QUIZ MANAGER

-Final Project Documentation-

Alexandra-Elena Achimov
WEB DEVELOPMENT

Table of Contents

General Objective

My goal with this project is to create a functional, interactive and aesthetically-pleasing quiz game. In the process, I want to test my current knowledge in Web Development by making use of the HTML, CSS and JavaScript concepts I have been learning over the course of the last eight months.

The name of the game will be “Superlatives Quiz Game”, and the game will test the general knowledge in terms of ‘superlatives’ of the player on a various number of topics. It is important to have in mind that by ‘superlatives’ I do not refer to the superlative adjectives in the English language, but to the player’s general knowledge when it comes to ‘what is the hardest/easiest/ tallest/smallest/most/less....” etc. when it comes to a specific domain. The app will be as interactive as my current level of knowledge permits and will incorporate some customizable features as well.

The reason I chose this theme is because, upon searching the internet I couldn’t find a quiz game that centered on superlatives in terms of general knowledge and I thought it would be fun to test this knowledge.

Quiz Functionality

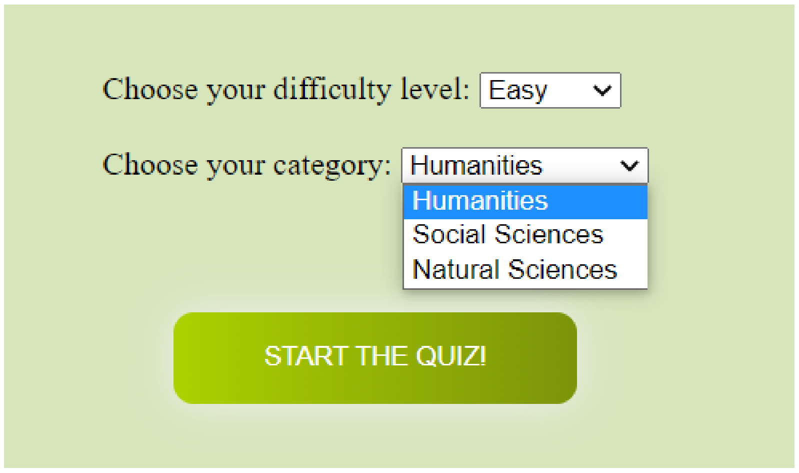
Questions

The quiz will consist of four large categories the player can choose from before starting the game: Humanities, Social Sciences and Natural Science. Changing the category mid-game will not be possible. Each category will consist of other three subcategories as follows:

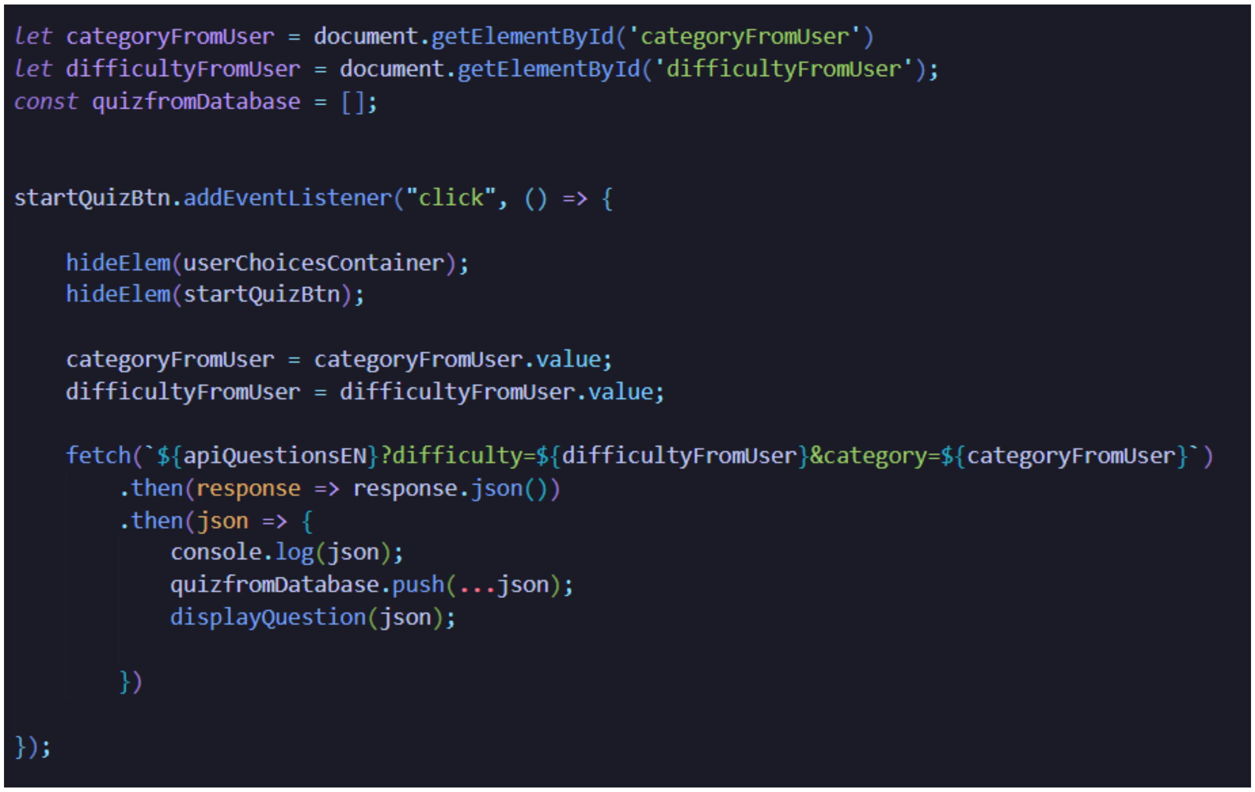
- Humanities: History, Philosophy, Literature
- Social Sciences: Psychology, Geography, Economics
- Natural Sciences: Biology, Astronomy, Chemistry

Each category will contain 15 questions, 5 from each subcategory. The questions will be mixed so that the player won't be put in the position to answer two questions from the same category two times in a row. The number of the questions won't be displayed to the player, neither before the game, neither during it.

The player will be able to choose between 3 already-generated options at each question. The questions will be displayed on a card.


Choose your difficulty level: Easy

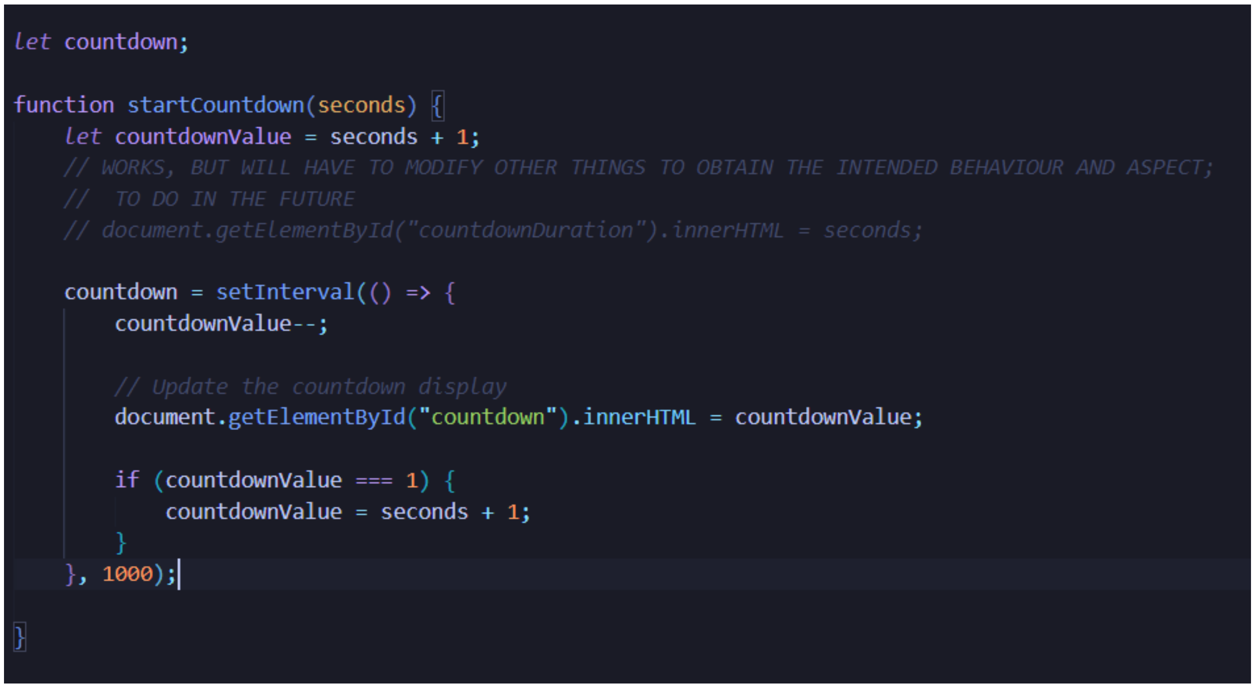
Choose your category: Humanities



```
let categoryFromUser = document.getElementById('categoryFromUser')
let difficultyFromUser = document.getElementById('difficultyFromUser');
const quizfromDatabase = [];


startQuizBtn.addEventListener("click", () => {

    hideElem(userChoicesContainer);
    hideElem(startQuizBtn);

    categoryFromUser = categoryFromUser.value;
    difficultyFromUser = difficultyFromUser.value;

    fetch(`${apiQuestionsEN}?difficulty=${difficultyFromUser}&category=${categoryFromUser}`)
        .then(response => response.json())
        .then(json => {
            console.log(json);
            quizfromDatabase.push(...json);
            displayQuestion(json);

        })

});
```

Timer

The player will have to answer each question provided within a time limit which will be displayed to them. The time limit will differ depending on the level of difficulty, getting shorter as the difficulty increases: 13 seconds for easy mode, 10 seconds for medium and 8 seconds for hard level. The timer will be displayed above the card on which the questions will appear.

