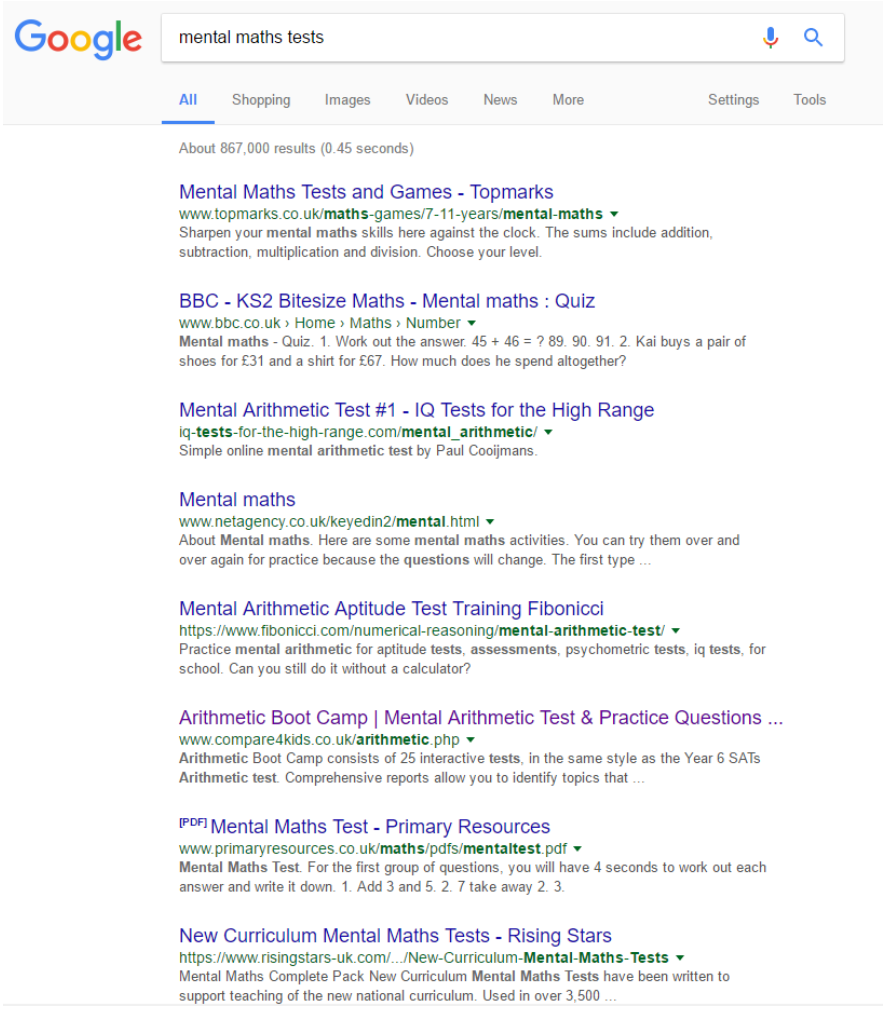
NEA: MentalMathsTest Website

Analysis:

Introduction

As students progress through school, improving their theoretical mathematical ability, emphasis is lost on the ability to quickly calculate and perform mathematical operations, mentally. My project aims to provide an online web application that allows users to test and improve their mental arithmetic ability. After a brief search, it seems as though such a simple concept is not fully provided on the internet, whereby a user can monitor their progress and their peer's progress too. A quick google search confirms that on first sight there are no websites solely dedicated to this purpose.



It is also worth noting that all of the first results are aimed at students currently in primary school or around that age. My project will be aimed at students currently in reception to year 13 (Since that's the extent of my mathematical knowledge). Also, it seems that most of the services listed by google generate pre-written questions with no means of variability. It's clear there's a lack of a fully functional AI to generate the questions based on the user's performance.

I will be creating this website with the aim of providing an accessible means of measuring and improving mental arithmetic ability, and providing data. Such as, how different demographics perform on the tests, in what areas of mathematics do most people struggle, etc. This data and analysis can be used by teachers, parents or students alike to track their progress. My Further Maths and Maths teacher Mr. Charles will be treated as a third-party, by which my site can be judged. Mr. Charles has suggested that in order to please him I should focus on making the number of different mathematical areas to potentially be assessed relatively large.

Mr Charles has also outlined his current solution to the problem that is the lack of online interactive mental arithmetic tests covering low to high level of mathematics. He uses a mix of “Kahoot!” and “mymaths”. What he has told me is that while Kahoot offers interactivity and an element of playfulness, there’s no support for mathematics (mathematical syntax, non multiple-choice, etc.). On the other hand, what mymaths offers in its dedication to online arithmetic, it lacks in customisation and interactivity with other users. He told me that in order to make a successful product I’d have to essentially combine the two concepts together.

Objectives

From what I have gathered through my third-party, research and understanding of what is possible; I have come up with some objectives that I feel are required in order for the product to be deemed a success:

- Allow people who access the website to make an account and thus be given special permissions.
- Securely store user’s data such as passwords and personal information.
- Be up to date legally (Privacy Policy, Terms and Conditions).
- Give users the option to securely store a minimalistic cookie on their machine so that their account can be remembered and automatically logged in.
- Make sure every page is aesthetically pleasing.
- Run advertisements on some pages to gain an income.
- Develop an Artificial Intelligence system that determines the nature of the next question given the state of the user’s answer to the last, and the numbers to be used pseudo-randomly.
- Allow people who aren’t registered to play a trial test.
- Derive a scoring system that accurately represents the user’s performance on any test.
- Include up to A2-Level further mathematics in the main test.
- Allow users to create their own tests.
- Allow these custom tests to be played by other users.
- Allow users to make their own worded questions in the custom tests.
- Have a “Leaderboard” page to track the scores submitted on a given test.
- Ensure the website is fully functional on 99% of devices (responsive).
- Have healthy SEO.
- Register the website under a suitable domain.
- Have a low page load time across the domain.

- Establish a mode whereby the questions go on forever until a wrong answer is given and difficulty increases constantly.

Some of these objectives were thought up by myself given the problem identified in the first paragraph. Others were desirable attributes thought up by my mum.

I'm well aware that I will have to learn lots of new programming and designing techniques that I am currently not aware of. I do, however, have experience in HTML, CSS, JavaScript (& jQuery), PHP, SQL and Python. The latter is unlikely to help me at all in this project, and my knowledge in PHP and SQL is negligible relative to the size of the task at hand.

Due to my third-party's particular interest in the number of mathematical areas, I have devised 15 in which I am confident in and I believe that are key to mathematical success throughout education. These are as follows:

1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Decimals
6. Exponentiation
7. Quadratics
8. Surds (Irrational roots)
9. Cubics
10. Factorials
11. Trigonometry
12. Logarithms
13. Complex Numbers
14. Harder Logarithms
15. Calculus

Concepts & Planning

What I noticed when outlining these topics was that I had to choose areas that were somewhat doable without writing down anything on paper and thus limiting my pool of mathematical areas, although I feel 15 will be more than enough to please Mr. Charles and each area of mathematics will contain 2-5 question formats. Resulting in approximately 53 different formats of question and of course a near infinite number of unique tests.

In order to deal with user accounts I'm well aware of the need for a local database on my server, so that I can access data on all the tests ever carried out, which demographics perform better than others etc. I'm also familiar with the role of session variables to keep users logged in through separate tabs or to transfer information when a user is being moved from one page to another.

I intend of having 3 main modes by which the user can be tested on:

- Normal Mode (The MMT), set out by me, aimed to increase mathematical broadness and agility.

- Custom Mode, where users play the tests created by other users. This encourages creativity and freedom within my website.
- Practice Mode, where users can play forever, until they get a question wrong. The questions will also get much harder as the user reaches higher levels. This means that users can focus on particular topics if they're struggling or if they need a confidence boost.

These 3 modes will be set in place to help accomplish whatever the user desires when being tested, whether that's an increase in mathematical ability, speed or knowledge – While inspiring creativity and freedom.

I feel that students would also desire a means by which to compare their mathematical ability to others of any specific demographic so that they can gain an understanding as to their ability relative to others. I think the best way to accomplish this would be to establish a page that shows the best ever scores on tests, in whatever mode, by whomever. To alter the filter, users could use a Javascript slider for age which will be recorded on sign up and a drop down menu for gender, then a text input for specific custom test tokens.

Having set out in my objectives to make the site relatively fast yet aesthetically pleasing, I'm aware of the challenges ahead as I'll have to find a balance between the two. Probably the best way to accomplish this feat is to aim to use Cascading Style Sheets (CSS) where possible because that'll be vector graphics calculated by the client browser/PC, as opposed to using uncompressed images all over the place which slow the site down horrendously. I feel as though this needs to be scoped early on in order to fulfil the outcomes of multiple objectives, since by doing so I'd be ticking the boxes of aesthetics, speed and a vast increase in SEO which leads to greater income too. Increasing user's speed among the site will drastically decrease retention time and as a direct result, more learning can be done and more data gathered.

Mr. Charles mentioned the concept of a healthy SEO in order to reap the benefits of increased user traffic on the website. I'm aware or some SEO concepts such as meta description tags, meta keyword tags, lightweight pages, correctly ordered header tags, etc. Of course content is the most key ingredient to a healthy SEO and as a result I believe a few paragraphs on the home page (index.php) describing how the website works and it's fundamentals could benefit both the client and the search engine bots, that crawl the internet to try understand the purpose of your site.

Documented Design

Obtaining a domain and hosting

As part of one of my objectives, I must obtain a domain that fits the purpose of the website. As a result, I went searching for any available domains to do with mental arithmetic tests. Pretty quickly I noticed the absence of any hosting upon the domain "www.mentalmathstest.com". Instantly I purchased this domain and some hosting to give me something to work with. With this considered, I decided the site should be called "MentalMathsTest", which could be occasionally shortened to the acronym "MMT". This will be useful for truncating information and possibly creating a favicon.

Developing a house style

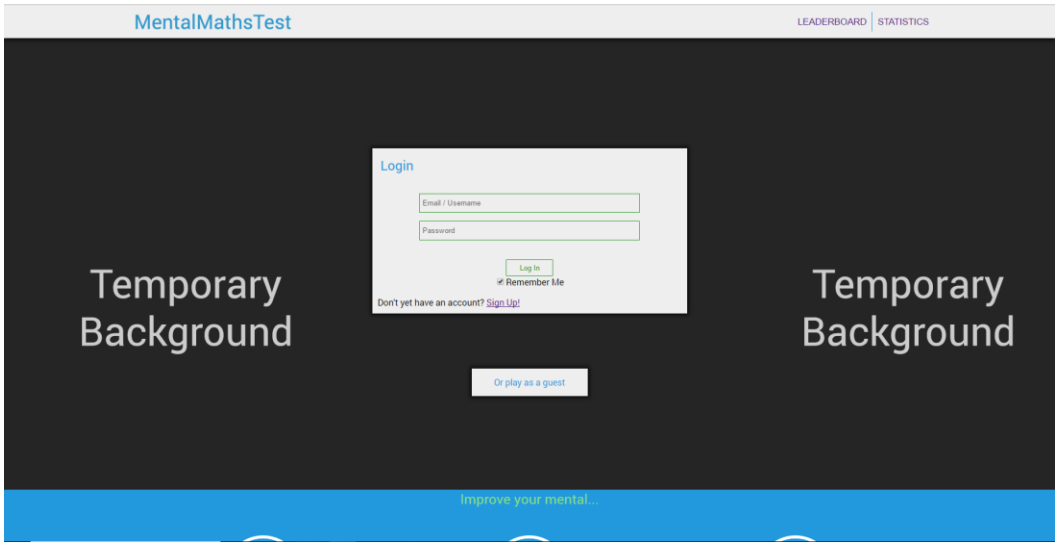
One of the objectives is to main a standard of good aesthetics. In order for this to be accomplished, I have some dedicated hexadecimal RGB codes that I believe complement each other well:

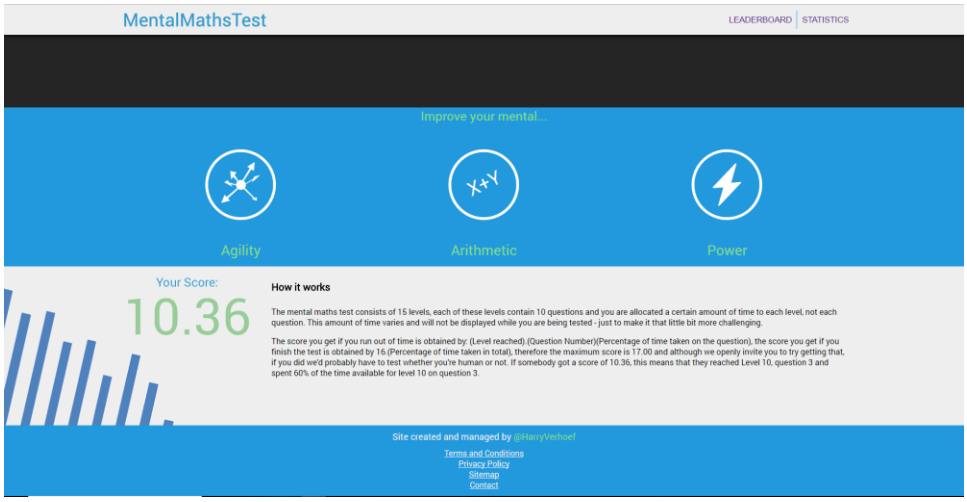
1 #29d (#2299dd)
2 #fff (#ffffff / white)
3 #8d8 (#88dd88)
4 #151515
5 #f93 (#ff9933)

Here they are demonstrated, respectively:

Initial Designing of index.php

Having developed this house style, I decided to design the landing page (index.php) in Photoshop. Obviously it's rather empty still and there are a lot of additions to be made when I start to code the website:



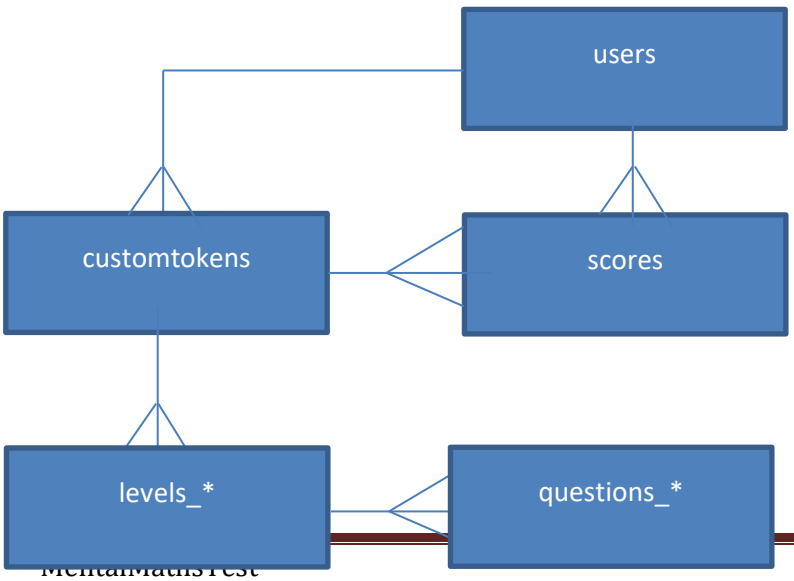


However, I feel this sets the tone as to what the website will look like and how the house style will be applied. I plan to have a compressed .mp4 video in place of the "Temporary Background", which I will edit so that it's shifted down in colour to complement the majority of bright components on the page. The background's position property is set to fixed. As a result, the background will not scroll with the rest of the site but will stay in place. However, due to a low *z-index* property, the blue "footer" part overlaps the background. I intend to have a lot of content below this screenshot that gives the visitor an insight into the purpose of the website and why they should join.

