 2). Guest user can participate only in one test and results of this test are not saved.

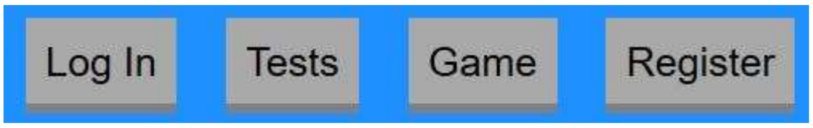
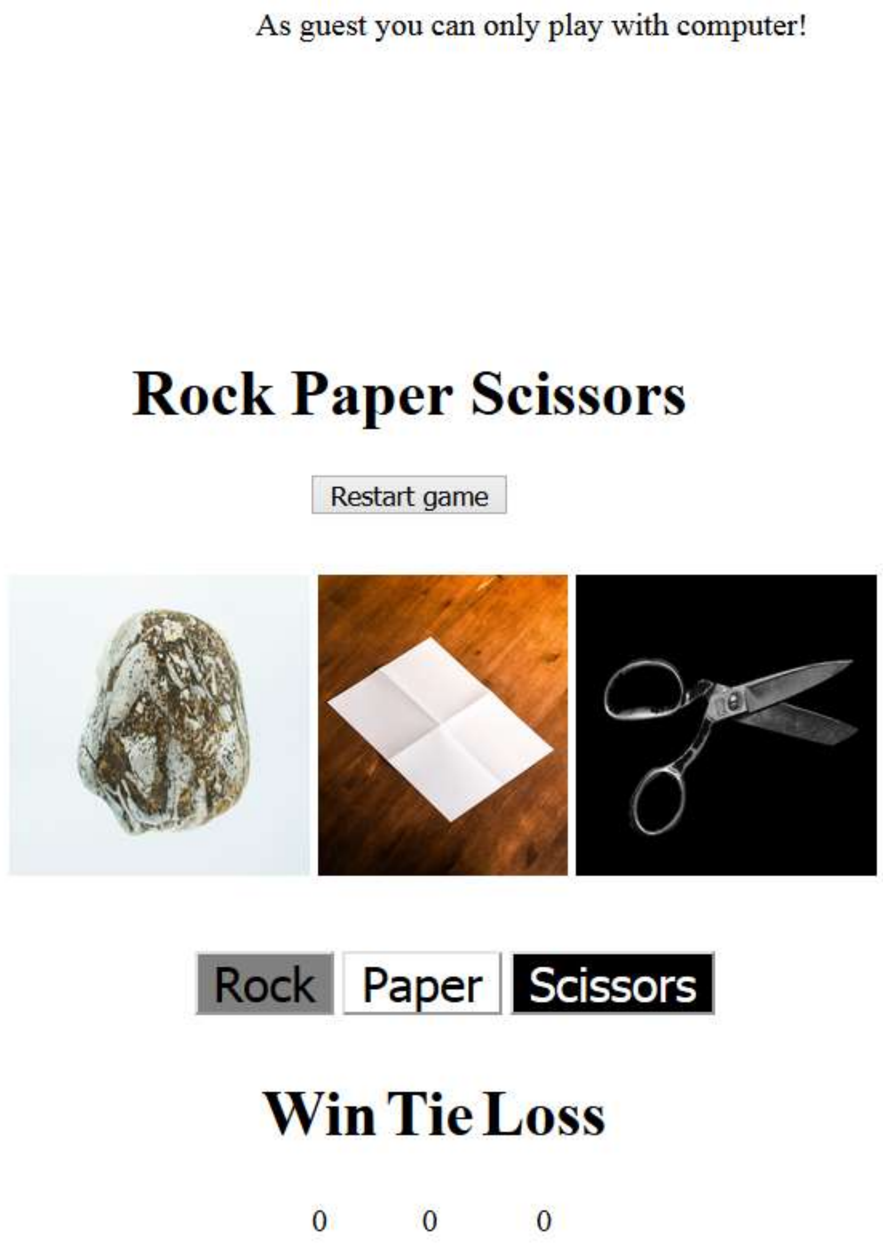


Figure 2. Guest user bar

After pressing button “Game”, game would be started with computer, because guest user cannot play game with other players (Figure 3).


As guest you can only play with computer!

Rock Paper Scissors

Win Tie Loss

0 0 0

Figure 3. Guest user game

3.2 Registration

To receive full functionality, user need to register in system. As seen in Figure 4, to create account user must choose username and create password.

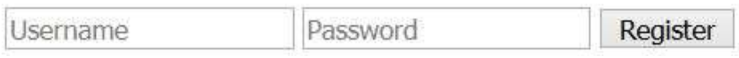


Figure 4. Registration form

System uses MySQL database to store and retrieve user data. So, when user press „Register” button new row in table is created, if username is unique. Otherwise, user can see error message above the registration form (Figure 5).

Form with text field, which is sending to server might be dangerous to database because

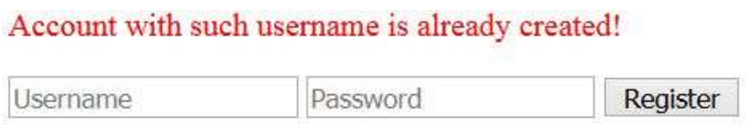

Account with such username is already created!

Figure 5. Error message, when creating account with already engaged username

SQL injection can be used. To prevent SQL injection in MySQL table function *mysql_real_escape_string()* is used. This function helps avoiding special characters [6].

3.3 Logging with authentication

When registered user is logging in to system, request to server is sent. Comparison of entered data and data stored on server is happening. If user is registered and password is correct, log in procedure is finished. Otherwise, user can see an error message (Figure 6).

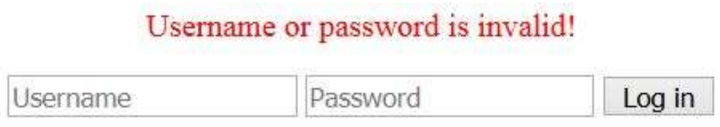
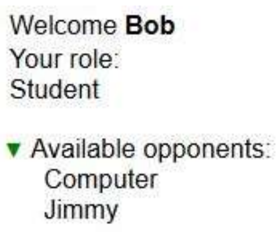

Username or password is invalid!

Figure 6. Incorrectly entered data

As was mentioned, this user send data to server via this form. Similarly to registration form (Figure 4), SQL injection should be prevented here. In order to prevent it function *mysql_real_escape_string()* is used [6].

After authentication user can see welcoming message, own role and list of users, who are currently online (Figure 7).



Welcome **Bob**
Your role:
Student

▼ Available opponents:
Computer
Jimmy

Figure 7. Authenticated
user screen

Like guest user, authenticated user can see a menu bar with available functionality. In contrast with guest user, authenticated user receives more functionality (Figure 8).

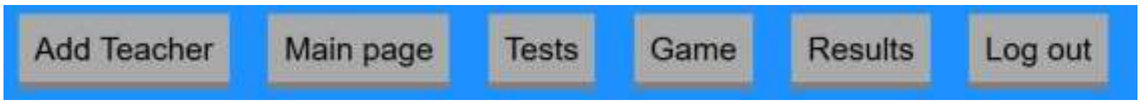


Figure 8. Authenticated user bar(teacher)

3.4 User activities

There are two types of registered users in system: *Students* and *Teachers*.

All available functionality for “Student” user can be seen in Figure 9. Students can participate in tests (choose suitable test and press button “Start test” (Figure 22), which makes frame with suitable test visible (Figure 20)), play a game with computer or other player (5.1), see results of done tests (4.4).

All available functionality for “Teacher” user can be seen in Figure 8. Teachers can use the same opportunities as students. Some functions are only available for teachers, such as role switching (3.5), test adding (4.2), test deleting (4.5).