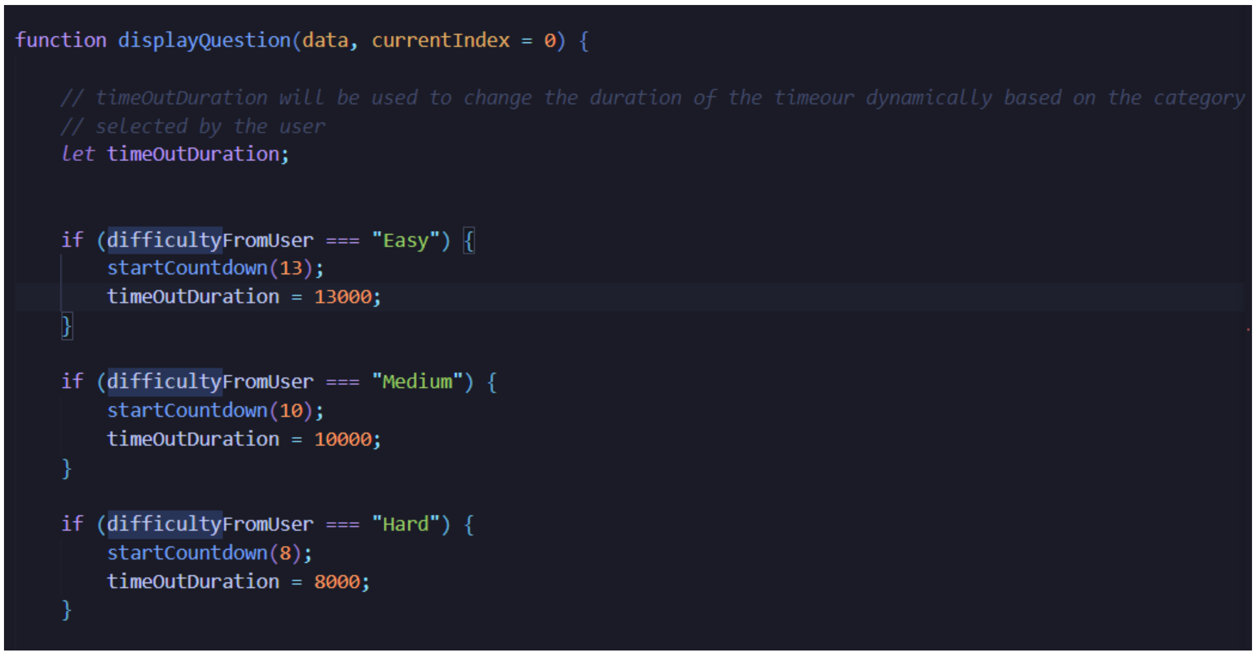


```
let countdown;

function startCountdown(seconds) {
    let countdownValue = seconds + 1;
    // WORKS, BUT WILL HAVE TO MODIFY OTHER THINGS TO OBTAIN THE INTENDED BEHAVIOUR AND ASPECT;
    //  TO DO IN THE FUTURE
    // document.getElementById("countdownDuration").innerHTML = seconds;

    countdown = setInterval(() => {
        countdownValue--;

        // Update the countdown display
        document.getElementById("countdown").innerHTML = countdownValue;

        if (countdownValue === 1) {
            countdownValue = seconds + 1;
        }
    }, 1000);

}
```

Difficulty Levels

There will be three levels of difficulty possible: easy, medium and hard. The player will have the possibility to choose their desired level of difficulty before beginning the game. After starting the game, changing its difficulty level during playing will not be possible.

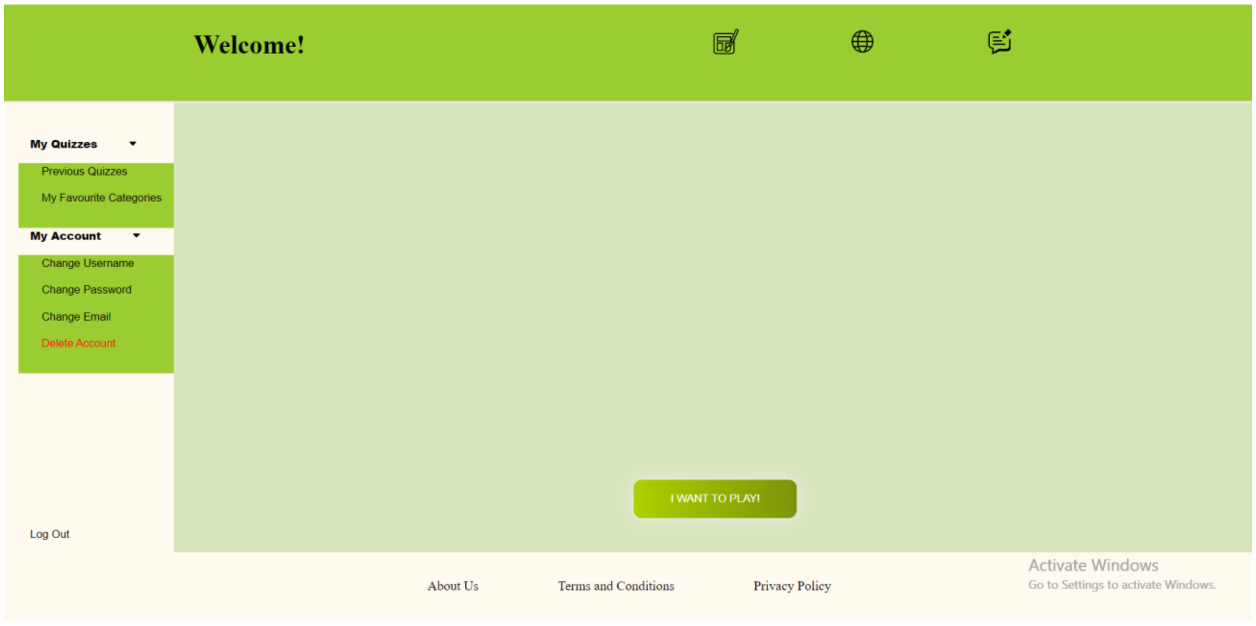
The difference between the three levels of difficulty will be given as follows: by the gradual increase in the difficulty and complexity of the questions, by the increased level of ambiguity of the options provided to the player and also by the increased time limit to answer each question. These differences in difficulty levels won't be communicated to the user.