Data Storage

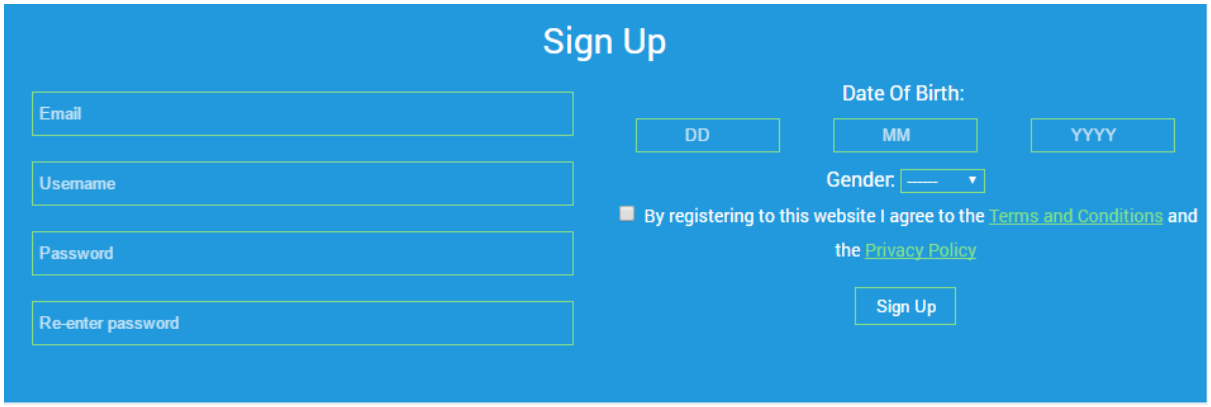
Thankfully my web-hosting providers give me unlimited access to infinite local databases, and the free usage of "phpMyAdmin" to administrate those databases. As a result, I did not need to run an SQL query to create my tables, I could instead work with a simple form that abstracts the entire process. I came up with the following tables: users, scores, customtokens, levels_* and questions_*. The asterisk indicates a token for a custom test. For example: levels_4baef stores data on the levels of custom test "4baef" and questions_4baef does the same but for the individual questions. I came up with these tables in the hopes of the database being in 3NF.

Here is an entity-relation diagram to represent how I believe my database will be linked together:



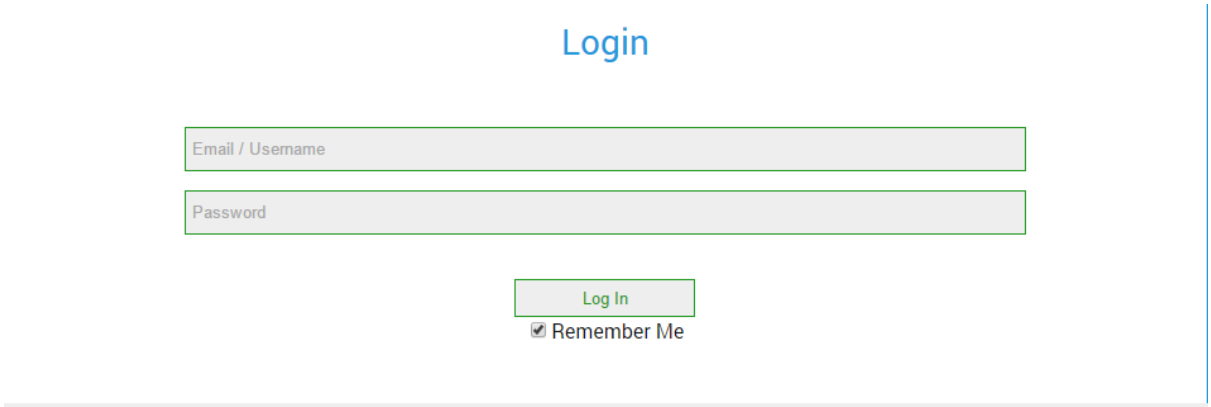
User Accounts

On my website I've implemented a user account system whereby a user can login to an account, attempt a variety of tests or create one and for their records to be saved and loaded on to their homepage. I was able to make this possible by doing the following:



Above is what I expect the HTML <form> sign up element displayed on index.php to look like where new clients can create an account. This form will use HTML5 attributes such as "required" so that client-side checks are made to ensure all data that is required is input. When the data from this form is submit, the data will be sent to my server over HTTPS (encrypted) where it will be cleansed and inserted into my database within the "users" table. During this sign-up process, a hash+salt algorithm will ensure that every user's account is incredibly secure, this algorithm is called bcrypt.

Hash + salt algorithms prevent passwords from being stolen by database administrators and malicious attackers using rainbow tables. They work by generating a long pseudo-random string, known as the "salt" which is almost definitely unique to that user. This salt will be concatenated to the password string and the resultant string will be put through a time-heavy hashing algorithm. The result of the hash and the token are both stored in the database.



This login interface will be displayed alongside the "sign up" one, on index.php and it is also a HTML form. Similarly to the "sign up" design, the data submit via the client-side form is sent as "POST" meaning the process of sending the data cannot be bookmarked, from there the server will receive the data and ensure everything's in accordance to the initial format, to prevent XSS attacks. One key feature of this login system is that it will attempt to login a user regardless of whether the user entered their username or email in the first input field. This can be achieved by first checking

whether the input is in the format of an email address using the PHP function *“filter_var($_POST["loginemail"], FILTER_VALIDATE_EMAIL)” where $_POST[“loginemail”]* is the decrypted string received by the server from the apparent input of the user in the “Email/Username field”. If so, then it will search the database for that email and if it is found, the salt and hash will be retrieved for further calculation. Similarly, if the input does not emulate that of an email address, it’ll search the database for that username since usernames are unique on my website and if one is found then it will also retrieve the salt and hash.

If the email/username search came back with no results, then it will output “Invalid login details.” So that the user doesn’t know which field was incorrect. Had the search come back with one result (the only other possible outcome), then the decrypted password string will be concatenated to the retrieved salt and the resultant string will be hashed. If this new hash is identical to the hash retrieved from the database then the user has input valid login details. It is imperative that the hash + salt algorithm used to ensure secure logins takes a considerable amount of processing power and more importantly time, so that a bot or macro cannot instantly make a login attempt one after another, brute-forcing the user’s password.

In order to further demonstrate my design of data storage, here is my predicted structure of all the tables:

users

USER_ID	USER_EMAIL	USER_NAME	USER_HASH	USER_DOB	USER_SEX	USER_AGEGROUP	USER_HIGHSCORE	USER_TOKEN

scores

SCORE_ID	USER_ID	LEVEL_REACHED	QUESTION_REACHED	TIME_TAKEN	FINAL_SCORE	DATE_RECORDED	TEST_TOKEN

customtokens

TOKEN_ID	TOKEN	CREATOR_ID

levels_*

*_LEVELID	NO_QUESTIONS	LEVELTIME

questions_*

*_QUESTIONID	QUESTION

To understand the denotation of the asterisk, refer to page 6.

Key Features

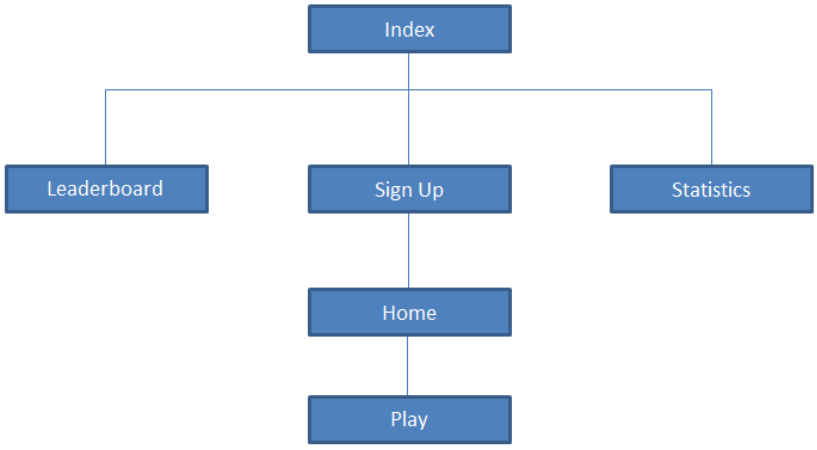
As I mentioned in the objectives, I intend on creating a system whereby a user can check a box and a cookie be stored on their local machine that securely lets the server know the user is already logged in. The way this will have to work is that on login, the user can tick a “Remember Me” box so that when the form is submitted to the server, the server generates a pseudo-random 16 byte string in hexadecimal format, this is the new token for the user. This token will replace whatever currently resides in the “USER_TOKEN” field of the user’s record. This token will then be hashed using SHA512 (Very secure – As of now, no pattern for collisions) and stored as a cookie on the client PC, only accessible via HTTPS.

Then, when a user may visit the site in the near future on the same device, the server will check if the “RememberMe” cookie resides on the user’s PC set by my website. If so, it will obtain the user’s ID from the cookie, from there it will query the server to find the user’s unhashed token. Then, if when the hash is applied to the result of the query, the result is identical to that of the token stored in the cookie, the user has a valid cookie. Otherwise, the session variable determining the users logged in state is set to the Boolean False. This system is incredibly secure, even more so since it will then proceed to generate another pseudo-random 16 byte hexadecimal string, set that as the user token in the table “users”, and update the cookie to contain the hashed version of that generated string. Therefore, whenever the user makes a successful login attempt whether through the “Remember Me” system, or a genuine manual login with “Remember Me” checked, the token on both the client and the server.

I could display the questions to the users through one of two ways the way I see it, I could a) generate the question and answer in plain text then parse this through a text to image functionality built into a php library and send the resultant image to the client. Or, b) Generate the question and answer in plain text, send the question back to the client in a special syntax that can be read by the lightweight JavaScript plugin “MathJax”, which uses LaTeX syntax to support abnormal characters and symbols over a multitude of browsers without using images. Both of these are aims to accomplish the ideal of impenetrable security. By this I mean that if I was to send back the question in plain text then users could easily program bots and macros to cheat the system and calculate the answer almost immediately using client-side simple JavaScript, I wouldn’t be able to do anywhere near the number of mathematical formats for questions I intend to if I displayed the questions in plain text, such as integration.

Option A	**Option B**
Much less processing power required from the server and as a result, less latency & faster.	Significantly better protected against bots and macros.
Better supported and less likely to throw an error.	More aesthetically pleasing (although can’t use css as effectively)

Here is a basic idea of what I picture the structure of the site to be like, through a sitemap:

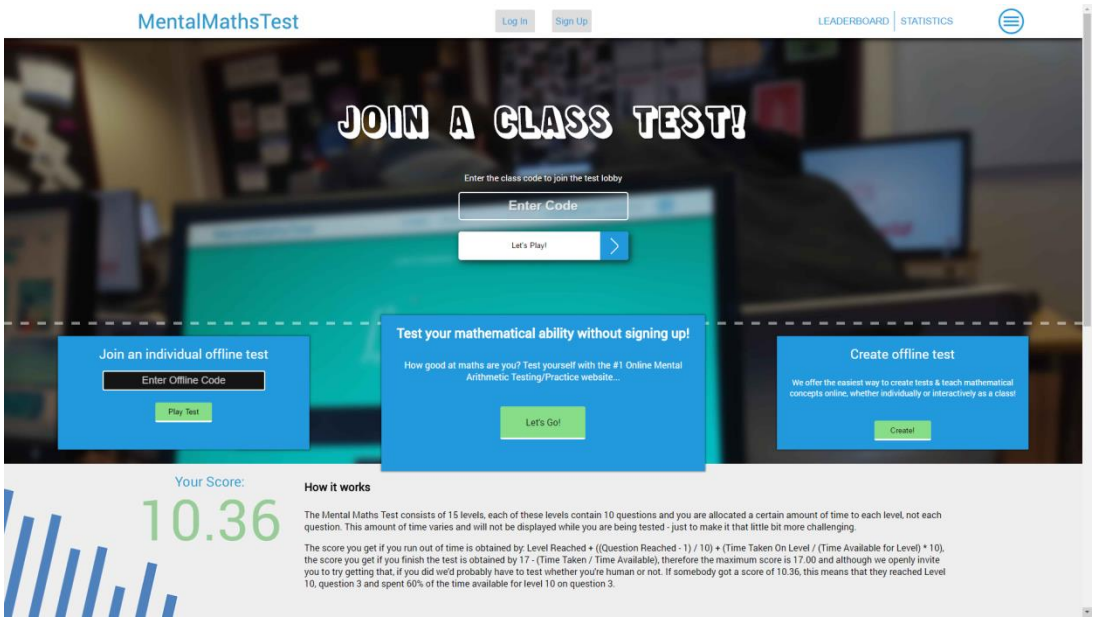


Obviously these are only the pages visible to the user, of which will be linked to 3 CSS documents each (header, unique body and footer).