3.5 Role switching

After registration user role is “Student”. Role can be switched to “Teacher”. For switching, “Add Teacher” button should be pressed (Figure 8). Students cannot switch their roles to “Teacher”. They do not have “Add Teacher” button (Figure 9).

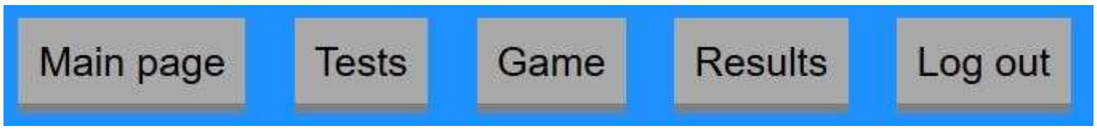


Figure 9. Authenticated user bar(student)

Only another teacher can perform this action. Teacher can enter username of a person who needs a teacher role into a text field (Figure 10).

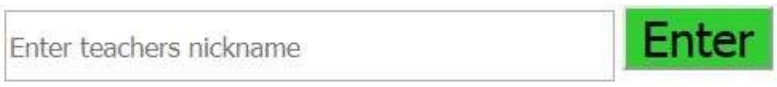


Figure 10. Adding new teacher

If entered username is wrong, alert message would be received. Message can be seen in Figure 11.

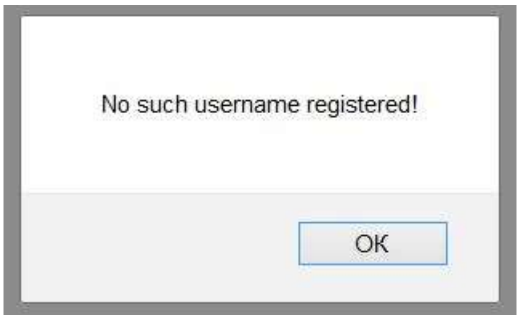

No such username registered!

Figure 11. Alert message, when incorrect username is entered

Otherwise, alert message of successful switching will be shown (Figure 12).

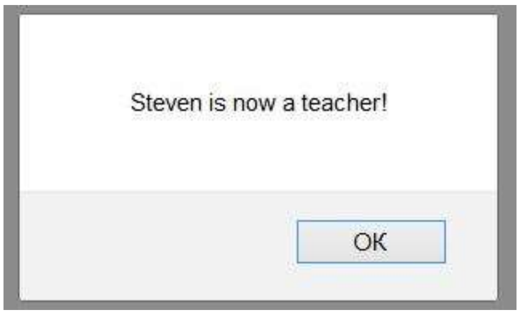

Steven is now a teacher!

Figure 12. Message of successful role switching

3.6 Logging out

When authenticated user finished participating in test or playing a game, “Log out” button from the bar (Figure 8) should be pressed. After button is pressed, session is destroyed and authenticated user become guest user in system. Moreover, user “disappears” from *Available opponents* list, so other users after refreshing page, can mention that user is not currently online. Also, all messages, which were sent by user are deleted after logging out.

3.7 Communication with MySQL database

As was stated in 3.2, system uses MySQL database. User send requests and receive responses from database, most of them during the game. For example, switching role to “Teacher”. Firstly, request with data is sent to PHP file (Figure 13).

```
var teacher = document.getElementById("adding").value.toString();
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        console.log(this.response);
    }
};
xhttp.responseType = 'json';
xhttp.open("POST", "add.php", true);
xhttp.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
xhttp.send("teacher=" + teacher);
```

Figure 13. Request sending

Secondly, query to server is made (Figure 14).

```
<?php
      include 'db.php';
    $teacher = mysqli_real_escape_string($connection, $_POST['teacher']);
$sql1 = "SELECT username FROM t155223_testing2 WHERE username = '$teacher'";
$server = $connection->query($sql1);
$teacherArray = Array();
$answer=mysqli_fetch_array($server);
if ($server = $connection->query($sql1)) {
        echo "";
    }
if ($server->num_rows < 1) {

} else {
    $sql = "UPDATE t155223_testing2 SET role = 'Teacher'
      WHERE username = '$teacher'";
    $server1 = $connection->query($sql);
    $teacherArray[] =  $answer['username'];
}
      echo json_encode($teacherArray);
?>
```

Figure 14. Query to server

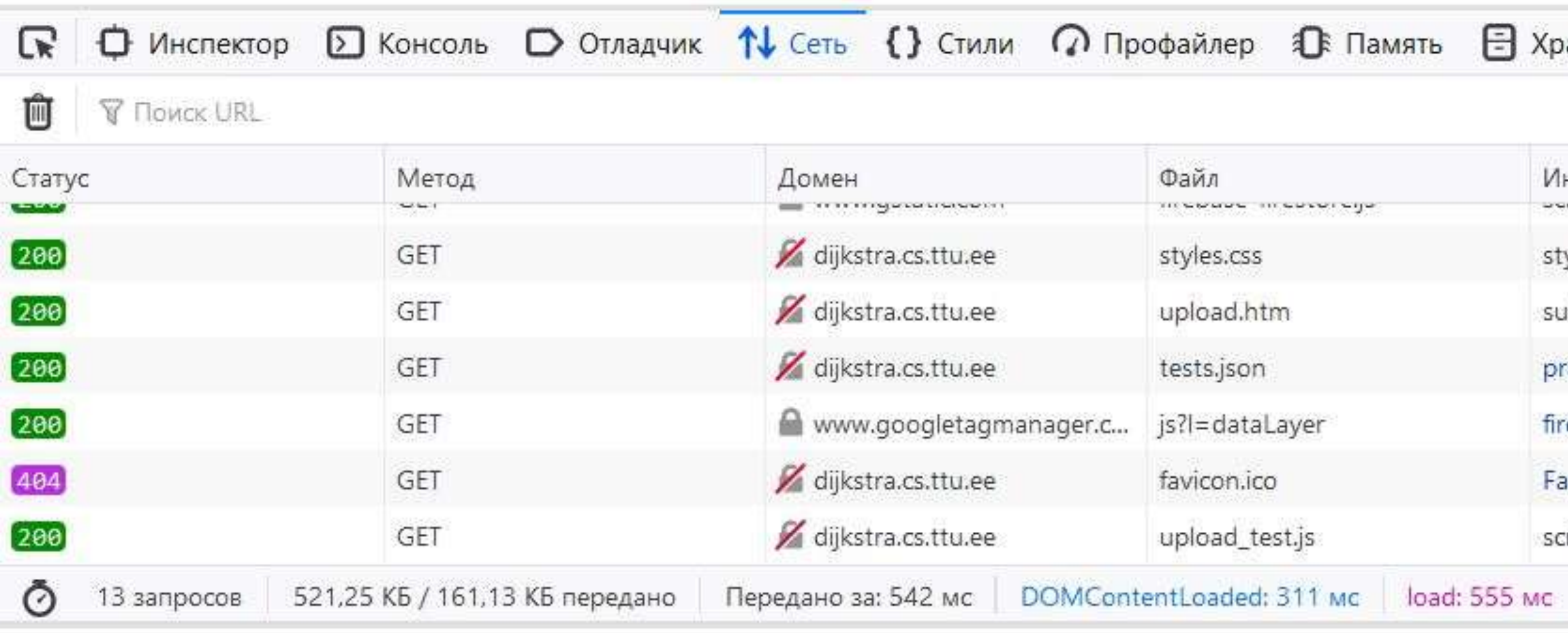
Finally, user is getting response (Figure 15).

```
xhttp.onload = function(data) {
                var obj = xhttp.response;
                    if (obj.length == 0) {
                        alert("No such username registered!");
                    } else {
                    alert(obj[0] + " is now a teacher!");
                    }
            };
}
```