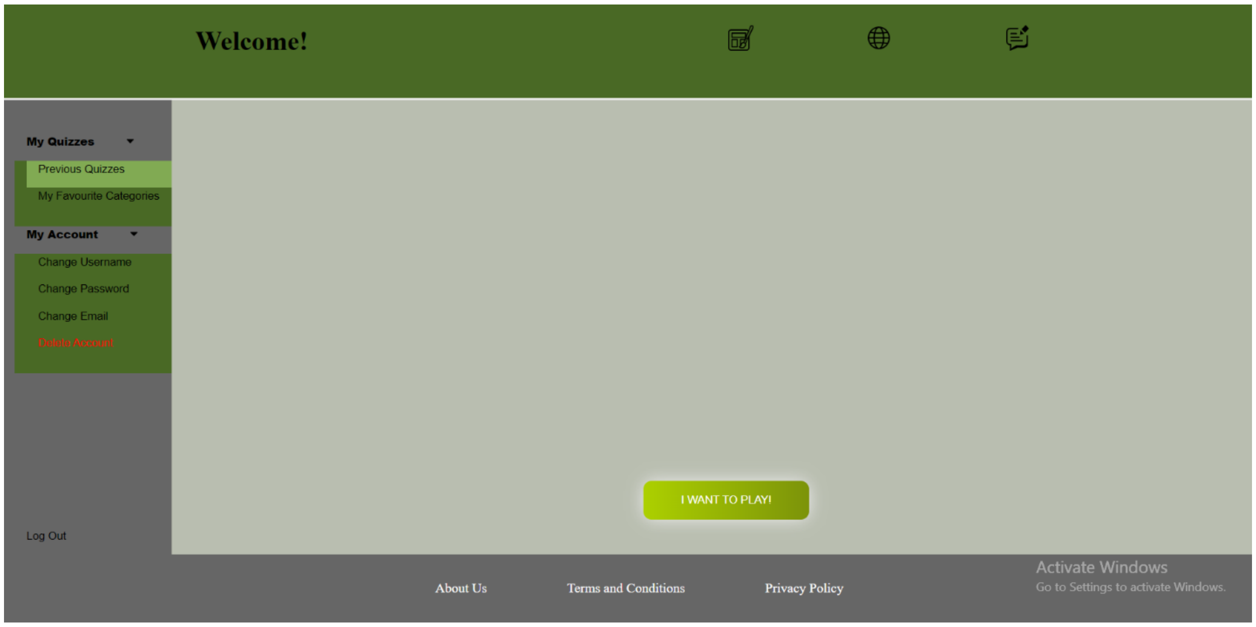


```
function displayQuestion(data, currentIndex = 0) {

    // timeOutDuration will be used to change the duration of the timeour dynamically based on the category
    // selected by the user
    let timeOutDuration;


    if (difficultyFromUser === "Easy") {
        startCountdown(13);
        timeOutDuration = 13000;
    }

    if (difficultyFromUser === "Medium") {
        startCountdown(10);
        timeOutDuration = 10000;
    }

    if (difficultyFromUser === "Hard") {
        startCountdown(8);
        timeOutDuration = 8000;
    }
```

Themes

The player of the game will have the possibility to choose between two themes: light and dark.



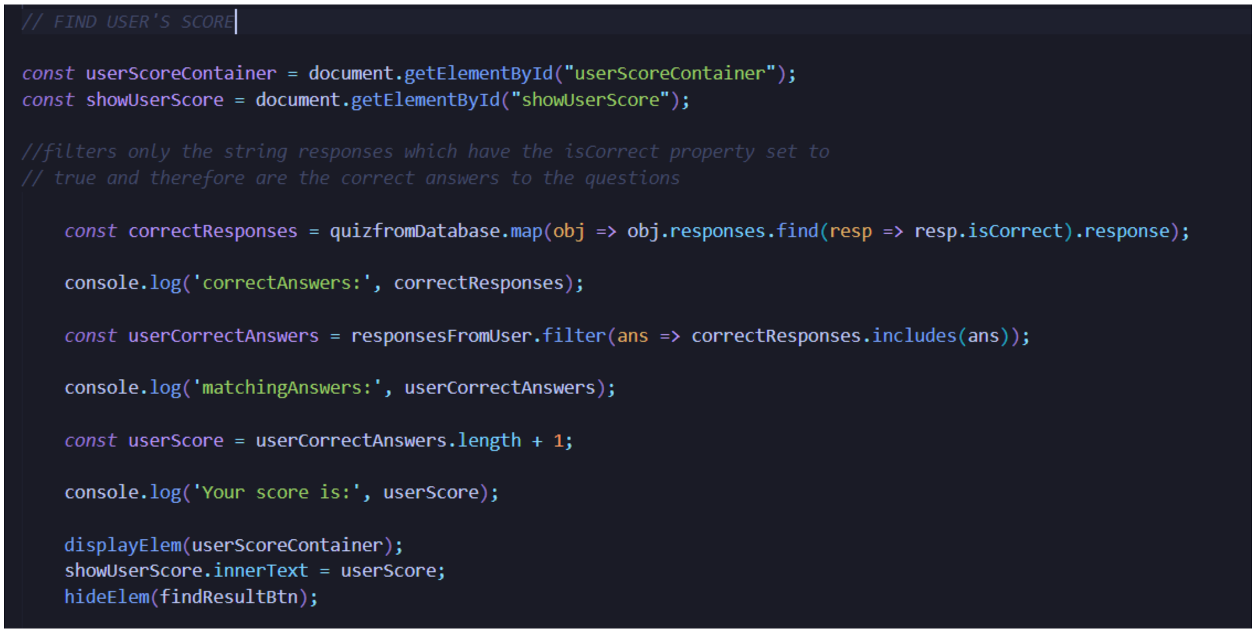


Language *(future improvement)*

The game will be available in two languages: English and Romanian. The player will have the possibility to change the language at any time, before or during the game.

Score

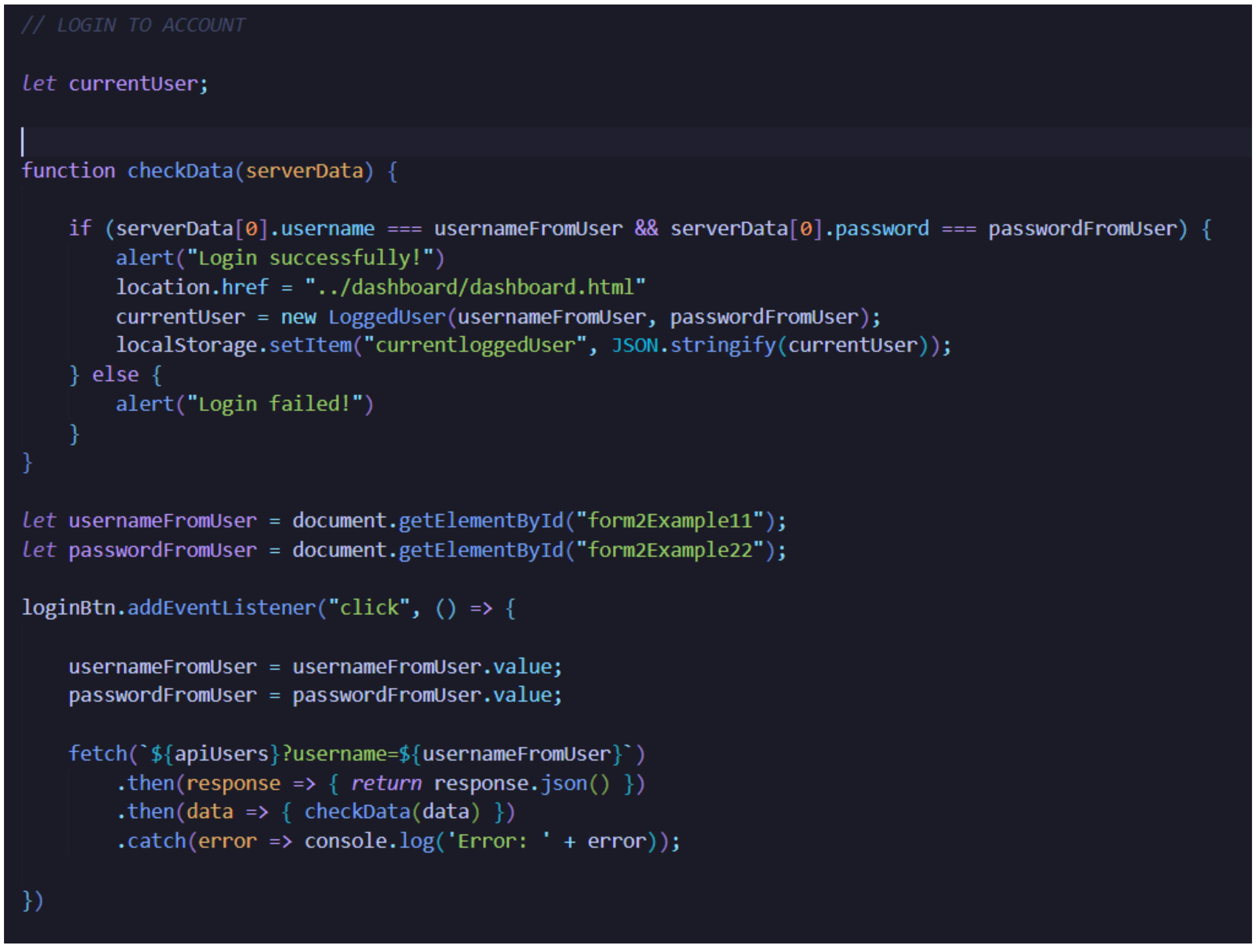
Each question answered correctly will increase the score with 1 point. Each question (regardless of the level of difficulty chosen by the player) equals 1 point. At the end of the game, the score obtained by the user will equal the number of questions answered correctly. Moreover, this number will be displayed to the player at the end of the game.


```
// FIND USER'S SCORE

const userScoreContainer = document.getElementById("userScoreContainer");
const showUserScore = document.getElementById("showUserScore");

//filters only the string responses which have the isCorrect property set to
// true and therefore are the correct answers to the questions

    const correctResponses = quizfromDatabase.map(obj => obj.responses.find(resp => resp.isCorrect).response);

    console.log('correctAnswers:', correctResponses);

    const userCorrectAnswers = responsesFromUser.filter(ans => correctResponses.includes(ans));

    console.log('matchingAnswers:', userCorrectAnswers);

    const userScore = userCorrectAnswers.length + 1;

    console.log('Your score is:', userScore);

    displayElem(userScoreContainer);
    showUserScore.innerText = userScore;
    hideElem(findResultBtn);
```