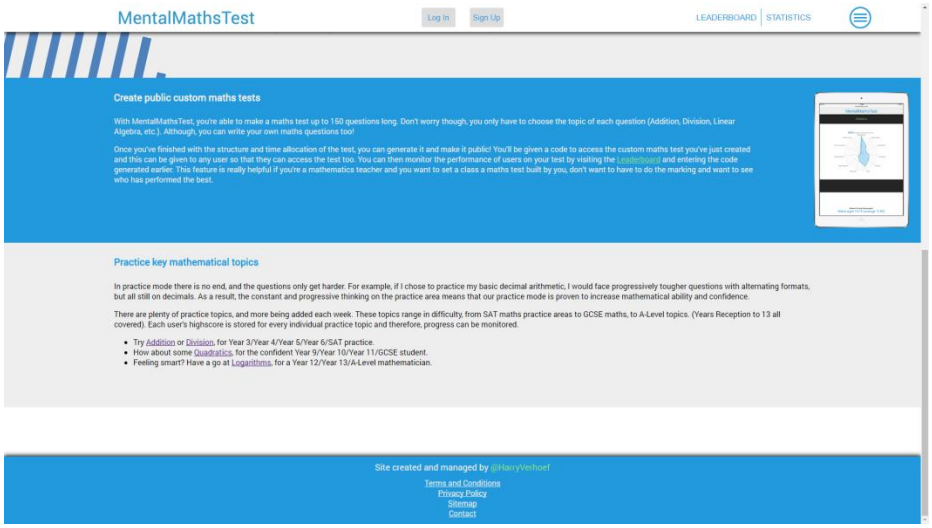
Technical Solution Walkthrough & More Documented Design:

I believe since the design stage that I have made some large changes to what I had planned and throughout this section I will also explain why I made said changes. Also, the structure of this section will be split into each page and then into each component which makes the page work.

1. Index.php



The "Join a class test" area and functionality is not a part of this coursework, since it was not involved in the plan and I've decided that this functionality would take way too long to develop. This is basically a functionality that would allow students of a class to connect to a session, and play together, similarly to "Kahoot!". However, I believe this is well beyond the complexity required for a complete A-Level project. As a result, this is only there because I will be developing it post coursework.



This is the page you get redirected to if you go to https://mentalmathstest.com and is thus considered the homepage of the website. I have tried to take this to my advantage, to try and entice the user to do any of many functionalities I have available. The first thing you'll notice I have changed from my plans is that instead of having a .mp4 file autoplay in the back, I actually have a still image and have drastically altered the layout of the page.

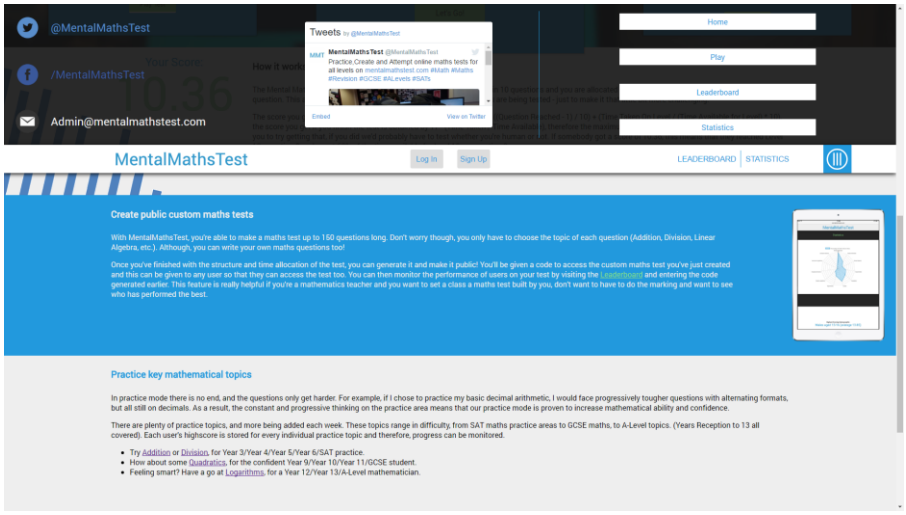
The Header

The header is on every single one of my pages and is linked to the same css file across the board – header.css. Here are the following style attributes of the header div:

```
.header {
    position: fixed;
    top: 0; left: 0;
    margin: 0; padding: 0;
    width: 100%;
    height: 60px;
    font-family: "Monsterrat", sans-serif;
    box-shadow: 0px 2px 13px #000;
    background-color: #fff;
    transition: top 0.4s;
    transition-delay: 0.4s;
    z-index: 1500;
}
```

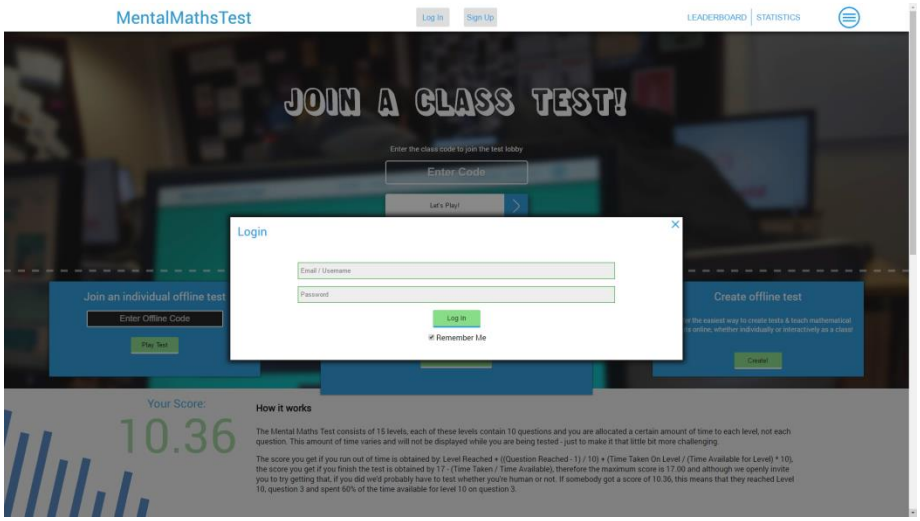
As you can see this is very basic css. I've already touched on the position fixed factor, the top & left = 0 is just to put the header in the top left corner. A box shadow coloured jet back Is directed down onto the page below for aesthetic effect. I included the transition style elements because when a user clicks the div "menuIcon", a menu opens up.

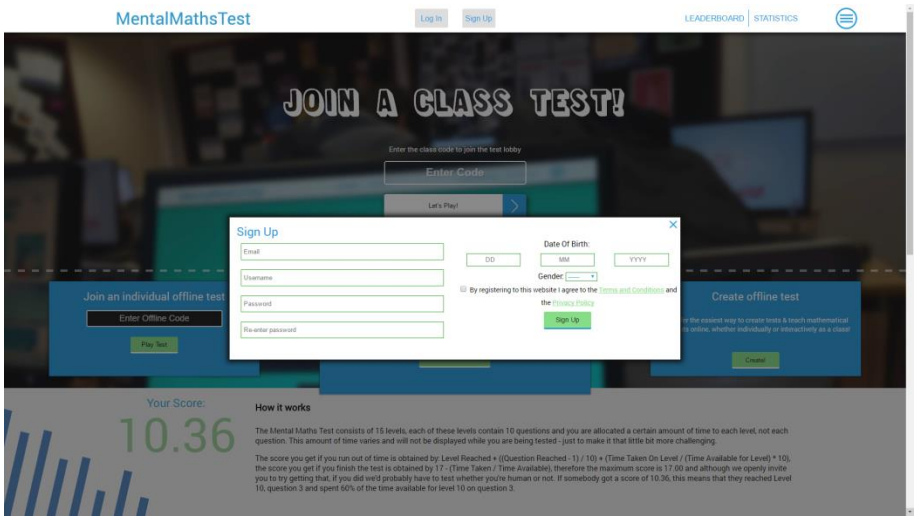
This is the div ".menuIcon". This is actually not an image but is in fact a parent div with 3 children divs and a border-radius of 50%, making it appear as a circle. When the user hovers over the div the colours are inverted and when the user clicks the div, it rotates and a menu opens up providing extra navigation and usability, with a link to social media to increase SEO (backlinks help improve SEO). I chose to make this using CSS for more freedom with hover and click effects, it also reduces page load time as it is considered a highly efficient vector object.



Instead of having a separate page for users to sign up on, I decided to attach it the button to the top of the header alongside the log in button so that users can log in from whatever page they're on.

Once clicked the log in and sign up functionalities look like this, respectively:





Both very similar, since they share the same parent div but this was accomplished using mainly a combination of CSS3 animations and JavaScript event handlers:

```
$("#LoginButton").click(function() {
    $("#UserAccountDialogHeader").html("Login");
    $(".active").removeClass("active");
    $("#UserAccountLoginForm,.userAccountDialog,.finishedOverlay").addClass("active");
});
$("#SignUpButton,.signUpButton").click(function() {
    $("#UserAccountDialogHeader").html("Sign Up");
    $(".active").removeClass("active");
    $("#UserAccountSignUpForm,.userAccountDialog,.finishedOverlay").addClass("active");
});
```

On the screenshot above it is evident I am using jQuery to handle the event whereby either of the divs are clicked individually using a very similar algorithm consisting of a literal function that adds and removes the “active” class from certain divs. This has an effect on the styling:

```
.finishedContent.active,.guestFinishedContent.active,.userAccountDialog.active {
    display: block;
    animation-name: finished;
    animation-duration: 0.55s;
    animation-fill-mode: forwards;
}
```

```
@keyframes finished {
    0% {
        transform: scale(0);
    }
    65% {
        transform: scale(1.07);
    }
    100% {
        transform: scale(1);
        box-shadow: 0px 0px 15px 5px #111111;
    }
}
```

Note that the log in and sign up buttons are not needed for users that are already logged in. Thus, I have programmed it in php so that if the user is considered to be logged in , different buttons appear.

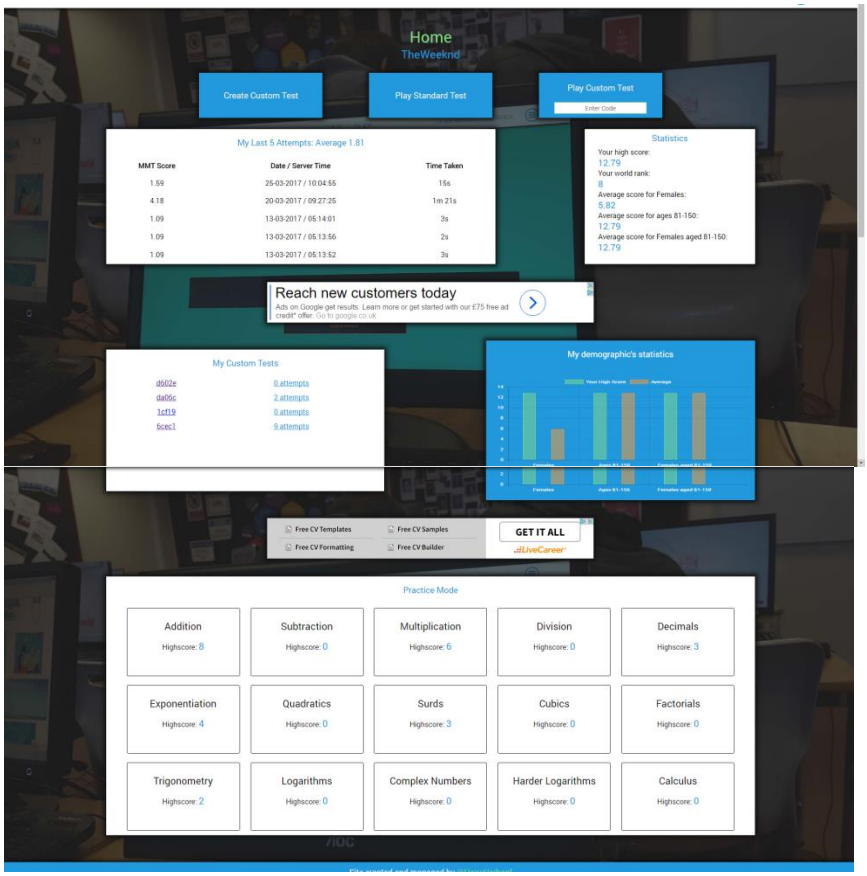
```php
<?php
session_start();
include "connect.php";
if (isset($_COOKIE["rememberMe"])) {
    include "connect.php";
    $query = mysqli_stmt_init($db);
    mysqli_stmt_prepare($query,"SELECT USER_TOKEN, USER_SALT FROM users WHERE USER_ID=?");
    mysqli_stmt_bind_param($query,"s",$id);
    $id = mysqli_real_escape_string($db,explode("_",$_COOKIE["rememberMe"])[0]);
    mysqli_stmt_bind_result($query,$token,$salt);
    mysqli_stmt_execute($query);
    mysqli_stmt_fetch($query);
    mysqli_stmt_close($query);
    if (hash("sha512",$token.$salt) === explode("_",$_COOKIE["rememberMe"])[1]) {
        include "rememberme.php";
        $_SESSION["loggedIn"] = true;
        $_SESSION["userID"] = $id;
    } else {
        $_SESSION["loggedIn"] = false;
        $_SESSION["userID"] = null;
    };
};
if ($_SESSION["loggedIn"] == true && isset($_SESSION["userID"])) {
    include "connect.php";
    $query = mysqli_stmt_init($db);
    mysqli_stmt_prepare($query,"SELECT USER_NAME FROM users WHERE USER_ID=?");
    mysqli_stmt_bind_param($query,"s",$id);
    $id = $_SESSION["userID"];
    mysqli_stmt_bind_result($query,$username);
    mysqli_stmt_execute($query);
    mysqli_stmt_fetch($query);
    mysqli_stmt_close($query);
};
?>
<!DOCTYPE html>
```

Above is an algorithm to determine whether the user can be logged in through the use of the cookie functionality explained earlier on page 8. Then, if the user is to be deemed logged in, it will display a home and log out button through this algorithm:

```php
if ($_SESSION["loggedIn"] == true && isset($_SESSION["userID"])) {
    echo "
    <div class='loggedInNav'>
        <ol>
            <a href='https://mentalmathstest.com/home'><li>Home</li></a>
            <a href='https://mentalmathstest.com/signout.php'><li>Sign Out</li></a>
        </ol>
    </div>
    ";
} else {
    echo "
    <div class='loggedInNav'>
        <ol>
            <li id='LoginButton'>Log In</li>
            <li id='SignUpButton'>Sign Up</li>
        </ol>
    </div>
    ";
};
```

The replacement if the user is logged in…

home.php



These images demonstrate accurately how home.php Is layed out. I wanted this to be the area you can branch off and begin any functionality. As you can see I obtain the user's username and output it one to the screen below the "Home" <h1> tag. In this case, it's "TheWeeknd". This was accomplished through obtaining the user's ID from a session variable and finding the corresponding user name from the local db on the server and then echo-ing back for the client. It is also immediately evident from these pictures that I have ads up and running on the website, I achieved this through a scheme called "Google AdSense" and had to re-apply twice due to lack of content until they accepted me having bulked up on visible content.

```
switch(true) {
    case $age < 11:
        $lowerLimit = 0;
        $upperLimit = 10;
        break;
    case $age < 13:
    $lowerLimit = 11;
        $upperLimit = 12;
        break;
    case $age < 15:
        $lowerLimit = 13;
        $upperLimit = 14;
        break;
    case $age < 17:
        $lowerLimit = 15;
        $upperLimit = 16;
        break;
```

Included on home.php is a list of the users last 5 Normal-mode tests and their certain statistics such as date recorded and time taken, along with score. Alongside that element is another div that displays certain catered statistics for the user. This was rather hard to make. If the user identifies as a female aged 81, as in this case, the average score for females, ages 81-150 and females aged 81-150 are shown to give the user an idea of how each demographic has an effect on the average score. Here is a brief look at the PHP switch statement that works out which agegroup the user is in:

This screenshot of the switch statement is only about a third of the entire statement but since the actual thing is so large and repetitive, it's

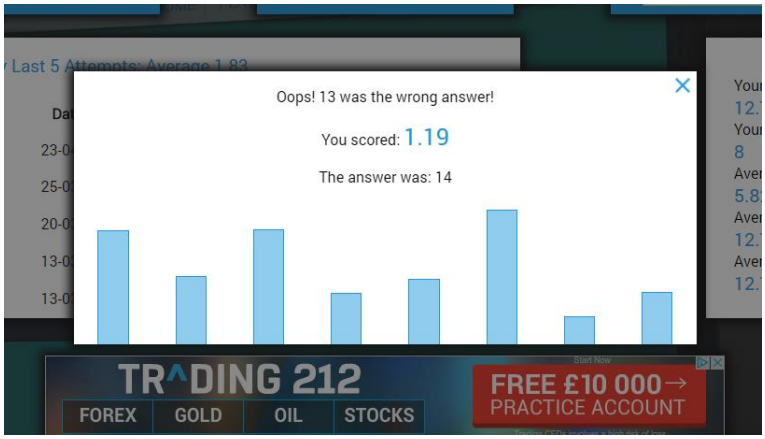
more convenient. I chose to allocate this task a switch statement rather than an if-statement because switch statements are more compatible when the conditions in question are so similar.