Figure 15. Response from server

3.8 Implementation

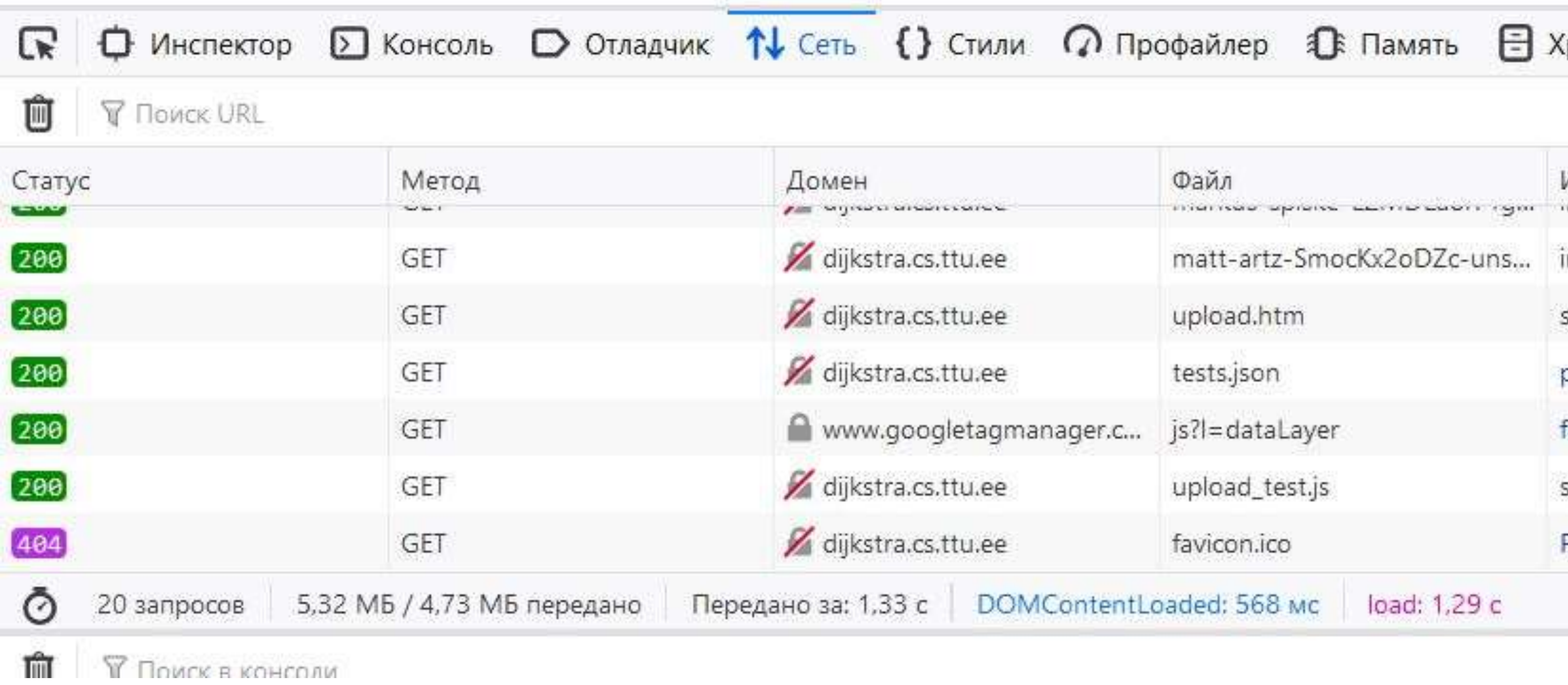
Libraries for design or JavaScript and HTML interaction were not used in implementation of this system. As the system was named *lightweight*, that imply for its performance. For example, *Bootstrap 4* and *jQuery* use, makes system load slower in almost 2.5 times (Figure 16 and Figure 17).



Статус	Метод	Домен	Файл	Ин
200	GET	dijkstra.cs.ttu.ee	styles.css	sty
200	GET	dijkstra.cs.ttu.ee	upload.htm	su
200	GET	dijkstra.cs.ttu.ee	tests.json	pr
200	GET	www.googletagmanager.c...	js?l=dataLayer	fir
404	GET	dijkstra.cs.ttu.ee	favicon.ico	Fa
200	GET	dijkstra.cs.ttu.ee	upload_test.js	sc

13 запросов | 521,25 КБ / 161,13 КБ передано | Передано за: 542 мс | DOMContentLoaded: 311 мс | load: 555 мс

Figure 16. System without Bootstrap and jQuery


Инспектор Консоль Отладчик Сеть Стили Профайлер Память

Поиск URL

Статус	Метод	Домен	Файл
200	GET	dijkstra.cs.ttu.ee	matt-artz-SmocKx2oDZc-uns...
200	GET	dijkstra.cs.ttu.ee	upload.htm
200	GET	dijkstra.cs.ttu.ee	tests.json
200	GET	www.googletagmanager.c...	js?l=dataLayer
200	GET	dijkstra.cs.ttu.ee	upload_test.js
404	GET	dijkstra.cs.ttu.ee	favicon.ico

20 запросов 5,32 МБ / 4,73 МБ передано Передано за: 1,33 с DOMContentLoaded: 568 мс load: 1,29 с

Поиск в консоли

Figure 17. System with Bootstrap and jQuery

3.9 Compatibility with mobile devices

Desktop version of system is not suitable for smartphones. Small buttons, small text boxes and overlapping images make webpage uncomfortable for use (Figure 18).

With CSS *media* is used to make webpage compatible for mobile devices without changing desktop version design [14]. Size and location of some object were changed for convenience. This can be seen in Figure 19.