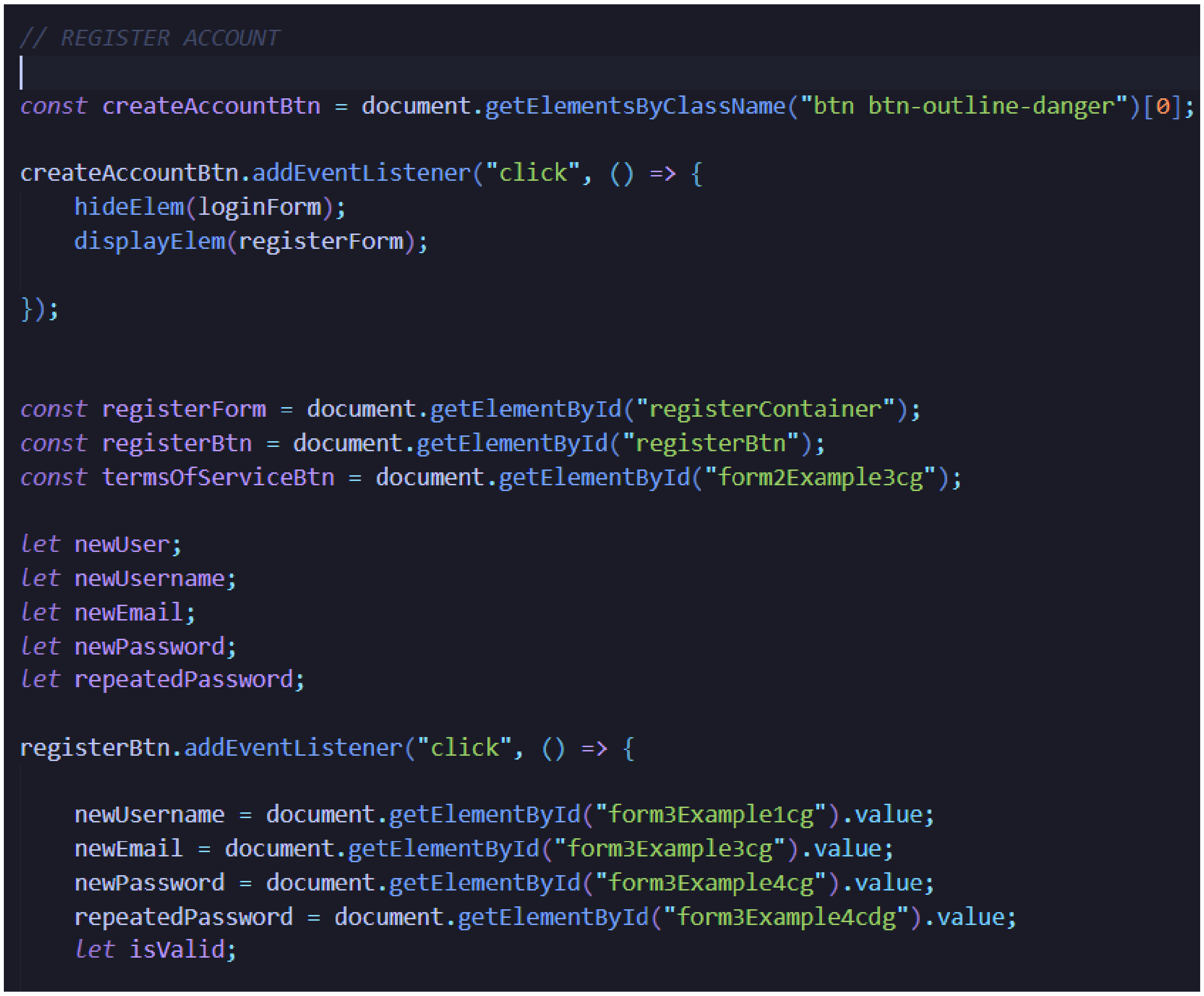
Login/ Register Functions

To play the game, the user will need to create an account if they haven't created done so before, or login if they have already created an account. They will be prompted to choose one option when entering the home page.

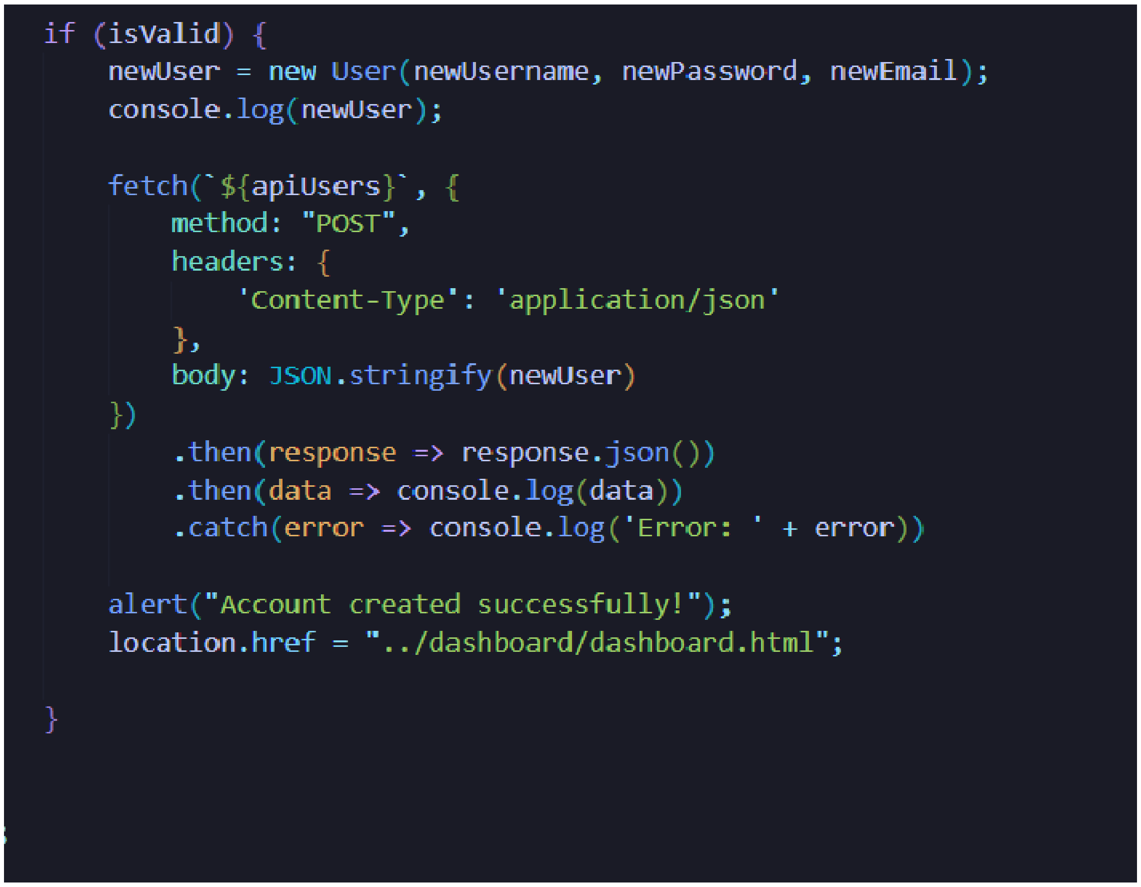
If they choose the log in option, they will be asked to provide the nickname and the password associated with their account; if they choose the register option, they will need to enter their e-mail address, a unique username to be associated with their account, and a password.



```
// LOGIN TO ACCOUNT

let currentUser;

function checkData(serverData) {

    if (serverData[0].username === usernameFromUser && serverData[0].password === passwordFromUser) {
        alert("Login successfully!")
        location.href = "../dashboard/dashboard.html"
        currentUser = new LoggedUser(usernameFromUser, passwordFromUser);
        localStorage.setItem("currentloggedUser", JSON.stringify(currentUser));
    } else {
        alert("Login failed!")
    }
}

let usernameFromUser = document.getElementById("form2Example11");
let passwordFromUser = document.getElementById("form2Example22");

loginBtn.addEventListener("click", () => {

    usernameFromUser = usernameFromUser.value;
    passwordFromUser = passwordFromUser.value;

    fetch(`${apiUsers}?username=${usernameFromUser}`)
        .then(response => { return response.json() })
        .then(data => { checkData(data) })
        .catch(error => console.log('Error: ' + error));

})
```



```
// REGISTER ACCOUNT

const createAccountBtn = document.getElementsByClassName("btn btn-outline-danger")[0];

createAccountBtn.addEventListener("click", () => {
    hideElem(loginForm);
    displayElem(registerForm);

});


const registerForm = document.getElementById("registerContainer");
const registerBtn = document.getElementById("registerBtn");
const termsOfServiceBtn = document.getElementById("form2Example3cg");

let newUser;
let newUsername;
let newEmail;
let newPassword;
let repeatedPassword;

registerBtn.addEventListener("click", () => {

    newUsername = document.getElementById("form3Example1cg").value;
    newEmail = document.getElementById("form3Example3cg").value;
    newPassword = document.getElementById("form3Example4cg").value;
    repeatedPassword = document.getElementById("form3Example4cdg").value;
    let isValid;
```



```
if (isValid) {
    newUser = new User(newUsername, newPassword, newEmail);
    console.log(newUser);

    fetch(`${apiUsers}`, {
        method: "POST",
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify(newUser)
    })
        .then(response => response.json())
        .then(data => console.log(data))
        .catch(error => console.log('Error: ' + error))

    alert("Account created successfully!");
    location.href = "../dashboard/dashboard.html";

}
```