Below the second horizontal ad is the practice area, where a user can select an area of mathematics and start practicing it. In each case the user's specific high score is retrieved from the db on the server and sent back to the user.

```
<li><a href="https://mentalmathstest.com/play?t=Addition"><div class="practiceTopic"><h4
    resultArray[0]; ?></span></p></div></a></li>
<li><a href="https://mentalmathstest.com/play?t=Subtraction"><div class="practiceTopic">
    echo $resultArray[1]; ?></span></p></div></a></li>
<li><a href="https://mentalmathstest.com/play?t=Multiplication"><div class="practiceTopi
"><? echo $resultArray[2]; ?></span></p></div></a></li>
<li><a href="https://mentalmathstest.com/play?t=Division"><div class="practiceTopic"><h4
    resultArray[3]; ?></span></p></div></a></li>
<li><a href="https://mentalmathstest.com/play?t=Decimals"><div class="practiceTopic"><h4
    resultArray[4]; ?></span></p></div></a></li>
```

The above is part of the code block that displays the highscore to the user.

When a user completes a test and is logged in, they are redirected back to home.php with a few URL parameters in, so that this happens:



This allows the user to recollect on their error and gain information on how they did it wrong. This was accomplished again by using a combination of jQuery and CSS3 to add classes and delicately parse parameters from the URL to the div. For example, the URL in this case would be:

https://mentalmathstest.com/home?s=wrong&1.19&14&13

This tells home.php that the status is the user has come back from the test and they got the question wrong, they scored 1.19, the correct answer was 14 and they suggested 13. Home.php uses "&" as delimiters since they're a safe option for URL's.

createtest.php

This page was without doubt the most complex to complete. This page allows users to click a mathematical area, this area will then be set as a question within a level. A level can contain up to 10 questions and a specific time limit is allocated to each level. There can be as many as 15 levels in a test and if the user chooses, they can create their own question with their own answer. Once complete, the user can click "Generate test" and they will be provided with a token that ca be used at any time by anyone visiting the site to partake in the custom test. User's scores can also be monitored on the custom test through the Leaderboard.php page but we'll get to that later. I also had to write my own search algorithm, and a means of deleting levels and questions alike.

I abstracted the system and gave each mathematical area a number from 1 to 28, when a question is clicked it creates another div in the assigned section, dependant on the current level the user has selected. This is the JavaScript algorithm I used to add questions:

```
$(".questionSelect > div").click(function() {
    $(".current .dragQuestion.active").html(
        "<div id='" + $(this).attr("id") + "Active" + "' class='" + $(this).attr("class") + "'>
});
$("#AddCreatedQuestionButton").click(function() {
    var numCreated = $(".custom").length;
    $(".current .dragQuestion.active").html(
        "<div class='custom'>\
            <h4 class='topicHeader'>Custom Question</h4>\
            <ol class='topicOl'>\
                <li><p class='topicQuestion'>Question: " + $("#CreatedQuestion").val() + "</p></li>\
                <li><p class='topicAnswer'>Answer: " + $("#CreatedAnswer").val() + "</p></li>\
            </ol>\
            <input type='hidden' value='" + $("#CreatedQuestion").val() + "&&&" + $("#CreatedAnswer")
                "' />\
            <img class='closeIcon' src='images/CloseIcon.png' alt='Remove Question'>\
        </div>"
    );
    $(".current .dragQuestion.active").addClass("inactive");
    $(".current .dragQuestion.active").removeClass("active");
    if ($(".current .dragQuestion").length < 10) {
        $(".current .dragQuestion.semi-active").addClass("active");
        $(".current .dragQuestion.semi-active").removeClass("semi-active");
        $(".current .dragQuestion.active").after("<div class='dragQuestion semi-active'><p>Next Ques
    } else if ($(".current .dragQuestion").length === 10) {
        $(".current .dragQuestion.semi-active").addClass("active");
        $(".current .dragQuestion.semi-active").removeClass("semi-active");
    };
});
```

And the algorithm I devised for searching purposes utilising jQuery's .each() function:

```
$("#SearchBar").keyup(function() {
    if ($("#SearchBar").val().length >= 3) {
        var searchQuery = $("#SearchBar").val().toLowerCase();
        var searchQuery = searchQuery.replace("adding"," addition ");
        var searchQuery = searchQuery.replace("plus"," addition ");
        var searchQuery = searchQuery.replace("minusing"," subtraction ");
        var searchQuery = searchQuery.replace("minus"," subtraction ");
        var searchQuery = searchQuery.replace("times"," multiplication ");
        var searchQuery = searchQuery.replace("product"," multiplication ");
        var searchQuery = searchQuery.replace("dividing"," division ");
        var searchQuery = searchQuery.replace("easy"," simple ");
        var searchQuery = searchQuery.replace("indice"," exponent ");
        var searchQuery = searchQuery.replace("log"," logarithms ");
        var searchQuery = searchQuery.replace("integral"," integration ");
        var searchQuery = searchQuery.replace("derivative"," differentiation ");
        var searchQuery = searchQuery.replace("trigonometry"," trigonometric ");
        var searchQuery = searchQuery.replace("bodmas"," bidmas ");
        var searchQueryArray = searchQuery.split(" ");
        searchQueryArray = searchQueryArray.filter(Boolean);
        $(".questionSelect > div").addClass("search-inactive");
        $(".questionSelect > div").each(function() {
            for (var i = 0; i <= searchQueryArray.length - 1; i++) {
                if ($(this).find(".topicHeader").html().toLowerCase().search(searchQueryArray[i]) >= 0) {
                    $(this).removeClass("search-inactive");
                };
            };
        });
    } else {
        $(".search-inactive").removeClass("search-inactive");
    };
});
```

However, I was unaware of how difficult it would be to set up the server-side part too. I had kept to the exact plan when it came to database table structure, but it turns out it's harder to calculate what level the user is in than I previously expected. Here is what I managed to derive:

```
elseif ($result = mysqli_query($db,"SELECT * FROM levels_".mysqli_real_escape_string($db,$_SESSION["customID"]))) {
  while ($levelRow = mysqli_fetch_row($result)) {
      $level[$levelRow[0]] = array($levelRow[1],$levelRow[2]);
      //level[1] === [Number of Questions, Time for Level];
  };
  $levelFound = false;
  $sum = 0;
  $index = 1;
  while ($levelFound === false) {
      $sum = $sum + $level[$index][0];
      // Cumulative questions between level 1 and level index;
      if ($index > 15) {
          break;
      };
      if (intval($_SESSION["currentcustomQuestion"]) <= $sum) {
          $outputArray[0] = $index;
          $outputArray[1] = ($_SESSION["currentcustomQuestion"] - $sum + $level[$index][0]);
          $levelFound = true;
      };
      $index = $index + 1;
  };
```

This complex iteration structure is very efficient in calculating what level the user is in using the intervals provided from the DB and thus the styling effects can take place on play.php, even when the user is playing a custom test.

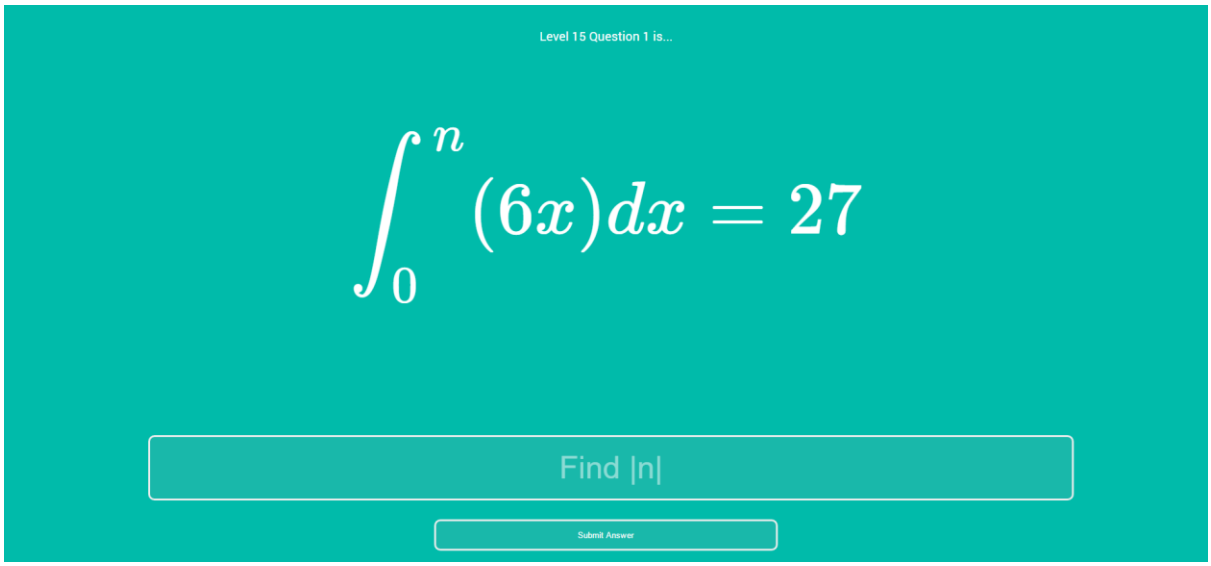
TOKEN_ID	TOKEN	CREATOR_ID
92	85714	73
93	6977a	104
94	a78b9	73
95	4ae3d	105
96	4cfc4	0
97	a53a9	0
98	4749a	0
99	feff7	0
100	d8fde	0

This is a screenshot from phpMyAdmin that demonstrates the table structure of "customtokens". Where CREATOR_ID = 0 is where the user who created the test is not logged in and thus deemed a guest. This is how users are able to see their custom tests listed in front of them on home.php.

play.php

The file where all of the testing takes place, I managed to utilise jQuery's asynchronous JavaScript and XML functionality in order to have a constant back and forth communication between the server and the client without refreshing the page. This is what the page looks like when you start:

Although, like I planned, after 150 or so pseudo-randomly generated questions using my AI, it gets a little harder:



Also, notice how the background-color css property has changed. This was accomplished through a very simple JavaScript function I created that changes the colour of the background every level, this simple line of code, however, is hidden among complex AJAX of dataType json so that an array could be passed from the server to the client.

```
$.ajax({
    url: "getQuestion.php",
    data: {answer: $("#Answer").val(),answerbutton: $("#AnswerButton").val()},
    dataType: "json",
    type: "post",
    success: function(output) {
        console.log(output);
        $("#LevelCounter").html(output[0]);
        $("#QuestionCounter").html(output[1]);
        $(".bodyContent").css("background-color","#"+["151515","284","b07070","664444","0
        if (output[2] === "wrong" && output[4] === "notGuest") {
            window.location = "https://mentalmathstest.com/home?s=wrong&" + output[8][0]
        } else if (output[2] === "wrong" && output[4] === "guest") {
            window.location = "https://mentalmathstest.com?s=guestwrong&" + output[8][0]
        };
```

```
};
if (output[3] === "end" && output[4] === "notGuest") {
    window.location = "https://mentalmathstest.com/home?s=end&" + output[8][0];
} else if (output[3] === "end" && output[4] === "guest") {
    window.location = "https://mentalmathstest.com?s=guestfinished";
};
$("#Answer").attr("placeholder",output[6]);
$("#Question").html(output[7].replace("\"",""));
setTimeout(function(){ $("#Question").addClass("active"); }, 400);
MathJax.Hub.Queue(["Typeset",MathJax.Hub], function() {
    $("#Question").css("margin-top",((($(window).height()-380)-$("#Question").height()+(
});
```

So, the AJAX takes in the user's input, sends it to the server and in particular: "getQuestion.php" this file is the AI that decides what question to output depending on the user's performance. It has a combined total of over 1000 lines of code. This file also caters for if the user is using a custom test, if the user is in practice mode or if the user is in normal mode. It caters for if the question is wrong and if the test has ended too.