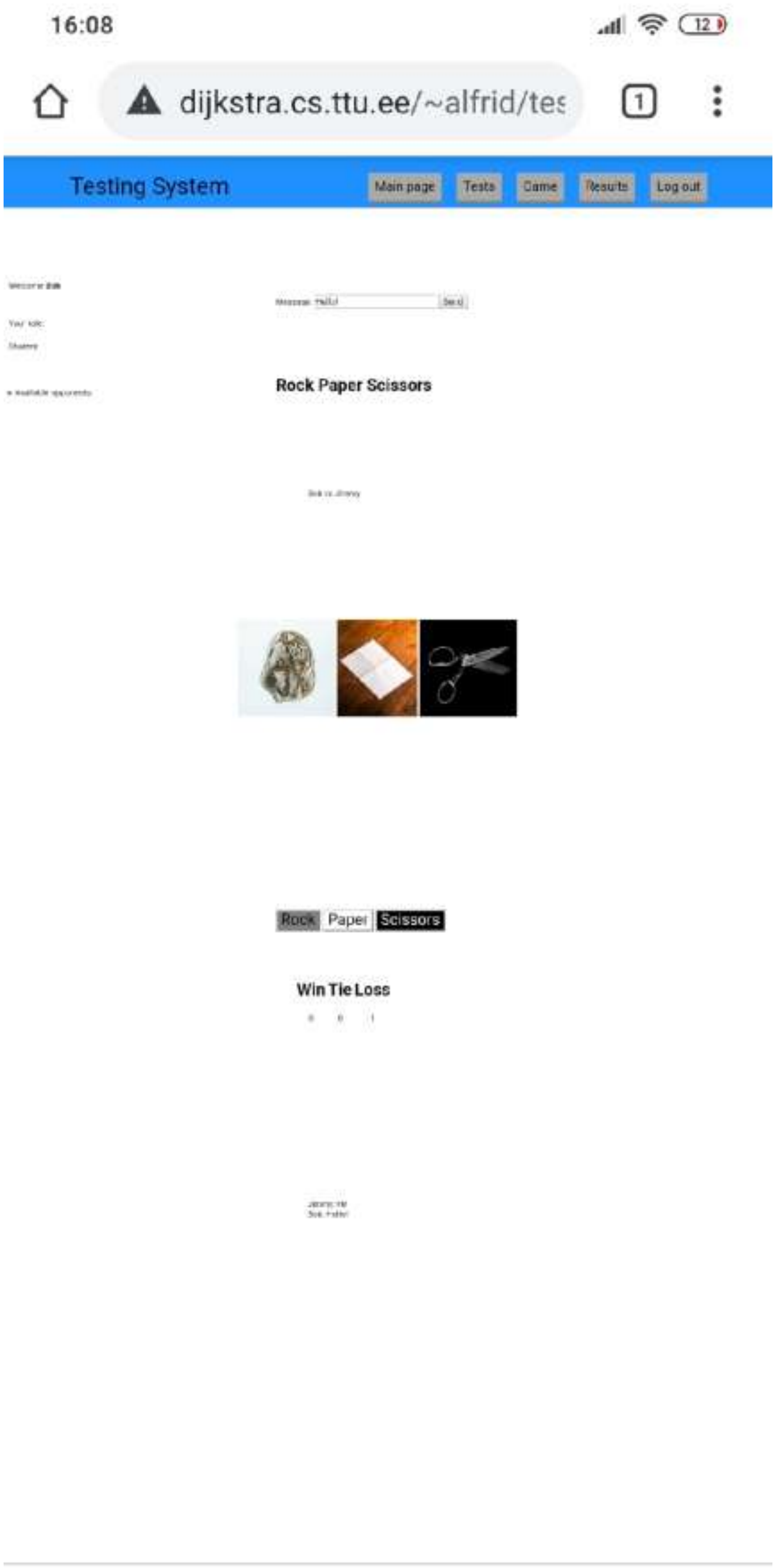
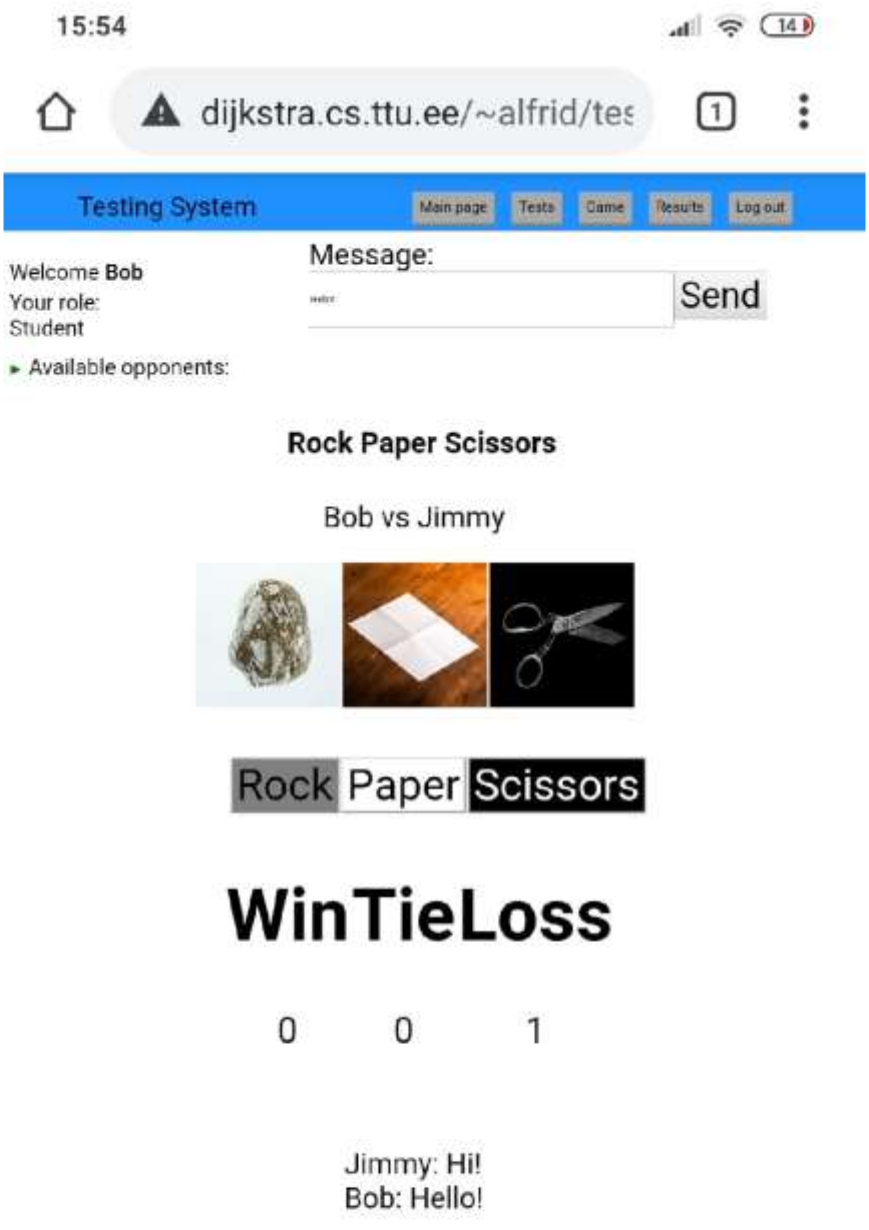


Figure 18. Mobile version with no compatibility



Jimmy: Hi!
Bob: Hello!

Figure 19. Mobile version with compatibility

4 Tests

4.1 Test creation

Teacher can choose a way to create a test. It can be created manually with HTML/JavaScript or some program can be used for creation of test [2]. For example, free application *Hot Potatoes* can be used [7].

4.1.1 Requirements for test

Test, which is needed to be uploaded, should be suitable for system. It means that test file should be an HTML page (*htm* or *html* format). Styles and scripts can be added to this file under <*style*> and <*script*> tags or uploaded as separate files (in *css* and *js* formats). Script should have function for calculating result of the test. Saving test results to *Firebase* is described in 4.3. *Hot Potatoes* application is advised to use, because it allows generating a test file in *htm* format. Source files can be modified, so it will not be needed for teacher to modify each test file manually.

4.2 Adding a test

After test is created, it should be added to web page. As was stated earlier test should be embedded to web page. Suitable decision is *iframe* [8]. Use of inline frame is shown in Figure 20.

```
<div id = "testing">
<iframe id ="currentTest" onfocusout=this.src='about:blank' width="750"
height="550"></iframe>
</div>
```

Figure 20. Test inline frame

Teacher should choose suitable file and upload it to system. Button “Upload Files” should be pressed (Figure 21).

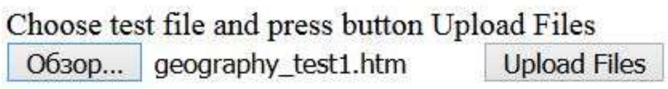


Figure 21. Test upload

Option values for *select* element, which make participating in test and getting the results possible will be created automatically (Figure 22).