Board Page

After logging onto the platform, the user will be redirected to the board page. On this page the user will have access the following functionalities of the project:

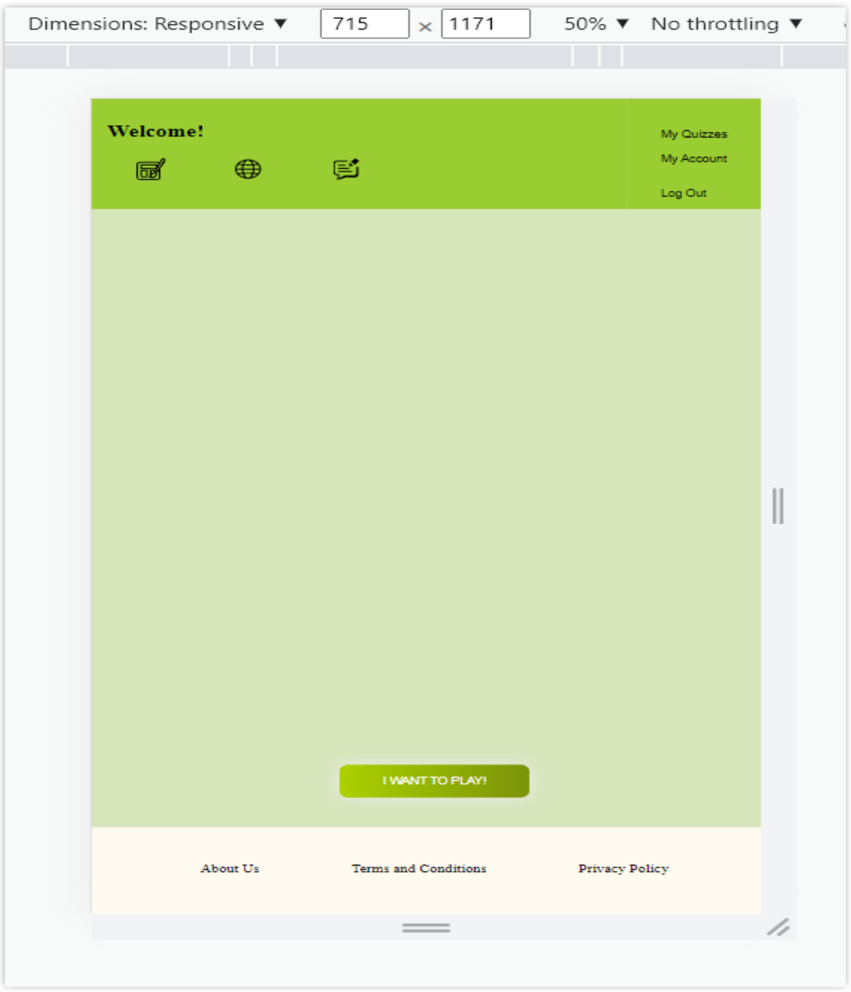
- to start playing the game; after pressing the corresponding button, the user will be redirected to another page and prompted to select the category and difficulty level of their choice;
- to view their previous completed quizzes, along with the scores obtained and the difficulty level & category associated with those quizzes;
    - the answers they provided, and whether they were correct or not, won't be shown to the user;
- to change the theme;
- to change the language;

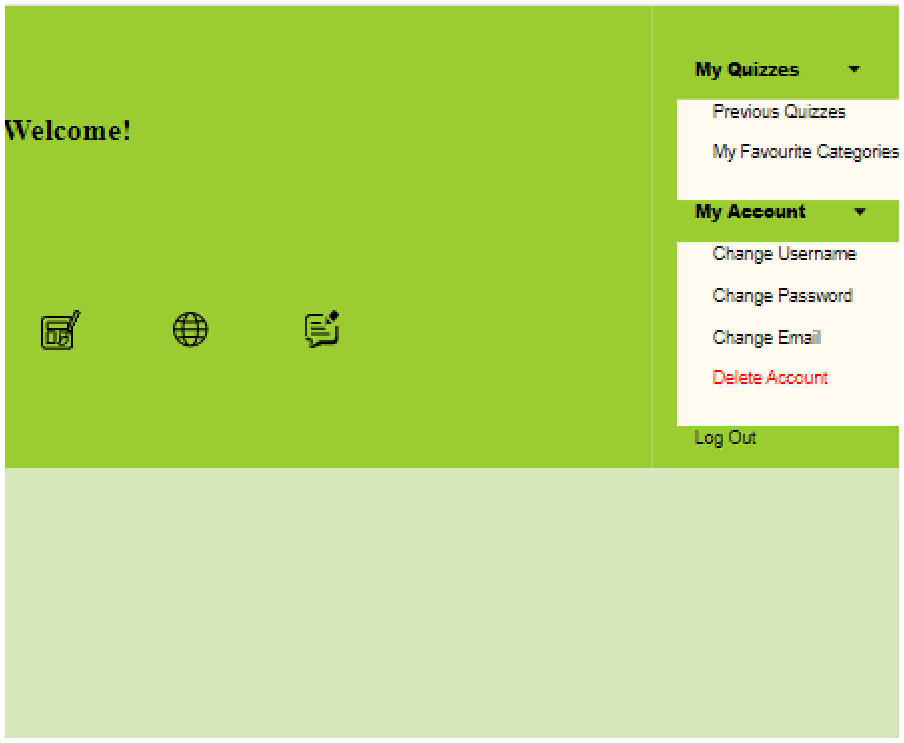
- to change their username;
- to log out;
- to send feedback to the creator of the user via the contact form;

Responsive Design

In terms of responsive design, the quiz will have 3 thresholds and be responsive on each one of them:

- desktop (less than 1280 px),
- tablet (less than 962 px),
- mobile (less than 720 px);





Feedback Button

The user will have the possibility to send his thoughts on the project or questions related to the game via a contact us form.

Layout

In terms of layout, the project will consist of 2 pages:

- the 1st page will be represented by the home page where the user will be prompted to login or register in order to play the game,
- the 2nd page will be the board game containing all its associated functionalities. The board game will be further composed from a header, a body and a footer. In

the header, the options to change the language and the theme, along with the feedback button will be present; the body will contain the account functionalities along with the options to start the game, ~~which, if chosen, will redirect the user to the 3rd page~~; and finally, the footer will contain 3 hyperlinks: About Us, Terms and Conditions and Privacy Policy,