For example, here is the code that generates that generates the pseudo-random question for level 12 question 1:

```
elseif ($_SESSION["currentQuestion"][0] === 12) {
  if ( in_array($_SESSION["currentQuestion"][1],array(1))) {
      $practiceCoefficient = $practiceCoefficient > 7 ? 7 : $practiceCoefficient;
      $rand1 = rand(0,5 + $practiceCoefficient);
      $answer = gmp_intval(gmp_fact($rand1));
      $_SESSION["answer"] = $answer;
      $outputArray[7] = "\[".$rand1."!\]";
```

And this is a snippet of the code for level 11 questions 1-3:

function checkFactor($b,$c,$d,$n) {

*if ((pow($n,3) + ($b * pow($n,2)) + ($c * $n) + $d) === 0) {*

return 1;

} else {

```
                return 0;
        };
};
$b = rand(-3,3 + round($practiceCoefficient/5));
$c = rand(-3,3 + round($practiceCoefficient/5));
$d = 0 - pow($answer,3) - ($b * pow($answer,2)) - ($c * $answer);
while ((checkFactor($b,$c,$d,-3)+checkFactor($b,$c,$d,-2)+checkFactor($b,$c,$d,-1)+checkFactor($b,$c,$d,1)+checkFactor($b,$c,$d,2)+checkFactor($b,$c,$d,3)+checkFactor($b,$c,$d,0)) !== 1) {
        $b = rand(-3,3 + round($practiceCoefficient/5));
        $c = rand(-3,3 + round($practiceCoefficient/5));
        $d = 0 - pow($answer,3) - ($b * pow($answer,2)) - ($c * $answer);
};
```

This code is generating factors of a random cubic polynomial.

There is a method of generating this without an iteration structure such as a while loop, however this is the formula for solving cubics of form ax^3

$$
\begin{aligned}
x_1 = & -\frac{b}{3a} \\
& -\frac{1}{3a}\sqrt[3]{\tfrac{1}{2}\left[2b^3-9abc+27a^2d+\sqrt{\left(2b^3-9abc+27a^2d\right)^2-4\left(b^2-3ac\right)^3}\right]} \\
& -\frac{1}{3a}\sqrt[3]{\tfrac{1}{2}\left[2b^3-9abc+27a^2d-\sqrt{\left(2b^3-9abc+27a^2d\right)^2-4\left(b^2-3ac\right)^3}\right]} \\
x_2 = & -\frac{b}{3a} \\
& +\frac{1+i\sqrt{3}}{6a}\sqrt[3]{\tfrac{1}{2}\left[2b^3-9abc+27a^2d+\sqrt{\left(2b^3-9abc+27a^2d\right)^2-4\left(b^2-3ac\right)^3}\right]} \\
& +\frac{1-i\sqrt{3}}{6a}\sqrt[3]{\tfrac{1}{2}\left[2b^3-9abc+27a^2d-\sqrt{\left(2b^3-9abc+27a^2d\right)^2-4\left(b^2-3ac\right)^3}\right]} \\
x_3 = & -\frac{b}{3a} \\
& +\frac{1-i\sqrt{3}}{6a}\sqrt[3]{\tfrac{1}{2}\left[2b^3-9abc+27a^2d+\sqrt{\left(2b^3-9abc+27a^2d\right)^2-4\left(b^2-3ac\right)^3}\right]} \\
& +\frac{1+i\sqrt{3}}{6a}\sqrt[3]{\tfrac{1}{2}\left[2b^3-9abc+27a^2d-\sqrt{\left(2b^3-9abc+27a^2d\right)^2-4\left(b^2-3ac\right)^3}\right]}
\end{aligned}
$$

As you can see, this incorporates complex numbers, of which cannot be simply computed. Thus, I decided to implement a while loop that iterates until atleast one of [-3,-2,-1,0,1,2,3] is a factor of the cubic polynomial.

The answer to the question is always calculated algebraically from the pseudo-random parameters generated earlier on in the indented block of code. This answer is then stored in a session variables so that when the next iteration of AJAX takes it's turn the answer can be checked using checkAnswer.php:

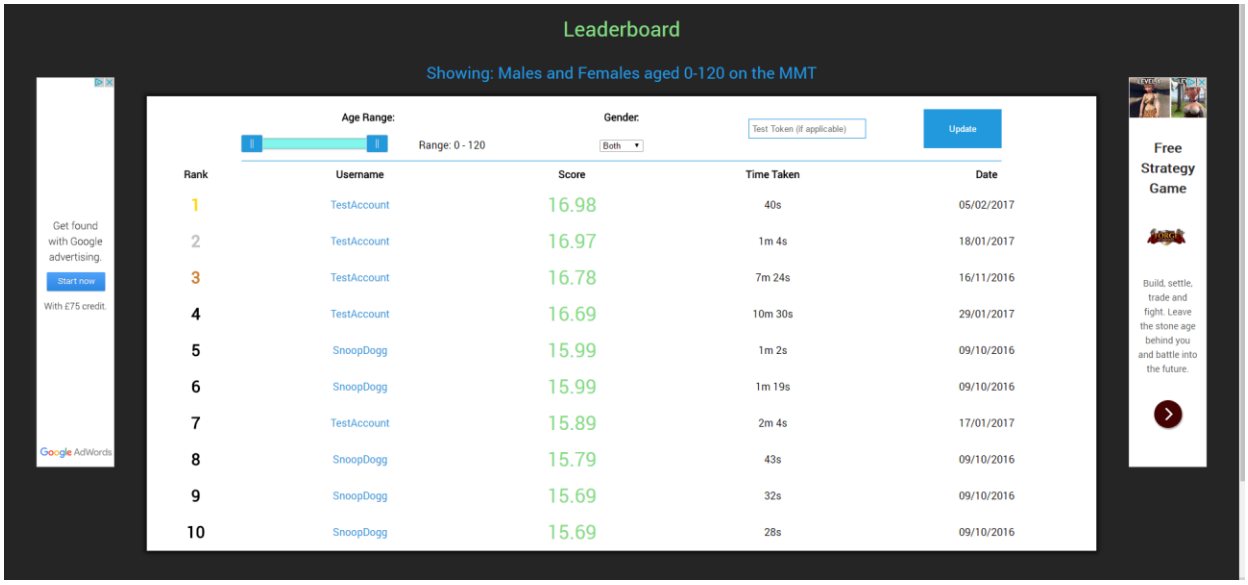
This increments the normal-mode and custom question counter given that the answer is correct:

```
$valid = true;
$_SESSION["currentQuestion"][1] = $_SESSION["currentQuestion"][1] + 1;
$_SESSION["currentcustomQuestion"] = intval($_SESSION["currentcustomQuestion"]) + 1;
if ($_SESSION["currentQuestion"][1] > 10) {
    $_SESSION["currentQuestion"][0] = $_SESSION["currentQuestion"][0] + 1;
    $_SESSION["currentQuestion"][1] = 1;
    if ($_SESSION["currentQuestion"][0] > 15 && !$_SESSION["customTest"]) {
        $end = true;
    };
};
```

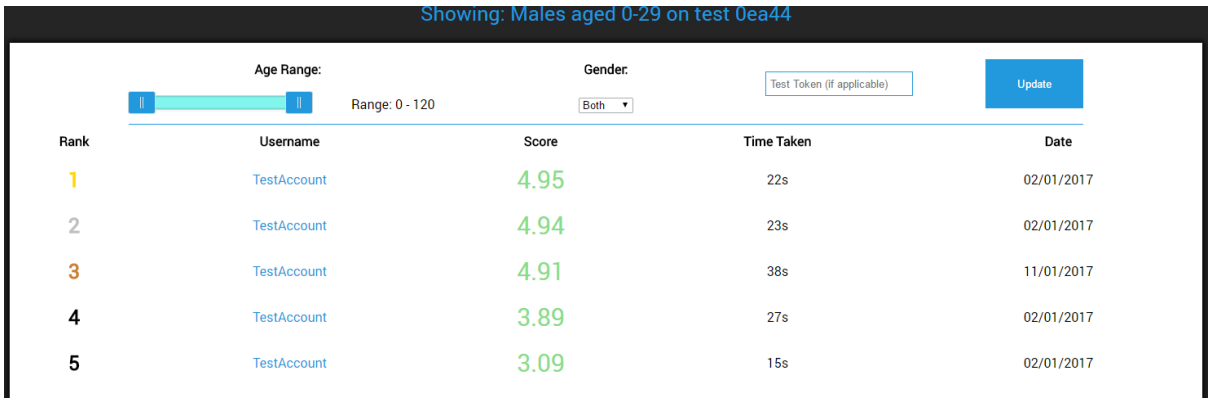
This is a snippet of code for when the question the user has answered to is a special question. For example if I am required to find the factors of an^2+bn+c=0, there may well be two acceptable answers so I have to check whether the question was of such format and then strip the answer of any inconsistencies such as capital letters, spaces or wrongly-arranged factors to check if the answer is essentially correct:

```
if ($_SESSION["currentQuestion"][0] === 10 && in_array($_SESSION["currentQuestion"][1], range(7,10,1))) {
    if (in_array((double)$answerArray[0],$_SESSION["answer"]) && in_array((double)$answerArray[1],$_SESSION["answer"]) && $
        $correct = true;
    } else {
        $correct = false;
        $_SESSION["answer"] = $_SESSION["answer"][0]."_".$_SESSION["answer"][1];
    };
} elseif ($_SESSION["currentQuestion"][0] === 11 && in_array($_SESSION["currentQuestion"][1], range(8,10,1))) {
    if (strpos($_POST["answer"], ",") !== false && in_array($answerArray[0],$_SESSION["answer"]) && in_array($answerArray[1
        in_array($answerArray[0],array($answerArray[1],$answerArray[2]))) {
        $correct = true;
    } else {
        $correct = false;
        $_SESSION["answer"] = $_SESSION["answer"][0]."_".$_SESSION["answer"][1]."_".$_SESSION["answer"][2];
    };
};
```

Leaderboard.php



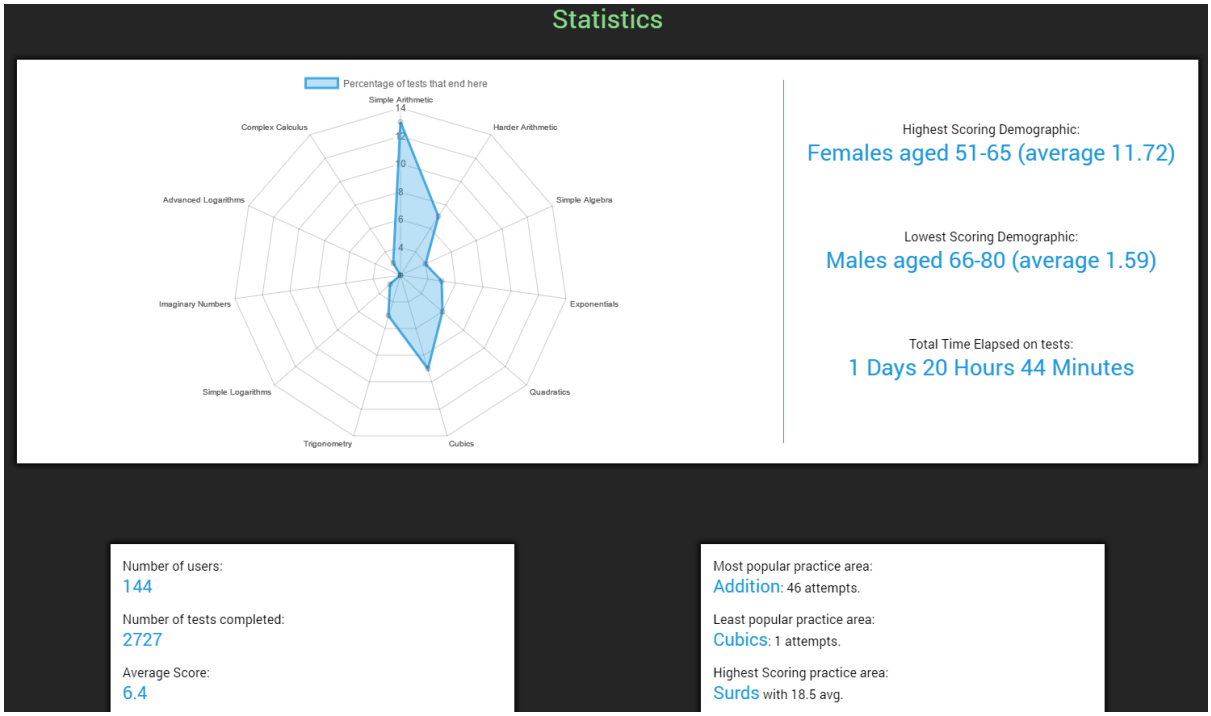
This is the default leaderboard.php, it shows the top 10 tests ever taken on the MMT. The blue <h2> tag is what shows the user what they're searching for, although this can be altered using a combination of a JavaScript slider, drop down menu and a text input. For example, if I wanted to the view the best scores on custom test "0ea44", only males and of age below 30:




```
<?
include "connect.php";
$query = mysqli_stmt_init($db);
if (isset($_POST["testtoken"]) && $_POST["testtoken"] !== "") {
    $token = mysqli_real_escape_string($db,$_POST["testtoken"]);
    mysqli_stmt_prepare($query,"SELECT USER_NAME, FINAL_SCORE, TIME_TAKEN, DATE_RECORDED FROM scores INNER JOIN u
        USER_SEX=? OR USER_SEX=?) AND (TIMESTAMPDIFF(YEAR, users.USER_DOB, CURDATE()) BETWEEN ? AND ?) ORDER BY s
    mysqli_stmt_bind_param($query,"sssss",$token,$sex1,$sex2,$lowerLimit,$upperLimit);
} else {
    mysqli_stmt_prepare($query,"SELECT USER_NAME, FINAL_SCORE, TIME_TAKEN, DATE_RECORDED FROM scores INNER JOIN u
        scores.TEST_TOKEN IS NULL) AND (USER_SEX=? OR USER_SEX=?) AND (TIMESTAMPDIFF(YEAR, users.USER_DOB, CURDAT
    mysqli_stmt_bind_param($query,"ssss",$sex1,$sex2,$lowerLimit,$upperLimit);
};
mysqli_stmt_bind_result($query,$username,$score,$time,$date);
mysqli_stmt_execute($query);
$index = 0;
```

This screenshot shows the long SQL query required to cater for all variables.

statistics.php



This image represents what I was aiming for in my data analysis. It shows a variety of different data structures including a spider graph, averages and demographic statistics as well as insightful analytics. All of this is achieved through various SQL queries that are cached by the server to prevent approximately 20 sql queries being made every time this page is loaded.