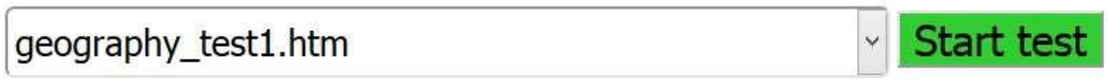


Figure 22. Select element for choosing test

```
function testing() {
    var iframe = document.getElementById("testToStart").value;
    var frame = document.getElementById('currentTest');
    frame.src = iframe;
    document.getElementById("testButtons").style.visibility = "hidden";
    document.getElementById("testing").style.visibility = "visible";
    document.getElementById("newTest").style.visibility = "hidden";
    document.getElementById("deleteButtons").style.visibility = "hidden";
}
```

Figure 23. Function for making test visible

Figure 23 is showing a function, which makes test visible. Shown frame and function are the same for all test, so only option values for select element, should be added. Option value creation is shown in Figure 24. Argument *jsonData* is data from file *tests.json*, which contains information about uploaded tests.

```
function setTest(jsonData) {
    for (i = 0; i < jsonData.length; i++) {
    var option = document.createElement("option");
    option.text = jsonData[i].test;
    document.getElementById("testToStart").add(option);
    }
}
```

Figure 24. Function for creating option values

4.3 Saving test results

In order to save results, test code should be modified. *Firebase* is used to store test results [15]. Firstly, necessary scripts should be added (Figure 25).

```
<!-- Insert these scripts at the bottom of the HTML, but before you use any
Firebase services -->

  <!-- Firebase App (the core Firebase SDK) is always required and must be
listed first -->
  <script src="https://www.gstatic.com/firebasejs/7.11.0/firebase-
app.js"></script>

  <!-- If you enabled Analytics in your project, add the Firebase SDK for
Analytics -->
  <script src="https://www.gstatic.com/firebasejs/7.11.0/firebase-
analytics.js"></script>

  <!-- Add Firebase products that you want to use -->
  <script src="https://www.gstatic.com/firebasejs/7.11.0/firebase-
firestore.js"></script>
<script>
```

Figure 25. Necessary scripts

Secondly, Firebase configuration should be added (Figure 26).

```
var firebaseConfig = {
    apiKey: "AIzaSyCUNnlvU8pl8CJP1jsADQvt6TEjhDdop8g",
    authDomain: "testing-2ab9d.firebaseapp.com",
    databaseURL: "https://testing-2ab9d.firebaseio.com",
    projectId: "testing-2ab9d",
    storageBucket: "testing-2ab9d.appspot.com",
    messagingSenderId: "641209812428",
    appId: "1:641209812428:web:1b0cd65f5c36e17bb9c969",
    measurementId: "G-Q9CT68ZW0G"
  };
  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);
  firebase.analytics();
```

Figure 26. Firebase configuration

These scripts and configuration can be added only once to Hot Potatoes source file (*jquiz7.ht*). After this, web page with test will get necessary part of code automatically. Finally, when test is done, data should be sent to server. In case of test, created with Hot Potatoes, function *CheckQuestionsCompleted()* should be modified (Figure 27). Modification is the same for every test. This modification can be also done only once in source file (*jquiz7.js*).

```
        if (QsCompleted >= QArray.length){
          CalculateOverallScore();
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        //console.log(this.response);
    }
};
xhttp.responseType = 'json';
xhttp.open("POST", "testing.php", true);
xhttp.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
xhttp.send();
xhttp.onload = function(data) {
                var obj = xhttp.response;
                var db = firebase.firestore();
                var name =
location.pathname.substring(location.pathname.lastIndexOf("/") + 1);
                db.collection("tests").add({
                user: obj[0],
                name: name,
                score: Score
                })
    .then(function(docRef) {
    console.log("Document written with ID: ", docRef.id);
})
.catch(function(error) {
    console.error("Error adding document: ", error);
                });
};
             return ExerciseCompleted;
```

Figure 27. function CheckQuestionsCompleted() modification

4.4 Getting test results

After the end of testing, teacher must have possibility to get students result. Only users with role „Teacher" can view the results of all participants. Other users can only see own result.

Similarly to adding, retrieving results also require creating option values. Function *answers()* should be called to show results. This function can be seen in Figure 30. Option values creation is shown in Figure 29. Argument *jsonData* is data from file *tests.json*, which contains information about uploaded tests. Function *answers()* shows teacher result of the test in descending order. That means, that students with biggest amount of points are above those, who have smaller score (Figure 31). To see result of the test, this test should be chosen in select element and button “Get Results” should be pressed (Figure 28).

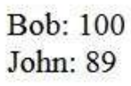
Figure 28. Select element for choosing test

```
function setResult(jsonData) {
    for (i = 0; i < jsonData.length; i++) {
    var option = document.createElement("option");
    option.text = jsonData[i].test;
    document.getElementById("resultOfTest").add(option);
    }
}
```

Figure 29. Option values creation

```
function answers() {
    var testName = document.getElementById("resultOfTest").value;
    role = document.getElementById("role").nextSibling.data;
    var username = "<?=$_SESSION['user']?>";
    var db = firebase.firestore();
    document.getElementById("answers").innerHTML = "";
    var test = db.collection("tests");
    test.orderBy("score", "desc").get().then(function(querySnapshot) {
    querySnapshot.forEach(function(doc) {
        // doc.data() is never undefined for query doc snapshots
        var data = doc.data();
        if (data.name == testName) {
             if (role == 'Teacher') {
            document.getElementById("answers").innerHTML += data.user + ": "
+ data.score + "<br/>";
            //console.log(doc.id, " => ", doc.data());
        } else {
            if (data.user == username) {
               document.getElementById("answers").innerHTML += data.user + ":
" + data.score + "<br/>";
            }
        }
        }
    });
});
    document.getElementById("answers").style.visibility = "visible";
    document.getElementById("answerButtons").style.visibility = "hidden";
}
```

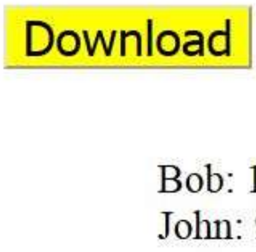
Figure 30. function answers()



Bob: 100
John: 89

Figure 31. Results of the test

Teacher can also download test results. Button “Download” should be pressed (Figure 32). Results can be downloaded in *txt* format. Function for creating and downloading file is shown in Figure 33.



Bob: 100
John: 93

Figure 32. Results download

```
function downloadResult() {
    var downloading = document.createElement('a');
    downloading.download = "result.txt";
    downloading.href = "";
    var text = document.getElementById("answers").childNodes;
    var results = "";
    for(i = 0; i < text.length; i++) {
        if (i % 2 == 0) {
        console.log(text.item(i).data);
        results += text.item(i).data + "\n";
        }
    }
    results = results.replace(/\n/g, "\r\n");
    var data = new Blob([results], {type: 'text/plain'});
    var url = window.URL.createObjectURL(data);
    downloading.href = url;
    document.body.appendChild(downloading);
    downloading.click();
    document.body.removeChild(downloading);
}
```

Figure 33. function downloadResults()

4.5 Test deletion

Test, which is not needed anymore can be deleted by user with role “Teacher”. Teacher should choose test, which is needed to be deleted and press button “Delete test” (Figure 34).

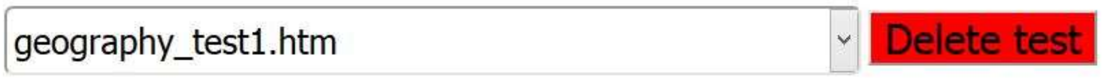


Figure 34. Test deletion

Pressing this button will modify file *tests.json*, which contains information about uploaded tests. Information about test, which was chosen in select element (Figure 34) will be deleted. After this it would be impossible to participate in this test or get result from it.

5 Game

As was stated earlier, game in this system is Rock-Paper-Scissors. Main differences between guest player and authenticated player are availability of encrypted chat and opportunity to play with another player (for authenticated player).

5.1 Starting a game

To start a game, authenticated user should press button “Game” (Figure 8). After this user can enter suitable opponent for the game into the text field (Figure 35).