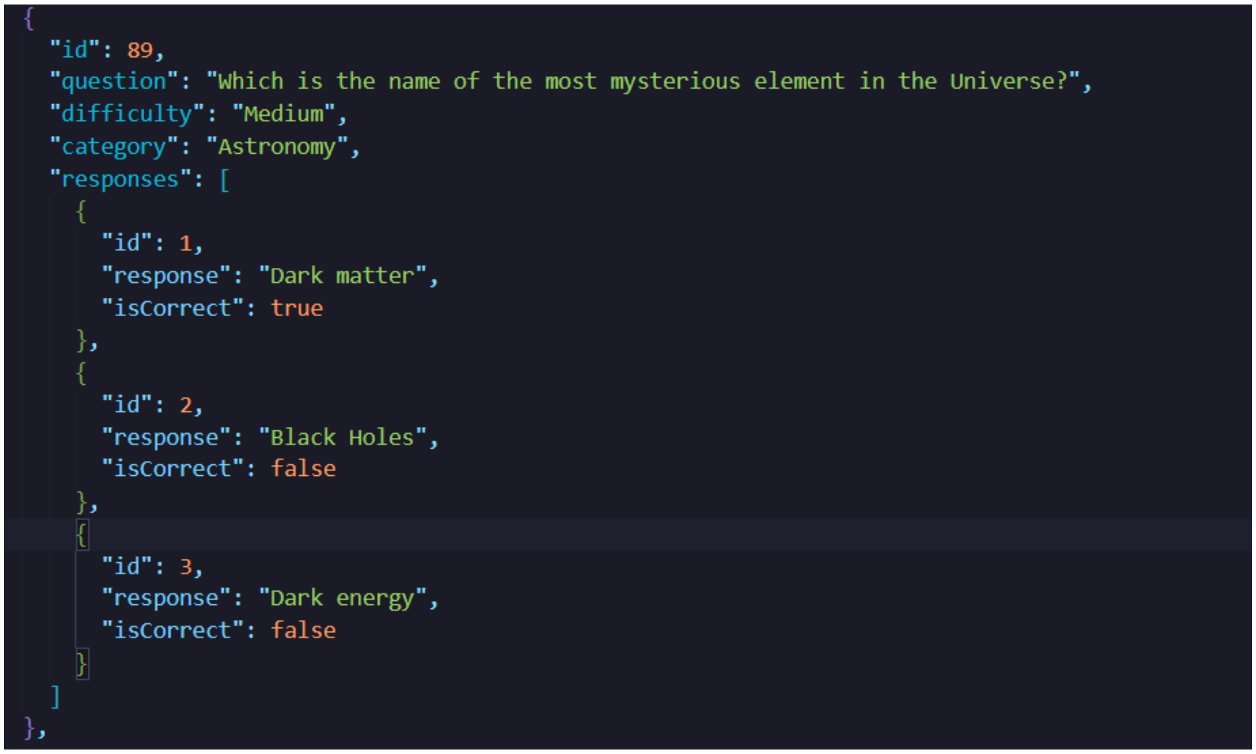
- ~~the 3rd page will consist of the quiz itself, along with its associated functionalities (categories, difficulty levels, timer).~~

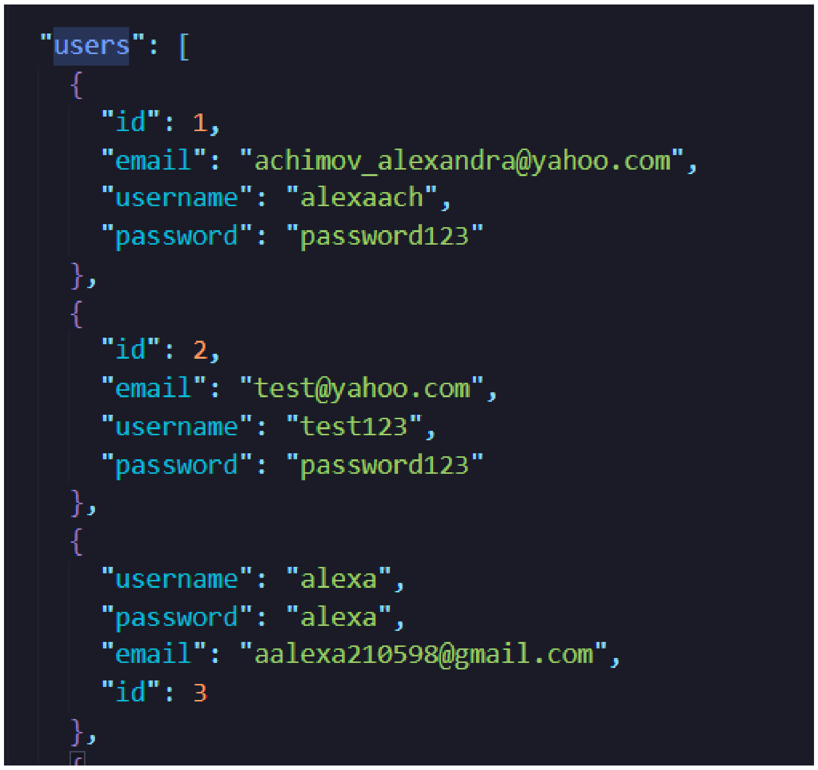
Implementation Flow

The implementation process will be organized in 4 stages:

Step 1: define the database

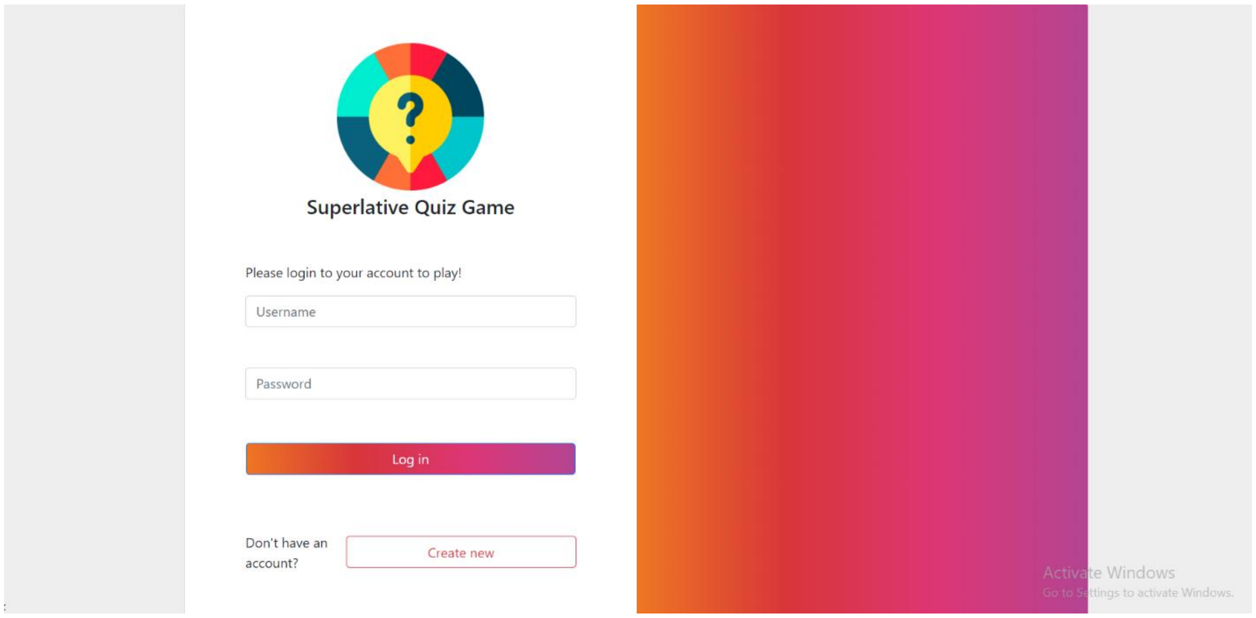
- all the questions will be stored in a JSON Database which will be accessed via a private API using Fetch Requests and Promises; all the entries will have a unique id associated with it which will make it possible to manipulate the date;
- *time necessary to complete the step:* since the volume of work is considerate and it also implies coming up with the questions themselves, I think it will take me around *3 days* to finish this task;


```
{
  "id": 89,
  "question": "Which is the name of the most mysterious element in the Universe?",
  "difficulty": "Medium",
  "category": "Astronomy",
  "responses": [
    {
      "id": 1,
      "response": "Dark matter",
      "isCorrect": true
    },
    {
      "id": 2,
      "response": "Black Holes",
      "isCorrect": false
    },
    {
      "id": 3,
      "response": "Dark energy",
      "isCorrect": false
    }
  ]
},
```


```
"users": [
  {
    "id": 1,
    "email": "achimov_alexandra@yahoo.com",
    "username": "alexaach",
    "password": "password123"
  },
  {
    "id": 2,
    "email": "test@yahoo.com",
    "username": "test123",
    "password": "password123"
  },
  {
    "username": "alexa",
    "password": "alexa",
    "email": "aalexa210598@gmail.com",
    "id": 3
  },
```

Step 2: implement the UI of the home page and the login and register functionalities

- the visual interface of the home page will be implemented via Bootstrap
- the login and register functionalities will be implemented via OOP, while all the account data (emails, usernames and passwords) will be stored in the database; a verification system will also be put into place; this system will be used to make sure that the data introduced by the user coincides with the data stored in the database (when the user logins in), that the username chosen by the user is not already taken (when the user creates an account), that the email the user tries to utilize is not already associated with another account etc.
- *time necessary to complete the step*: since I will make use of Bootstrap in designing the UI of the home page, I suppose I will finish this task in *1 day*