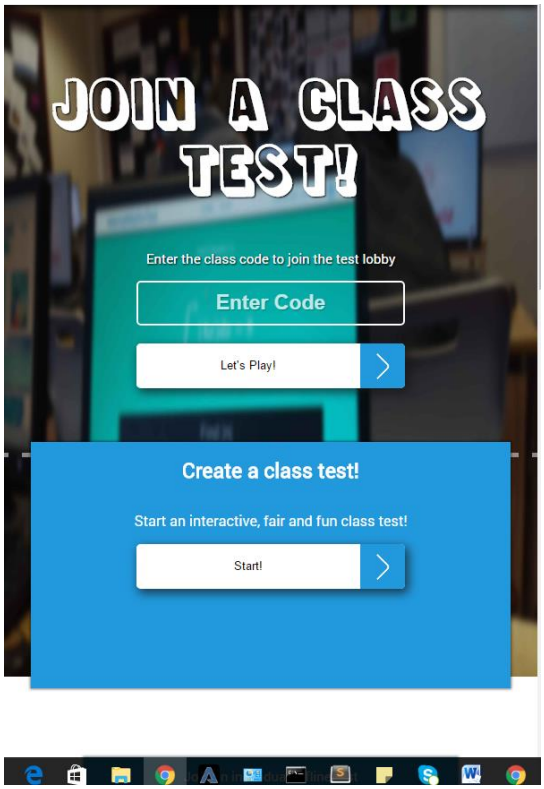
The spider graph is fed data from the server via SQL and the database but manipulates it through the a JavaScript plugin. Other than that this page is rather simple CSS wise.

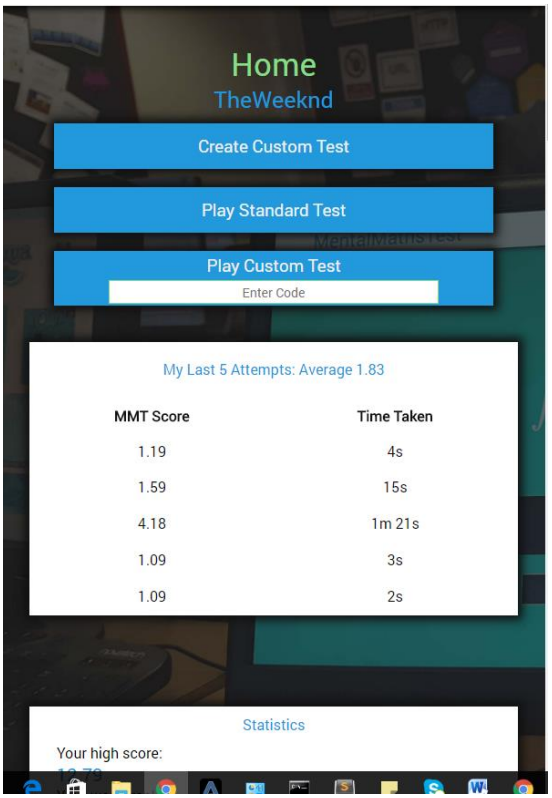
```
$result = mysqli_query($db,"SELECT TEST_TOKEN,COUNT(*) AS FREQ FROM scores WHERE TEST_TOKEN LIKE '
$popPractice = mysqli_fetch_assoc($result);
echo "<p class='pStat2'>Most popular practice area:<br><span class='practiceStat'>".explode("pract
$result = mysqli_query($db,"SELECT TEST_TOKEN,COUNT(*) AS FREQ FROM scores WHERE TEST_TOKEN LIKE '
$leastPopPractice = mysqli_fetch_assoc($result);
echo "<p class='pStat2'>Least popular practice area:<br><span class='practiceStat'>".explode("prac
$result = mysqli_query($db,"SELECT TEST_TOKEN,AVG(FINAL_SCORE) AS AVERAGE FROM scores WHERE TEST_T
$highScoringPractice = mysqli_fetch_assoc($result);
echo "<p class='pStat2'>Highest Scoring practice area:<br><span class='practiceStat'>".explode("pr
$result = mysqli_query($db,"SELECT TEST_TOKEN,AVG(FINAL_SCORE) AS AVERAGE FROM scores WHERE TEST_T
$lowScoringPracticeArea = mysqli_fetch_assoc($result);
echo "<p class='pStat2'>Lowest Scoring practice area:<br><span class='practiceStat'>".explode("pra
```

Above is a screenshot of a snippet of the code that gathers the miscellaneous average data from the database.

Media Queries

Not a file but a noteworthy feature of my design is that all the pages available to users are fully functional across a multitude of devices due to CSS3's media queries functionality. Here are a few examples:






```
1257  @media screen and (max-width: 1150px) {
1258      #YourScoreHeader, #ResponsiveHeader {
1259          font-size: 15px;
1260          margin-top: 10px;
1261          margin-bottom: 10px;
1262      }
1263      #LevelHeader {
1264          font-size: 15px !important;
1265          margin-bottom: 10px;
1266      }
1267      #MentalAgilityIcon, #MentalArithmeticIcon, #MentalPowerIcon {
1268          width: 100px !important;
1269          height: 100px !important;
1270      }
1271      #ImprovementsList h4 {
1272          font-size: 20px !important;
1273      }
1274      .levelBreakdown {
1275          font-size: 10px;
1276      }
1277      .lowerBody {
1278          height: 945px;
1279      }
```

An example of media queries to change the layout and flow of a document depending on certain constraint such as the width of the window, in this case.

The Login Algorithm:

I thought it would be convenient to add the login algorithm to the document, as an example of one tiny part in the whole network of files. For convenience, I have added this in text form instead of screenshots for readability and since the algorithm is just too large for screenshots, despite it being relatively small (**I have also removed all indentation for readability, since most lines are too long for the page. For the indented version, look to login.php in the code print-out**):

```
<?php
session_start();
include "connect.php"; // Connects server to local database
if (isset($_POST["loginemail"]) && isset($_POST["loginpassword"]) && isset($_POST["loginbutton"]) && $_POST["loginbutton"] == "Log
In") {
if (strlen($_POST["loginemail"]) < 64 && strlen($_POST["loginemail"]) > 0 && strlen($_POST["loginpassword"]) > 0 &&
strlen($_POST["loginpassword"]) < 64) { // More validity checks
if (filter_var($_POST["loginemail"], FILTER_VALIDATE_EMAIL)) { // Check valid email address
$query = mysqli_stmt_init($db); // Initialise prepared statement to database
mysqli_stmt_prepare($query,"SELECT USER_ID,USER_NAME,USER_HASH FROM users WHERE USER_EMAIL=?"); // prepare statement
mysqli_stmt_bind_param($query,"s",$escapedemail); // Bind statement paramaters to variables
$escapedemail = mysqli_real_escape_string($db,$_POST["loginemail"]); // Sanitize email address of SQL-Malicious characters
mysqli_stmt_execute($query); // Execute Query
mysqli_stmt_bind_result($query,$id,$username,$hash); // Prepare to assign result to variables
mysqli_stmt_fetch($query); // Get results & assign
if ($username === null) { // Invalid email address
header("Location: https://www.mentalmathstest.com?s=false"); // Redirect to homepage with invalid login flag
};
mysqli_stmt_close($query); // Close SQL Statement object
} else {
$query = mysqli_stmt_init($db);
mysqli_stmt_prepare($query,"SELECT USER_ID,USER_HASH FROM users WHERE USER_NAME=?");
mysqli_stmt_bind_param($query,"s",$escapedusername);
$escapedusername = mysqli_real_escape_string($db,$_POST["loginemail"]);
mysqli_stmt_execute($query);
mysqli_stmt_bind_result($query,$id,$hash);
mysqli_stmt_fetch($query);
mysqli_stmt_close($query);
};
if (password_verify($_POST["loginpassword"],$hash)) { // If hashed version of user input + their concatenated salt is identical to their hash
then password correct
// Successful Login
$_SESSION["loggedIn"] = true; // Assign session variables
$_SESSION["userID"] = $id;
$_SESSION["guest"] = "false";
ob_start(); // Begin output buffering
if (isset($_POST["loginrememberme"]) && $_POST["loginrememberme"] === "RememberMe") {
include "rememberme.php"; // Include rememberme cookie code
} else {
setcookie("rememberMe","",time()-3600,null,null,null,true);
};
echo "<script>window.location = 'https://www.mentalmathstest.com/home';</script>"; // Redirect to home.php
ob_end_flush(); // End output buffering
} else {
header("Location: https://www.mentalmathstest.com?s=false"); // Password incorrect
};
} else {
header("Location: https://www.mentalmathstest.com?s=false"); // Invalid login details
};
};
?>
```

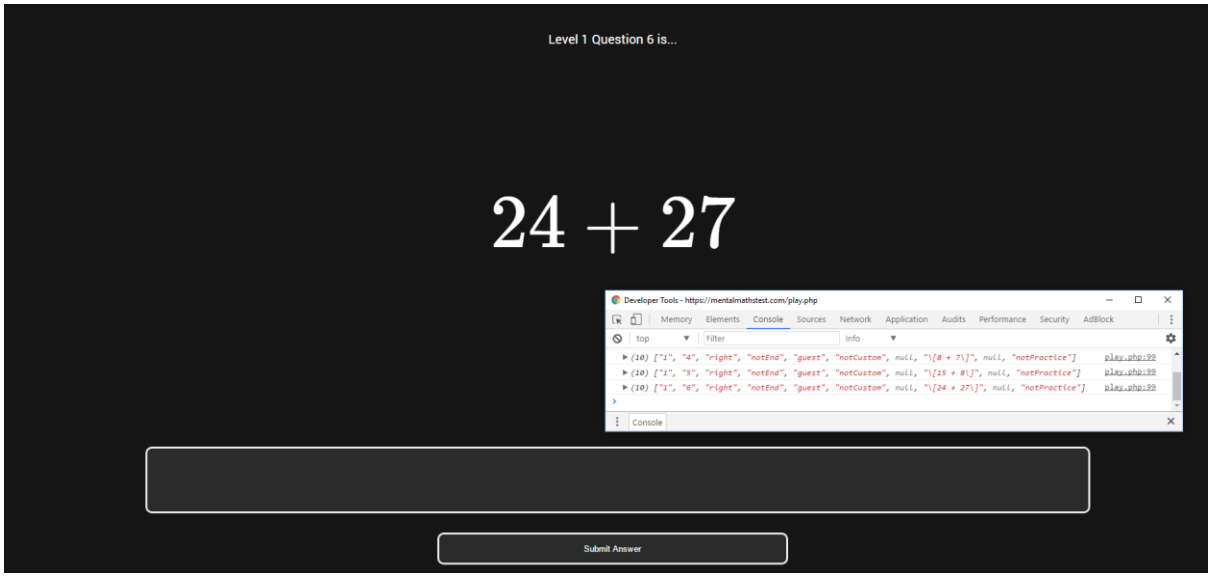
Evaluation

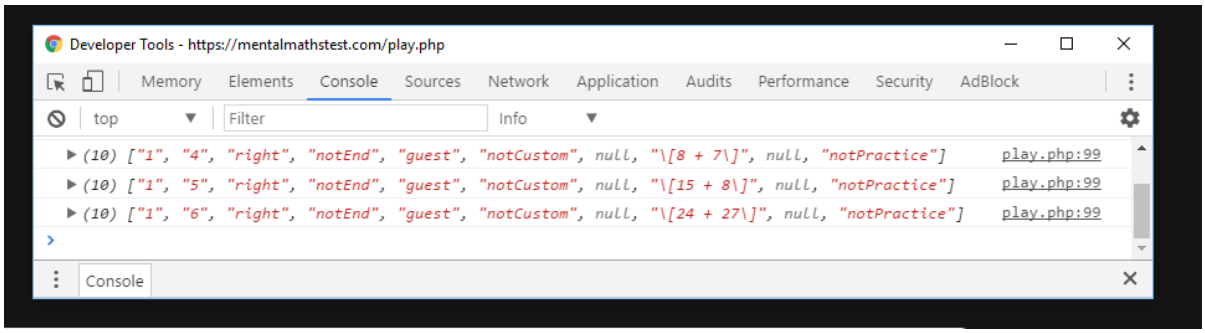
Test Table

Test No.	Description	Data Type	Expected Result	Pass/Fail	Evidence
1	Play a normal-mode test as a guest (no attempt to log in)	Typical	The guest should be able to play a normal-mode test as if a logged in user, but will receive a different dialogue box upon finishing/answering wrong.	Pass	Screenshot 1
2	Create a custom test as a guest	Typical	The guest should be able to create a test and then told what the generated code is	Pass	Screenshot 2
3	Play a custom test as a guest	Typical	The guest should be able to play any custom test.	Pass	Screenshot 3
4	Sign Up	Typical & Erroneous	The guest user should be able to sign up using commonly accepted data values. Also, incorrect values should not allow a user to sign up and should prompt the user to try again.	Fail	Screenshot 4
5	Log In	Typical & Erroneous	The signed up user should be able to log in given their inputs are correct. Otherwise, the user will be prompted that the login details are in fact false.	Pass	Screenshot 5
6	Create a test as a user	Typical	The logged in user should be able to create a custom test and for it to be saved, then to be displayed on their home.php.	Pass	Screenshot 6
7	Practice Logarithms as a user	Typical	The logged in user should be able to access all of the practice modes and thus be able to initiate them.	Pass	Screenshot 7
8	Play a custom test as a user	Typical	Any user, along with a guest can play any custom test as long as they have the generated code. Thus, I expect users to be able to play custom tests.	Pass	Screenshot 8
9	Get a question wrong.	Erroneous	I expect that once a user gets a question wrong they are redirected appropriately and prompted sufficiently	Pass	Screenshot 9
10	Finish a test	Extreme	The user will be redirected and prompted sufficiently.	Pass	Screenshot 10