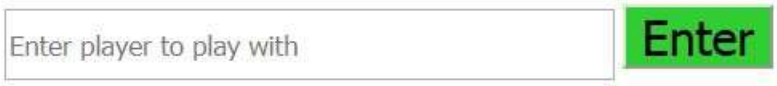
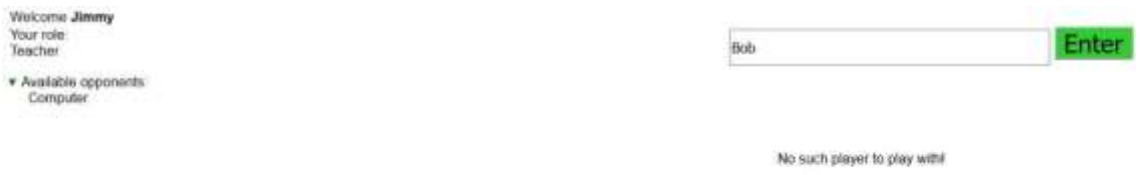


Figure 35. Game text field

In contrast with registration form, this text field should not be protected from SQL injection because this form is not sending data to server. It is working with data, which is already loaded from server.

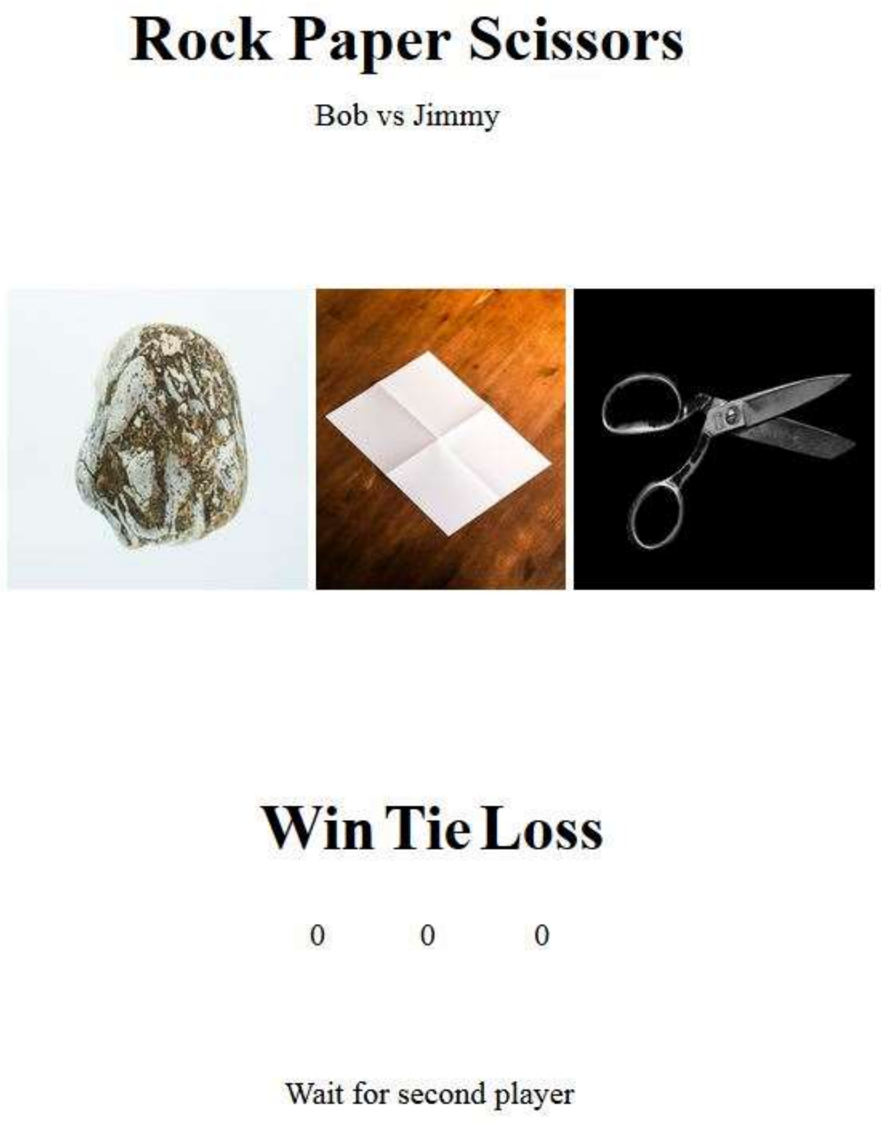
If entered opponent’s username is incorrect, play can see message (Figure 36).



No such player to play with!

Figure 36. Incorrectly entered opponent

Otherwise, user can see game menu (Figure 37).


Rock Paper Scissors

Bob vs Jimmy

Win Tie Loss

0 0 0

Wait for second player

Figure 37. Game menu

When second player enters a game, message “Wait for second player” disappearing and it is possible to play (Figure 38).