Step 3: implement the visual interface of the board page

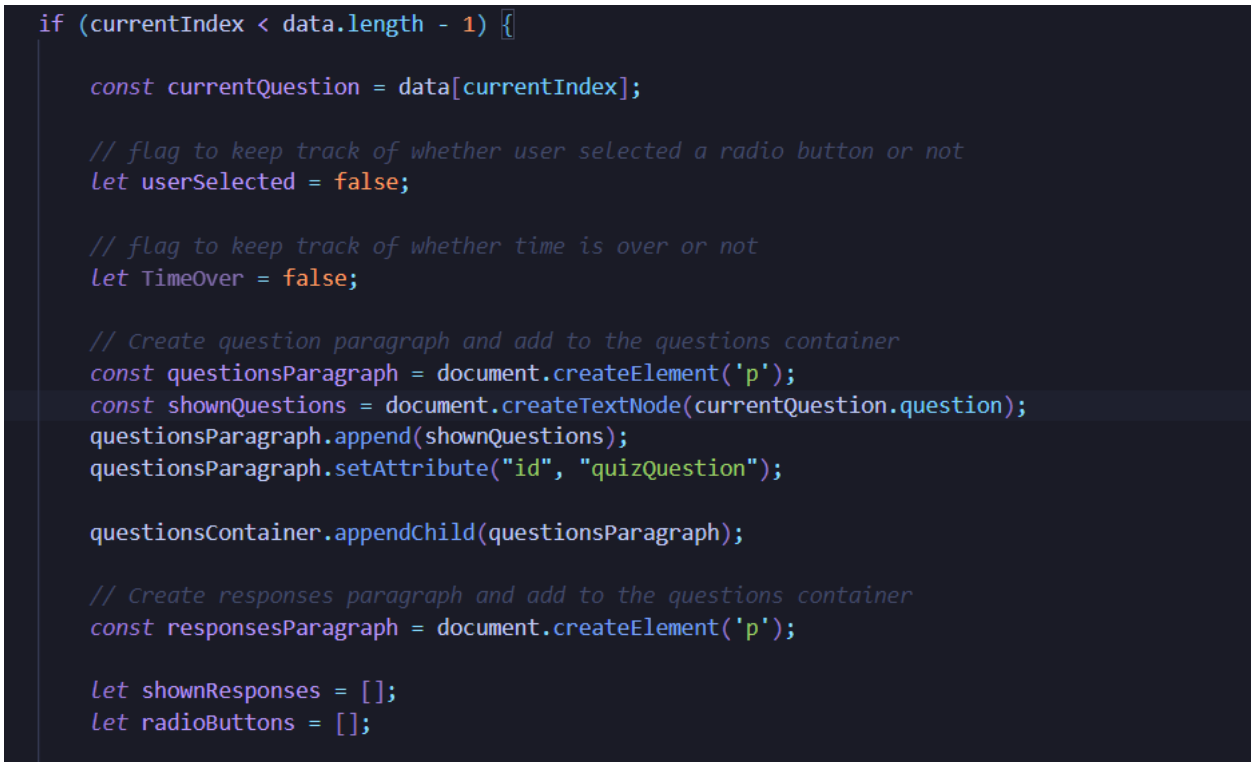
- this will be done making use of all the Bootstrap resources available
- *time necessary to complete the step*: *1 day*

Step 4: implement the functionalities found on the board page

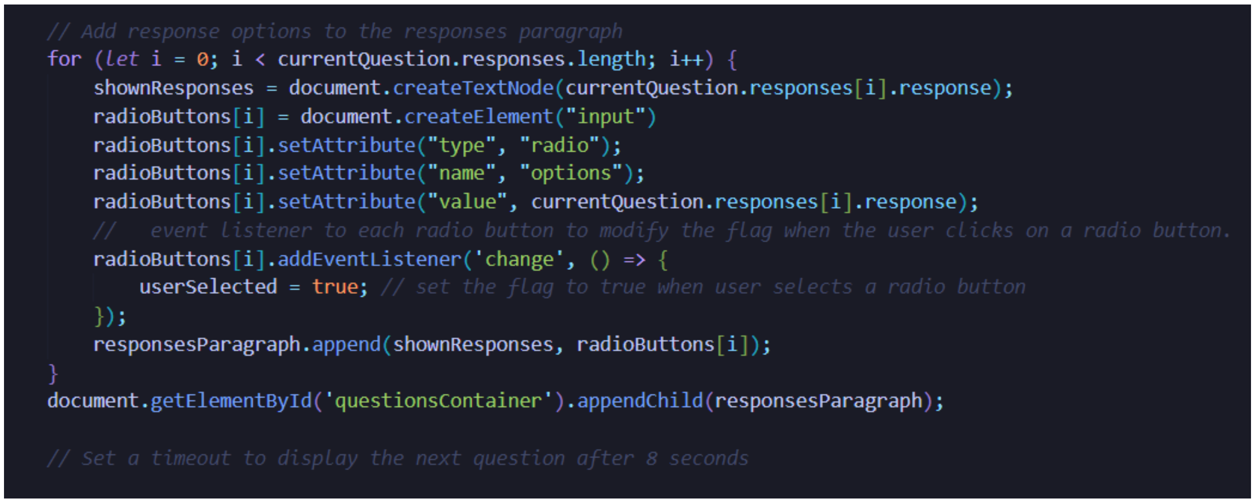
- in this step, the functionalities will be implemented progressively, starting from:
    - the functionalities found in the header: change language/change theme/send feedback
    - the functionalities found in the body of the page, which will be: account functionalities- change username/logout; previous quizzes functionalities: display previous completed quizzes along with the score obtained by the user, the category and the difficulty level chosen; and finally, start new quiz functionality, which will redirect the user to another page
    - the footer: the hyperlinks About Us, Terms and Conditions and Privacy Policy
- *time necessary to complete the step*: since this step will be the most extensive one, I assume it will take me *one week* to complete it

Step 5: implement the UI of the quiz page and the quiz itself

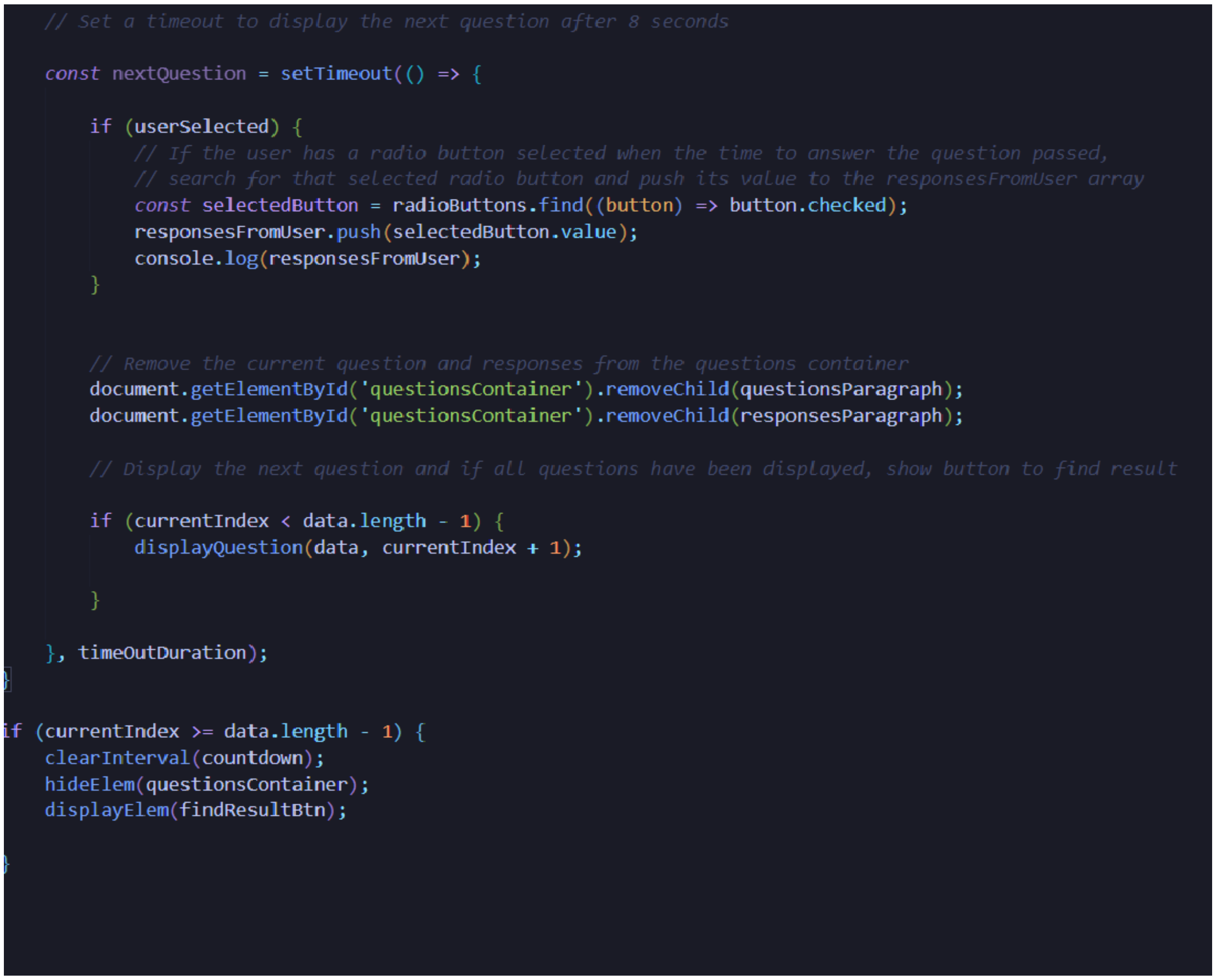
- *time necessary to complete the step*: I approximate it will take me *3 days* to complete this task



```javascript
if (currentIndex < data.length - 1) {

    const currentQuestion = data[currentIndex];

    // flag to keep track of whether user selected a radio button or not
    let userSelected = false;

    // flag to keep track of whether time is over or not
    let TimeOver = false;

    // Create question paragraph and add to the questions container
    const questionsParagraph = document.createElement('p');
    const shownQuestions = document.createTextNode(currentQuestion.question);
    questionsParagraph.append(shownQuestions);
    questionsParagraph.setAttribute("id", "quizQuestion");

    questionsContainer.appendChild(questionsParagraph);

    // Create responses paragraph and add to the questions container
    const responsesParagraph = document.createElement('p');

    let shownResponses = [];
    let radioButtons = [];
```



```javascript
// Add response options to the responses paragraph
for (let i = 0; i < currentQuestion.responses.length; i++) {
    shownResponses = document.createTextNode(currentQuestion.responses[i].response);
    radioButtons[i] = document.createElement("input")
    radioButtons[i].setAttribute("type", "radio");
    radioButtons[i].setAttribute("name", "options");
    radioButtons[i].setAttribute("value", currentQuestion.responses[i].response);
    //   event listener to each radio button to modify the flag when the user clicks on a radio button.
    radioButtons[i].addEventListener('change', () => {
        userSelected = true; // set the flag to true when user selects a radio button
    });
    responsesParagraph.append(shownResponses, radioButtons[i]);
}
document.getElementById('questionsContainer').appendChild(responsesParagraph);

// Set a timeout to display the next question after 8 seconds
```



```
    // Set a timeout to display the next question after 8 seconds

    const nextQuestion = setTimeout(() => {

        if (userSelected) {
            // If the user has a radio button selected when the time to answer the question passed,
            // search for that selected radio button and push its value to the responsesFromUser array
            const selectedButton = radioButtons.find((button) => button.checked);
            responsesFromUser.push(selectedButton.value);
            console.log(responsesFromUser);
        }


        // Remove the current question and responses from the questions container
        document.getElementById('questionsContainer').removeChild(questionsParagraph);
        document.getElementById('questionsContainer').removeChild(responsesParagraph);

        // Display the next question and if all questions have been displayed, show button to find result

        if (currentIndex < data.length - 1) {
            displayQuestion(data, currentIndex + 1);

        }

    }, timeOutDuration);
}

if (currentIndex >= data.length - 1) {
    clearInterval(countdown);
    hideElem(questionsContainer);
    displayElem(findResultBtn);

}
```