Evidence

Screenshot 1

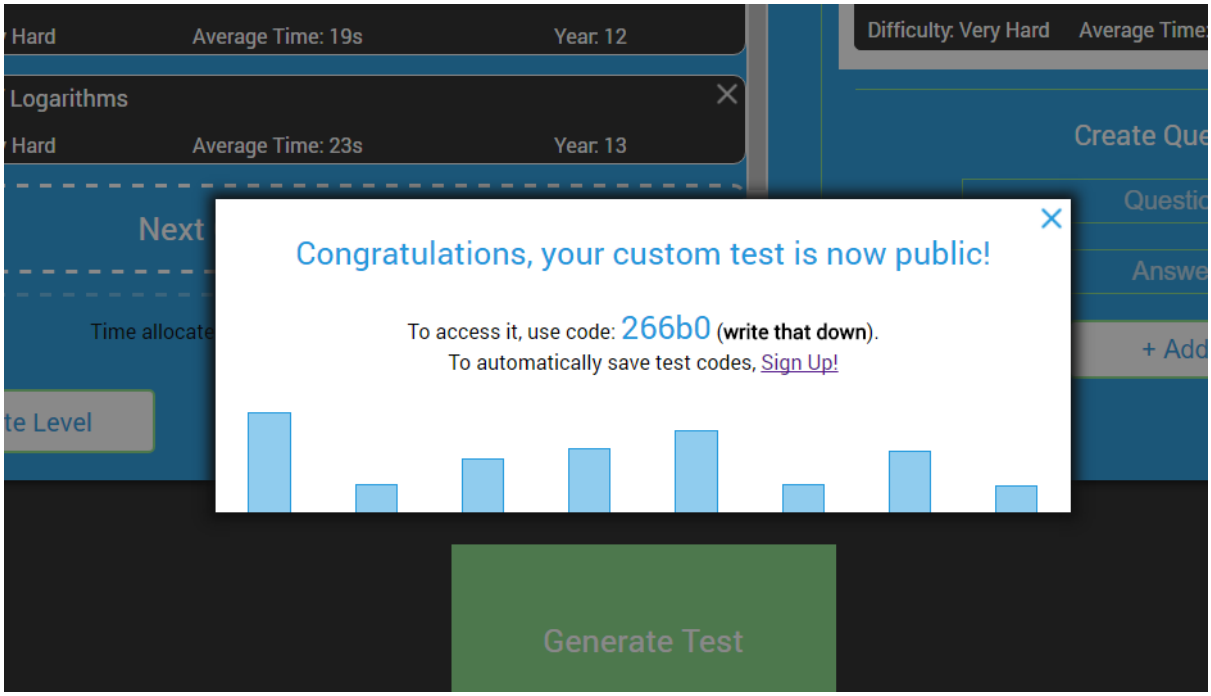




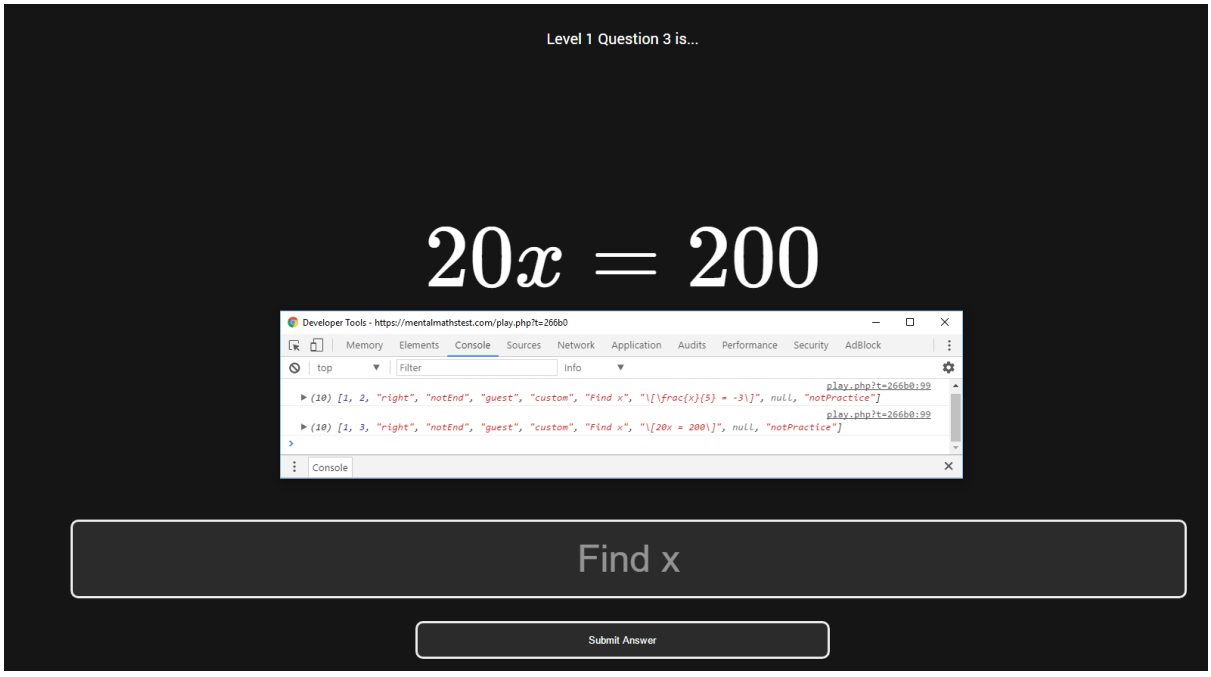
Console log proves that I a guest user during this test.

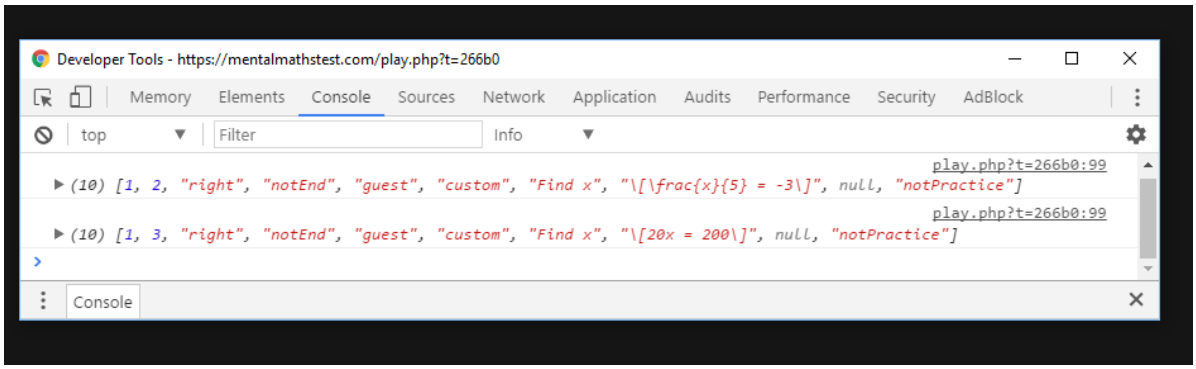
Screenshot 2

As you can see I was able to successfully access createtest.php as a guest user and as demonstrated by the next screenshot, I was able to generate the test and receive the test code.

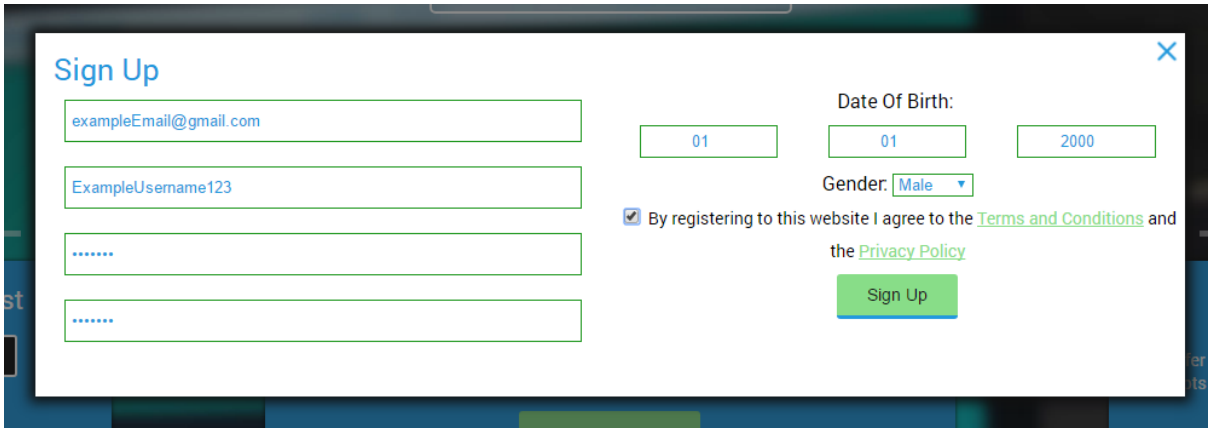


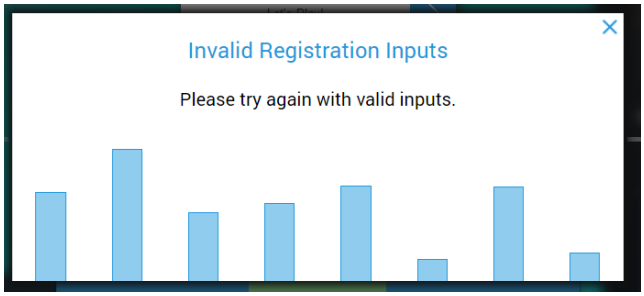
Screenshot 3





Screenshot 4





Having entered this valid information, I was unexpectedly greeted with the dialog box I programmed to be activated upon invalid information having been submitted. I then proceeded to try a few more times to make sure there wasn't an anomaly and the information I entered was valid. There was, it seems, a problem.

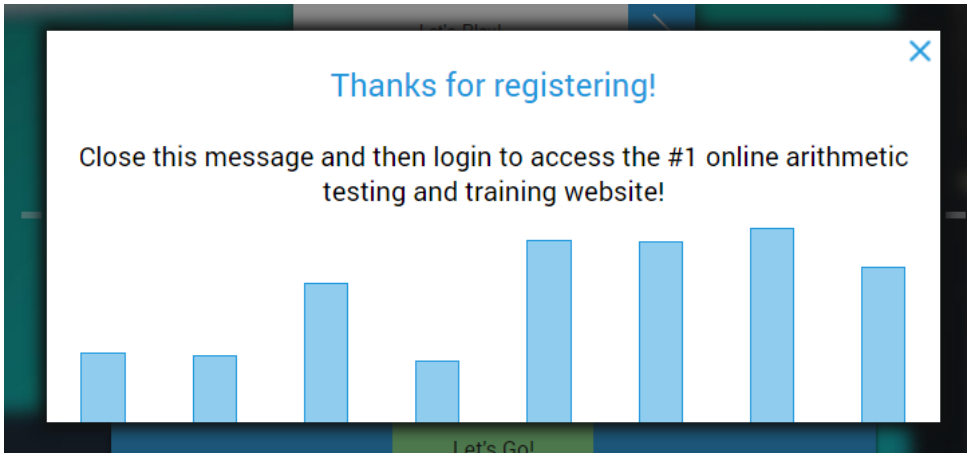
Firstly, I looked to the registration code, register.php. In order for this error message to appear, register.php redirected me to http://mentalmathstest.com?s=false, identifying through URL parameters that there was an issue with registration. In my registration code, there are 7 ways in which the user can be redirected with s = false:

1. Password 1 has invalid number of characters.
2. Invalid number of characters for days in DOB.
3. Invalid number of characters for months in DOB.
4. Invalid number of characters for years in DOB.
5. Gender is invalid.
6. Sign Up button not submit or invalid.
7. Not all fields have been submit.

So, in order to target what was causing the error, I temporarily redirected the user to s = false + errorNumber for each possible fault. For example, if the fault was that password 1 had an invalid number of characters, I would be redirected to s=false1. It turns out that I was redirected to s = false7. I then found that isset($_POST["signupgender"]) was returning false. Thus, the server was under the impression that the gender information had not been set.

```
<form id="UserAccountSignUpForm" action="register.php" m
    <div class="signUp1">
        <input type="email" placeholder="Email" id="Sign
        <input type="text" placeholder="Username" id="Si
        <input type="password" placeholder="Password" id
        <input type="password" placeholder="Re-enter pas
    </div>
    <div class="signUp2">
        Date Of Birth: <br><input type="number" placehol
        <input type="number" placeholder="MM" id="SignUp
        <input type="number" placeholder="YYYY" id="Sign
        Gender: <select form="SignUpForm" name="signupge
            <option value="test">------</option>
            <option value="Male">Male</option>
            <option value="Female">Female</option>
        </select><br>
```

Notice that the form id is "UserAccountSignUpForm" , and also notice that the form attribute for the <select> tag is "SignUpForm". Thus, since the client thinks that this select tag is for a different form, the data is not sent when the form is submit. Changing the form attribute on the select tag quickly fixed this:

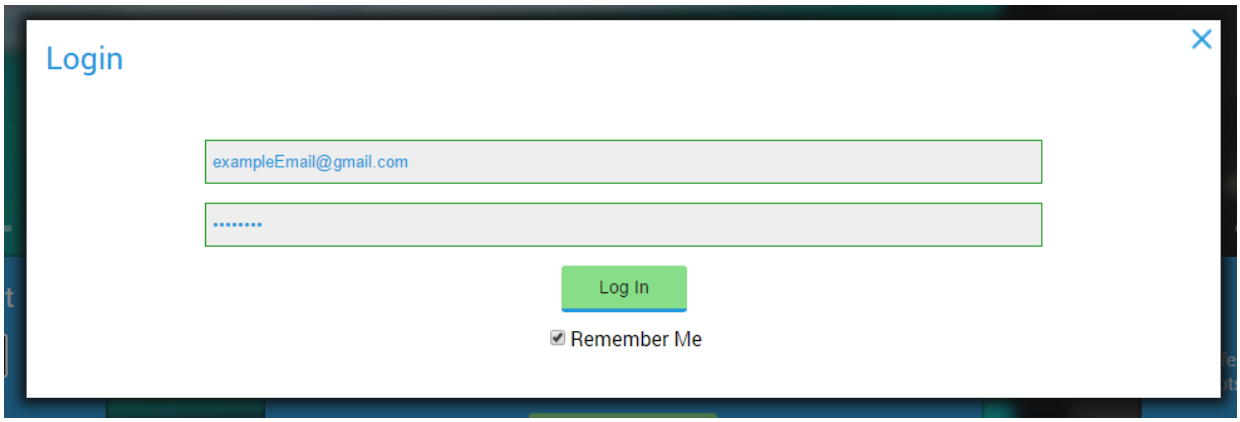


This is the dialog box that lets users know they have successfully signed up to the website.

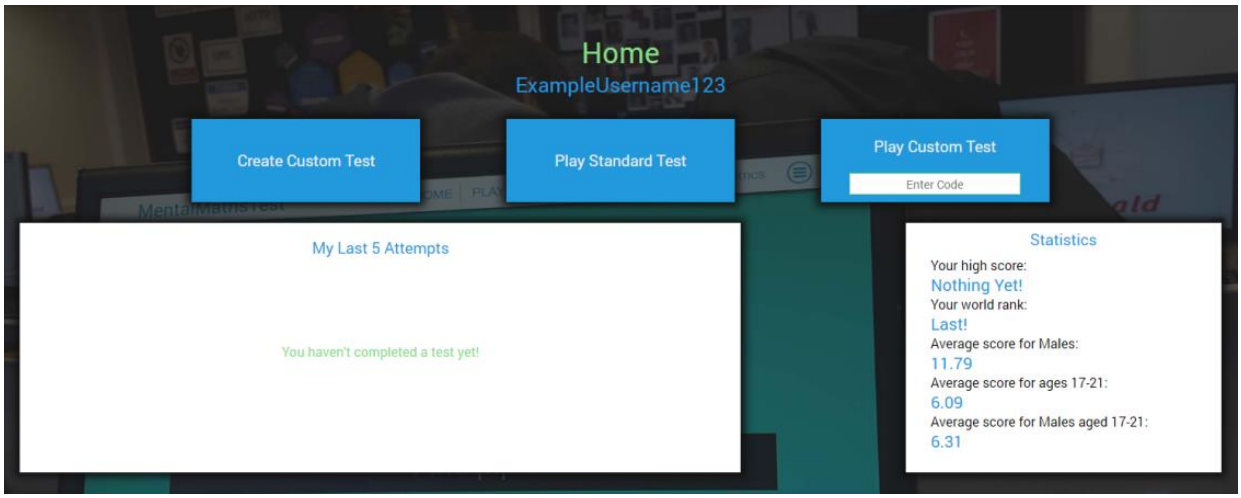
USER_ID	USER_EMAIL	USER_NAME
121	exampleEmail@gmail.com	ExampleUsername123

To the left is evidence that the user has been successfully added to the database (users table).