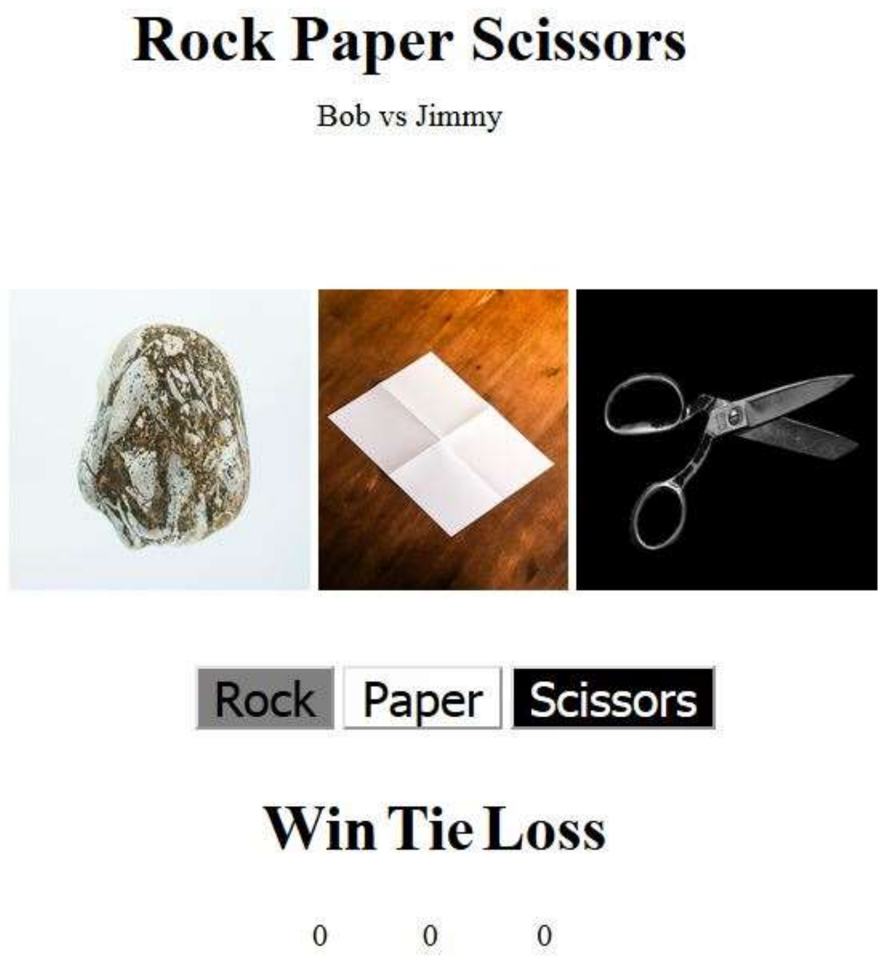
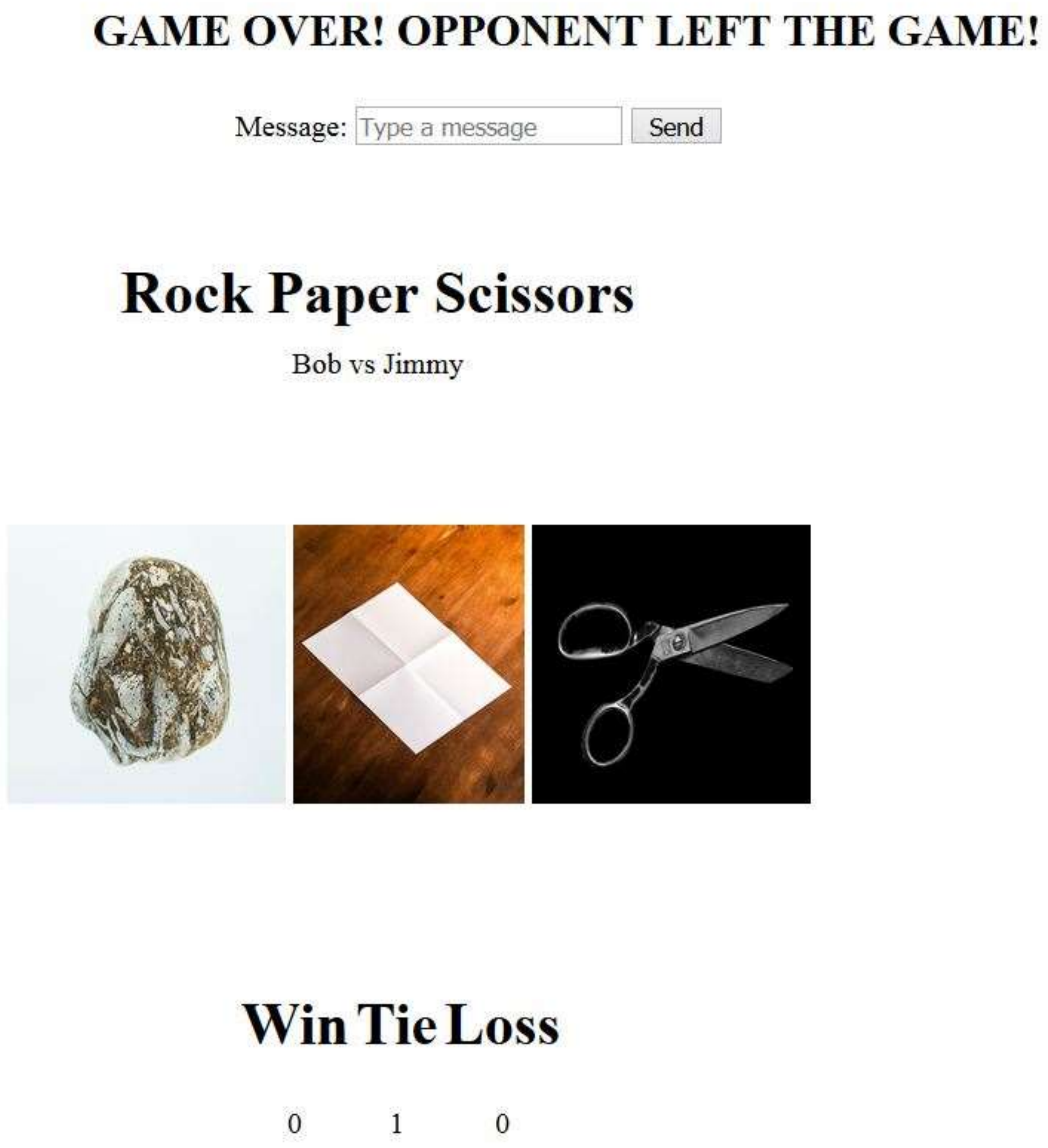


Figure 38. Game with both players entered

5.2 Quitting a game

If player during a game press “Log out” button or started a new game with another player, his opponent must know that opponent is not playing anymore. If opponent is quitting, player after making new move can see warning message over the game menu (Figure 39).


GAME OVER! OPPONENT LEFT THE GAME!

Rock Paper Scissors

Bob vs Jimmy

Win	Tie	Loss
0	1	0

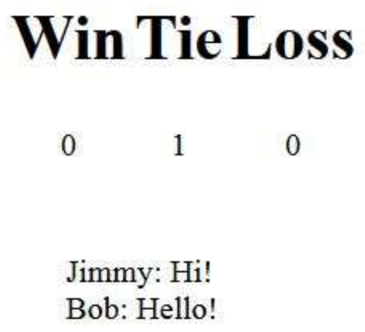
Figure 39. Game after opponent’s leaving

5.3 Chat

After game started, when first moves are done, possibility to communicate with opponent is available. Form to send messages can be found over the game menu (Figure 40).

Figure 40. Chat form

Already sent messages can be found under game menu (Figure 41).


Win Tie Loss

0 1 0

Jimmy: Hi!
Bob: Hello!

Figure 41. Example of chat

5.3.1 Encryption

As was mentioned before chat should be encrypted. Two main types of encryption are symmetric and asymmetric. In case of this chat symmetric encryption is more suitable, because encryption and decryption are happening with one key [9]. That means only one key should be generated.

5.3.2 Encryption keys

Considering the fact, that encryption keys should be created locally, players moves can be used for generating key. When two users are playing with each other, their moves are converted to hexadecimal string with appropriate length. *AES-128* algorithm is used for encryption, so sufficient length of hexadecimal string used for key is 32 characters [10]. AES-128 is suitable because keys with bigger number of bits would slow down the encryption/decryption process [11]. Considering the fact, that there are no successful attacks on AES-128 at present, there is no practical use of algorithms with longer keys in described game [20]. Creation of key is shown in Figure 42. Variable *obj* in function *convert(obj)* is an array with all moves of the players.

```
function convert(obj) {
    var string = "";
    if (obj.length % 2 == 0 && obj.length > 0) {
    for(i = 0; i < obj.length; i++) {
        string += obj[i];
    }
    binary(string);
    }
}
function binary(string) {
    var converted = "";
    for (i = 0; i < string.length; i++) {
      converted += "0" + string[i].charCodeAt(0).toString(2);
  }
    hexConvert(converted);
}
var send = "";
function hexConvert(converted) {
    question = BigInt(converted);
    var hex = question.toString(16);
    if (hex.length < 32) {
        console.log("Make more moves!");
    } else {
        var key = "";
        var answer = hex.split("", 32);
        for (i = 0; i < answer.length; i++) {
            key += answer[i];
        }
        keys = key;
        document.getElementById("sending").style.visibility = "visible";
        document.getElementById("resulting").style.visibility = "visible";

    }
}
```

Figure 42. Creation of key

5.3.3 Message encryption with given key

Stanford Javascript Crypto Library is used for encrypting message [12]. Message encryption is shown in Figure 43. Function *encrypt(key, message)* from library is used, where *message* is text, which user want to send and *key* is encryption key generated locally.

```
function encryptMessage(message, key) {
    send = sjcl.encrypt(key, message);
}
```

Figure 43. Message encryption

Encryption key is sensitive information, so it cannot be stored on server. So, even in case of data leak strangers would not get access to messages. Message content would be unreadable. Message format on server can be seen in Table 1.

id	username	game	message
147	Bob	with Jimmy	{"iv":"8wMC9RDGGLglU0imriKXMA==","v":1, "iter":10000, "ks":128,"ts":64,"mode":"ccm","adata":"","cipher":"aes", "salt":"r56TiAzsG28=","ct":"PDyEH3DMEvejDJs="}
148	Jimmy	with Bob	{"iv":"Mlc1TMaEOd7wPQWazEKI/Q==","v":1, "iter":10000, "ks":128,"ts":64,"mode":"ccm","adata":"","cipher":"aes", "salt":"r56TiAzsG28=","ct":"9IC9pq9nufBwq3YkM54="}