Used Technologies

GIT:

Git is a free and open source distributed version control system that tracks changes in any set of computer files, usually used for coordinating work among programmers. This tool will be used to keep track of the project.

GitHub:

GitHub is a cloud-based service that helps developers store and manage their code, as well as track and control changes to their code. This platform will be used to make the project available to the larger public.

HTML5:

HTML5 is a markup language used for structuring and presenting content on the World Wide Web. It is the fifth and final major HTML version that is a World Wide Web Consortium (W3C) recommendation. This language will be used to create the structure of the project.

CSS3:

CSS is the stylesheet language of web used to control the styling of a website. CSS3 is the latest version of CSS. This language will be used to implement the visual representation of the project and also to make its design responsive on several devices.

Bootstrap:

Bootstrap is the most popular HTML, CSS, and JavaScript framework for developing responsive, mobile-first websites. In this project, Bootstrap will be used where possible to implement responsive, specific UI Components.

JavaScript:

JavaScript is a prototype-based programming language that is used in Front-End Development to give functionality to a web application. In this project, the latest EcmaScript6 standard will be utilized to implement logic into the project and to make it dynamic.

TypeScript:

TypeScript is a strongly-typed superset of JavaScript that tops the latter programming language with new functionalities. TypeScript will be used to predefine the type of data utilized in order to make the code more readable, easier to understand, and less error-prone.

JSON Database:

A JSON document database is a type of database in which the data is stored as JSON. This type of database will be used to store all the questions of the project along with the users registered on the platform.

Angular:

Angular is a TypeScript-based, free and open-source Front-End framework used for developing web applications.

Visual Studio Code:

Visual Studio Code is a free source code editor that operates on Windows, macOS and Linux. This application will be used to code the project and its associated database.

Manual Testing

In terms of browsers, the site will be tested on Google Chrome, Safari and Mozilla Firefox, to make sure that it is responsive on the most used browsers. In terms of devices, it will be tested on different brands of smartphones (in principle, iPhone and Samsung devices).

In this project, the following testing methods will be used:

Cross Browser Testing

Cross browser testing refers to the practice of verifying that web applications work as expected across many different combinations of web browsers, operating systems, and devices. Though all web browsers support the common web standards (including HTML and CSS) developed by the World Wide Web Consortium (W3C), browsers can still render code in different ways. These disparities in appearance and function can arise from various factors, such as:

- Differences in the default settings on a browser or operating system (for example, the default font used by a browser).
- Differences in user-defined settings, such as screen resolution.
- Disparities in hardware functionality, which can lead to differences in screen resolution or color balancing.
- Differences in the engines used to process web instructions.
- Variations among clients in the version support for recent web standards, such as CSS3.

- The use of assistive technologies, such as screen readers.

Cross Browser Testing will be used to test if the project is responsive on Google Chrome, Mozilla Firefox and Safari.

Exploratory Testing

Essentially, exploratory testing tries to mimic individual end-users' personal freedom and choices. It is all about discovery, investigation, and curiosity. It is a simultaneous process in which testers go through the software of their own accord to get a sense of the quality of the user experience it provides.

Exploratory testing involves minimal planning. Testers create a test idea to understand what they are looking for and start exploring the software. Testers make spontaneous decisions about what features and actions to test, thus reflecting the individual interests of end-users. This type of testing will more likely find more issues and edge cases than you would through traditional test cases.

In this project, exploratory testing will be applied via emulating a user using the app for the first time and pressing on each and every button to discover what the app was to offer.

Performance Testing

Performance testing is the practice of evaluating how a system performs in terms of responsiveness and stability under a particular workload. Performance tests are typically executed to examine speed, robustness, reliability, and application size.

For a website, it is crucial that it renders the content in less than 2 seconds. Every loading and process that takes more than 2 seconds to finish means that the user will lose their patience and leave the website. For this reason, performance testing is crucial.

Responsiveness

Making use of Dev Tools, the UI part will be firstly designed for mobile (according to the mobile first principle) and then progressively adjusted to become responsive on tablets and laptops respectively.

Future Improvements

The project can be improved in the future in the following ways:

1. by implementing more quiz categories (for example, a Real Sciences Categories which will consist of the subcategories Informatics, Mathematics and Statistics),
2. by making it possible for the user to play the game with a computer,
3. by implementing a better system of scoring (for example, by taking the timer into account how fast they answer the question),
4. by implementing a "Forgot Password" functionality,
5. by implementing a more dynamic and aesthetically pleasing UI interface'

- by moving the project in Angular and rewriting it in Typescript,
- by implementing the "Delete Account" and "Change username/password/email address" functionalities,
- by implementing the "Change Language" and "Send feedback" functionalities
- by implementing the form validation that will check whether the username or email address that the user wants to use are not already taken,
- make the project fully responsive on tablet as well,
- better test the project;

Bibliography

1. *Performance testing, best practices, metrics & more*
 https://www.tricentis.com/learn/performance-testing

2. *Exploratory Testing: A Detailed Guide*
 https://www.browserstack.com/guide/exploratory-testing

3. *Cross Browser Testing Overview*
 https://www.datadoghq.com/knowledge-center/cross-browser-testing/#:~:text=Cross%20browser%20testing%20refers%20to,%2C%20operating%20systems%2C%20and%20devices

4. *Signup and Login*
 https://canjs.com/doc/guides/recipes/signup-simple.html

5. *Object Oriented Programming in JavaScript – Explained with Examples*
 https://www.freecodecamp.org/news/how-javascript-implements-oop/

6. *Angular docs*
 https://angular.io/docs

7. *TypeScript Documentation*
 https://www.typescriptlang.org/docs/

8. *JavaScript Classes – How They Work with Use Case Example*
 https://www.freecodecamp.org/news/javascript-classes-how-they-work-with-use-case/

9. *Promise*
https://developer.mozilla.org/enUS/docs/Web/JavaScript/Reference/Global_Objec
ts/Promise

10. *How to use promises*
https://developer.mozilla.org/enUS/docs/Learn/JavaScript/Asynchronous/Promise
s