Table 1. Example of messages stored on server

5.3.4 Message decryption with given key

Stanford Javascript Crypto Library is also used for decryption [12]. Message decryption is shown in Figure 44. Function *decrypt(key, message)* from library is used, where *message* is encrypted text and *key* is the same key, which was used for encrypting.

```
function decryptMessage(message, key) {
    var decrypted = sjcl.decrypt(key, message);
        texting = decrypted;
}
```

Figure 44. Message decryption

It is needed to generate key only once for each chat. If key would change with every move, there would no possibility to read old messages with this key. During the game the sequence of player’s moves is increasing. For successful communication key must be the same for both who participate in dialog. When first message is sent, in function *printMessages(obj)* key, which was used for encrypting this message is saved. Key will be the same until new game is started. Function *printMessages(obj)* can be seen in Figure 45.

```
function printMessages(obj) {
    document.getElementById("resulting").innerHTML = "";
    if (stableKey === "") {
    for(i = 0; i < obj.length; i++) {
        if ((i + 1) % 2 == 0) {
        var messageFromServer = obj[i];
        decryptMessage(messageFromServer, keys);
        stableKey = keys;
        document.getElementById("resulting").innerHTML += author + ": " +
texting + "<br/>";
        } else {
            author = obj[i];
        }
    }
} else {
    for(i = 0; i < obj.length; i++) {
        if ((i + 1) % 2 == 0) {
        var messageFromServer = obj[i];
        decryptMessage(messageFromServer, stableKey);
        document.getElementById("resulting").innerHTML += author + ": " +
texting + "<br/>";
        } else {
            author = obj[i];
        }
}
}
}
```

Figure 45. function printMessages(obj)

5.3.5 Opportunities for attacks

Messages are stored on server. When user start a new game or log out, old messages are deleted. Attack in real time can be done. If someone would gain access to server, messages can be seen in table (Table 1). Messages are encrypted, so they cannot be read. Without an encryption key it would take incredibly long to decipher message [20].

So, if it is too hard to brute force, someone would try to retrieve encryption key. If sequence of player moves is gained, suitable key for algorithm can be created. Unfortunately for this “bad guy” sequence of all players moves would not help him to generate right key. Key for each dialogue is generated after first message is sent. So not only sequence of moves is needed to replicate used key in dialogue, but also it is needed to know exact moment when communication was started (sequence of players moves at this exact moment).

Rock-Paper-Scissors is well-known game. There are only 3 combinations for each move. Encryption key can be generated without knowing the right sequence of moves. It depends on number of possible combinations. 3^{2n} is number of possible combinations, there n is number of moves made by each player. If each player made 3 moves and after that conversation was started, that means 729 different keys could be generated.

5.4 Images used in game

Some images are used in game. Images can be seen in Figure 46. These pictures are downloaded from *Unsplash*. It is a site, that provides stock photographs, which are free to use [13].

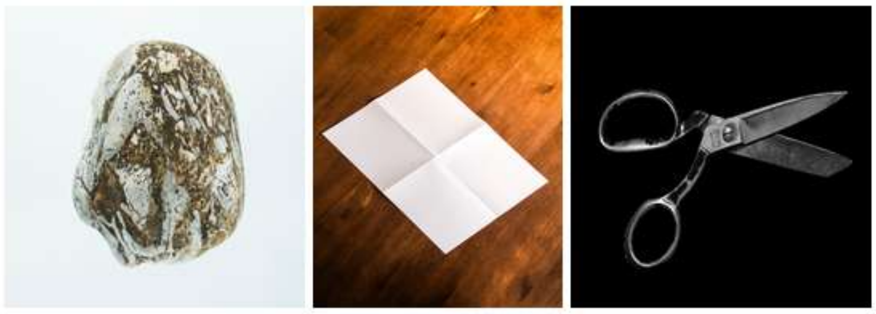


Figure 46. Images in game

6 Summary

The aim of this work was to create system, where teacher can get results of tests from students without checking them manually. Also, multiplayer game with encrypted chat was created.

Created system allows teacher to upload a test and to see student’s results from uploaded tests. Uploading tests and receiving results are happening directly from browser, so it is not needed to handle server directly. Result of the test is automatically calculated, so there is need for teacher to check each work separately.

A lot of applications for creating test already exists [2]. For teacher it is better to have this implemented system, because it has profits comparing to existing applications. There is no trial version, so system can be used for years. Web tests are embedded to webpage, so users do not need to download third-party software. No limits for test, subject, question or result number.

System let users play Rock-Paper-Scissors game with another user and with possibility of secure communication. Security of communication is achieved with use of cryptography (AES-128 algorithm used for encryption and decryption; encryption keys generated locally).

Implemented system can be developed further. Getting several tests results at same time or getting specific student’s result could be done. System can be also used not only for educational purposes. For example, some other game could be implemented in system.

References

[1] Valitsus kuulutas Eestis välja eriolukorra 1. maini [WWW] https://www.valitsus.ee/et/uudised/valitsus-kuulutas-eestis-valja-eriolukorra-1-maini (25.03.20)

[2] Top 10 Quiz Makers for Teachers and Educators [WWW] https://www.digitalchalk.com/resources/blog/elearning-tools/top-10-quiz-makers-teachers-educators (25.03.20)

[3] How Does Online Gaming Affect Social Interactions [WWW] https://www.sciencedaily.com/releases/2007/09/070915110957.htm (26.03.20)

[4] Why encrypted messaging is more important than ever [WWW] https://techwireasia.com/2017/11/encrypted-messaging-important-ever/ (26.03.20)

[5] What is Bring Your Own Key (BYOK) [WWW] https://www.thalesesecurity.com/faq/key-secrets-management/what-bring-your-own-key-byok (26.03.20)

[6] mysql_real_escape_string [WWW] https://www.php.net/manual/en/function.mysql-real-escape-string.php (27.03.20)

[7] Hot Potatoes Home Page [WWW] https://hotpot.uvic.ca/ (30.03.2020)

[8] HTML <iframe> Tag [WWW] https://www.w3schools.com/tags/tag_iframe.asp (30.03.2020)

[9] Description of Symmetric and Asymmetric Encryption [WWW] https://support.microsoft.com/en-us/help/246071/description-of-symmetric-and-asymmetric-encryption (01.04.2020)

[10] How long (in letters) are encryption keys for AES? [WWW] https://security.stackexchange.com/questions/45318/how-long-in-letters-are-encryption-keys-for-aes/45334 (01.04.2020)

[11] The Doghouse: Crypteto [WWW] https://www.schneier.com/blog/archives/2009/09/the_doghouse_cr.html (01.04.2020)

[12] Stanford Javascript Crypto Library [WWW] http://bitwiseshiftleft.github.io/sjcl/ (01.04.2020)

[13] License [WWW] https://unsplash.com/license (03.04.2020)

[14] CSS Media Queries [WWW] https://www.w3schools.com/css/css3_mediaqueries.asp (13.04.20)

[15] Firebase [WWW] https://firebase.google.com/ (09.05.2020)

[16] D. Liu and C. Lin, „Sherlock: a Semi-Automatic Quiz Generation System using Linked Data," in *Proceedings of the ISWC 2014 Posters & Demonstrations Track a track within the 13th International Semantic Web Conference (ISWC 2014)*: CEUR Workshop Proceedings, 2014, pp. 9-12.

[17] 4 Reasons Why the Cloud Is More Secure Than Legacy Systems [WWW] https://www.tripwire.com/state-of-security/security-data-protection/4-reasons-why-the-cloud-is-more-secure-than-legacy-systems/ (17.05.20)

[18] Kahoot! [WWW] https://kahoot.com (19.06.20)

[19] Strawpoll [WWW] https://strawpoll.com (19.06.20)

[20] AES Encryption isn't Cracked [WWW] https://web.archive.org/web/20150108165723/https://blog.agilebits.com/2011/08/18/aes-encryption-isnt-cracked/ (25.06.20)