Screenshot 5

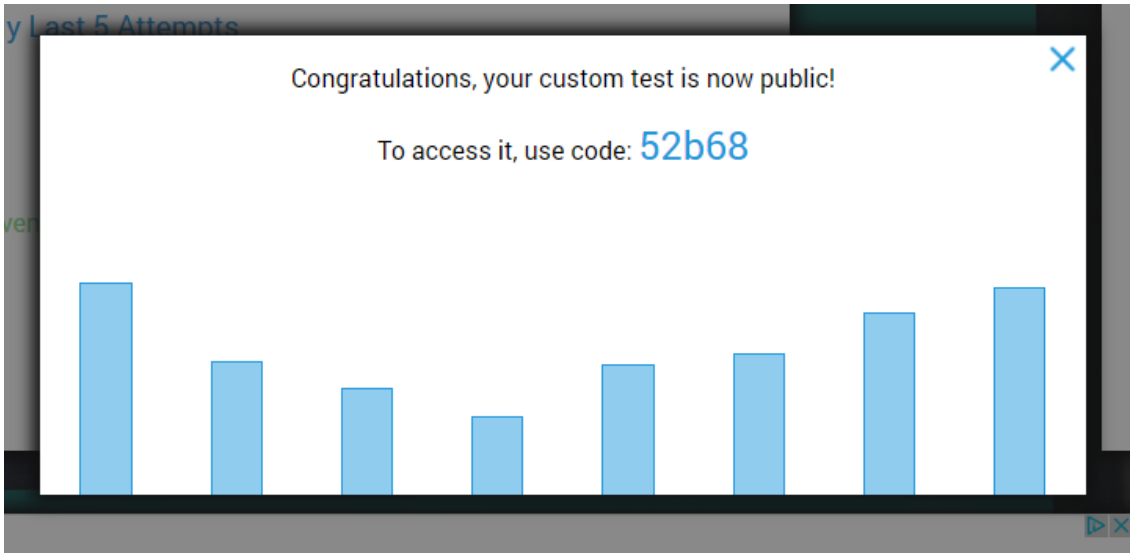


Above is the login screen, I am using the account I just created for this test and will be using it for all the remaining tests.

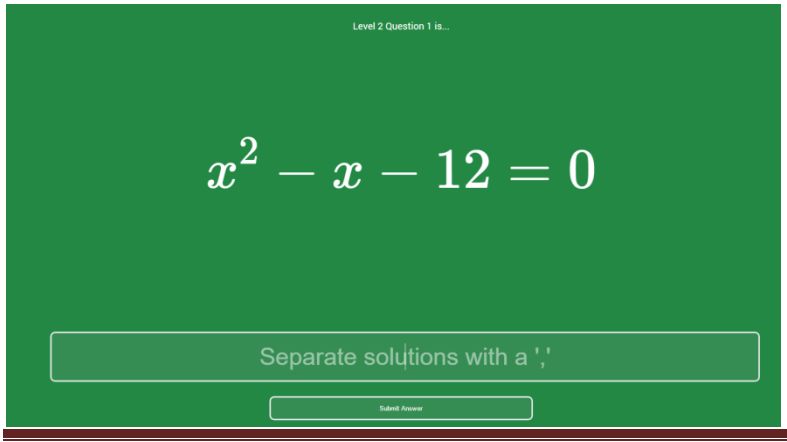


As you can see, the user has been successfully redirected to home.php and everything is working perfectly. The username is being output below the <h1>, exception cases are being output for when the user has no logged tests too.

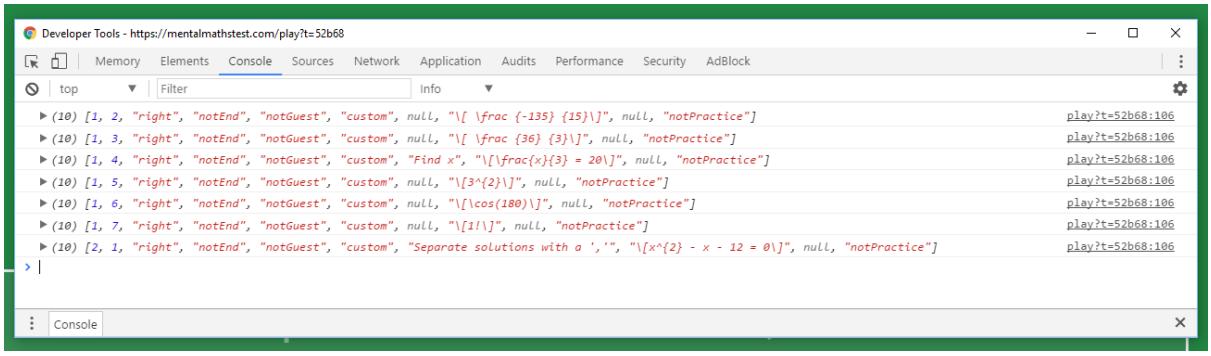
Screenshot 6



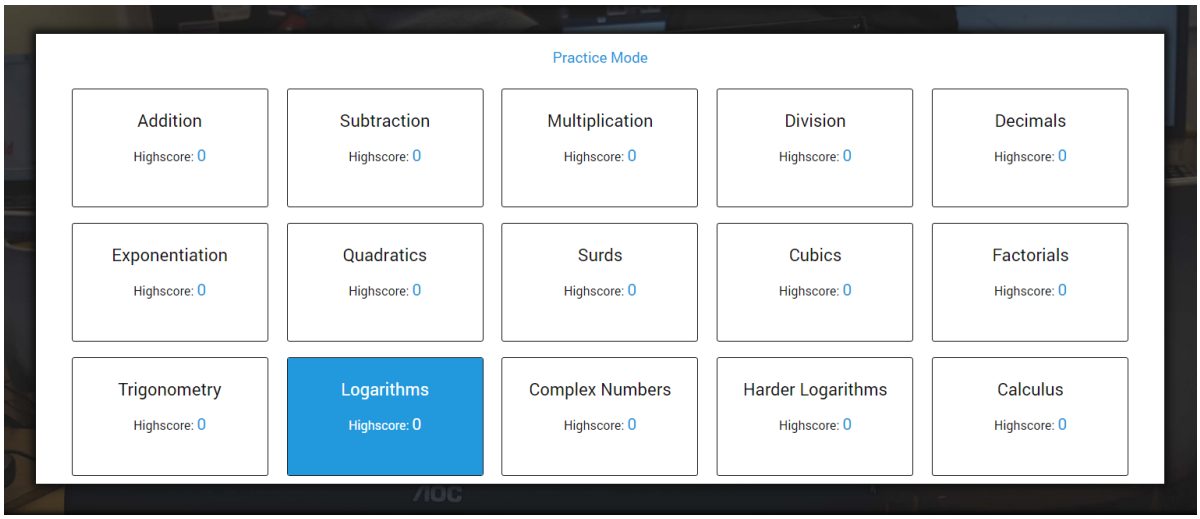
Above is proof the test has successfully been created, Below is evidence that the test can be successfully played and accessed by other users.



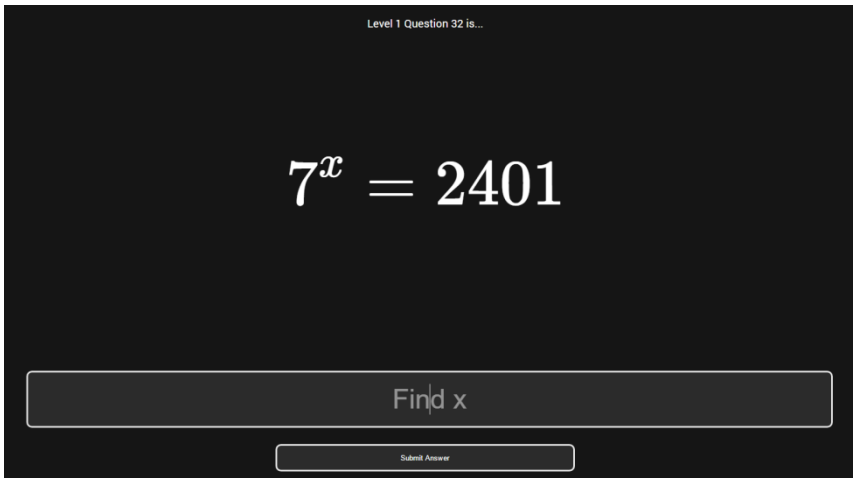
It is evident in this picture a custom test must be being displayed, since the AI is displaying a quadratic question on level 2 question 1, which would never normally happen. Also, below is the console log proving it is a custom test.



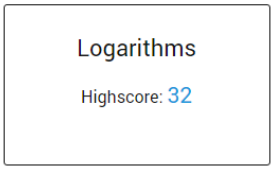
Screenshot 7



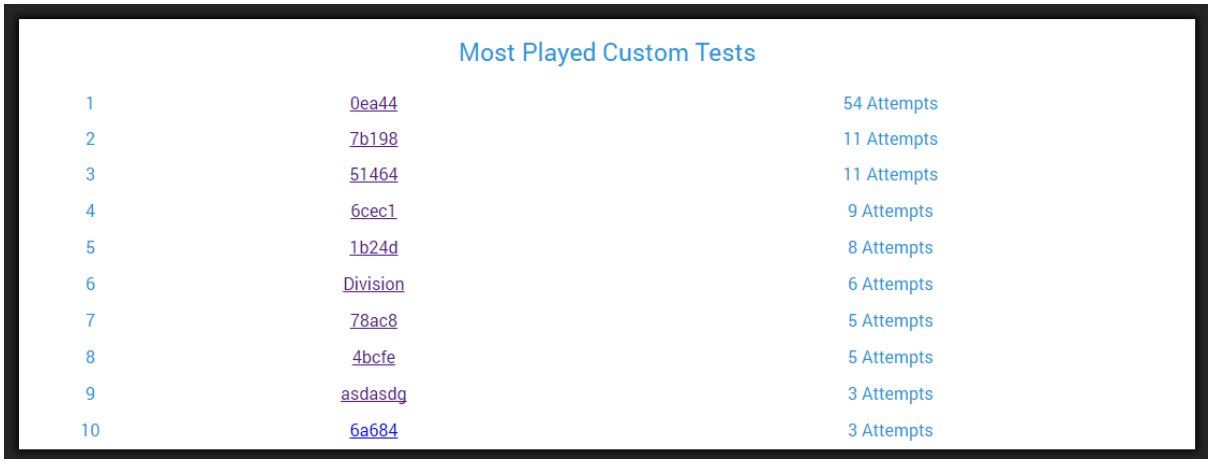
On the practice section, I'm just about to click logarithms. Notice that my high score is displayed as 0. This is true since I have not completed a logarithms practice test yet.



As you can see, if this were a normal or custom mode test, I wouldn't be able to reach question 32 and if it was normal or custom mode, there still wouldn't be any logarithm questions this complicated. Once completed, it is evident that my high score for practice-mode logarithms has been updated:



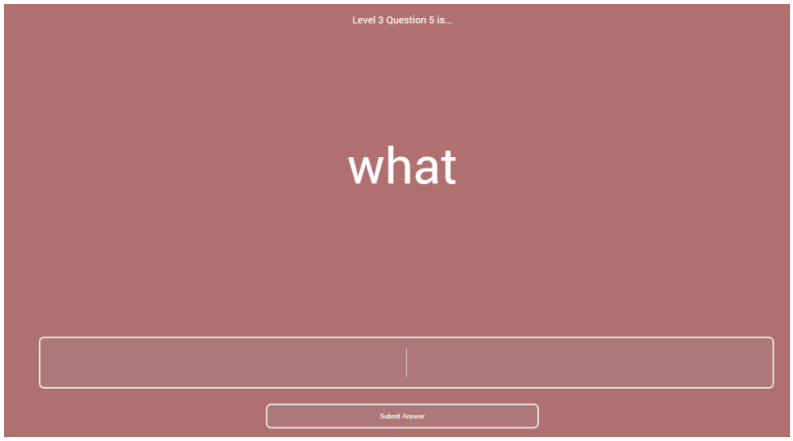
Screenshot 8



Most Played Custom Tests

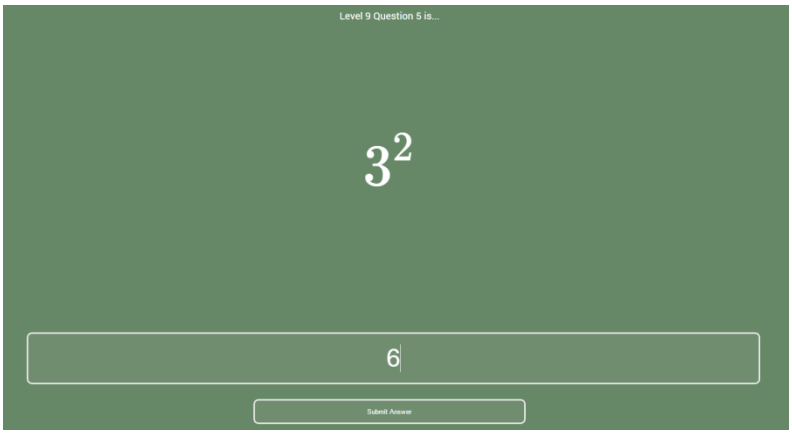
1	0ea44	54 Attempts
2	7b198	11 Attempts
3	51464	11 Attempts
4	6cec1	9 Attempts
5	1b24d	8 Attempts
6	Division	6 Attempts
7	78ac8	5 Attempts
8	4bcfe	5 Attempts
9	asdasdg	3 Attempts
10	6a684	3 Attempts

According to the statistics page, “0ea44” is the most popular custom test, so I will attempt to play that test as a logged in user.



As demonstrated on the left, is me playing a custom test which has a custom question.

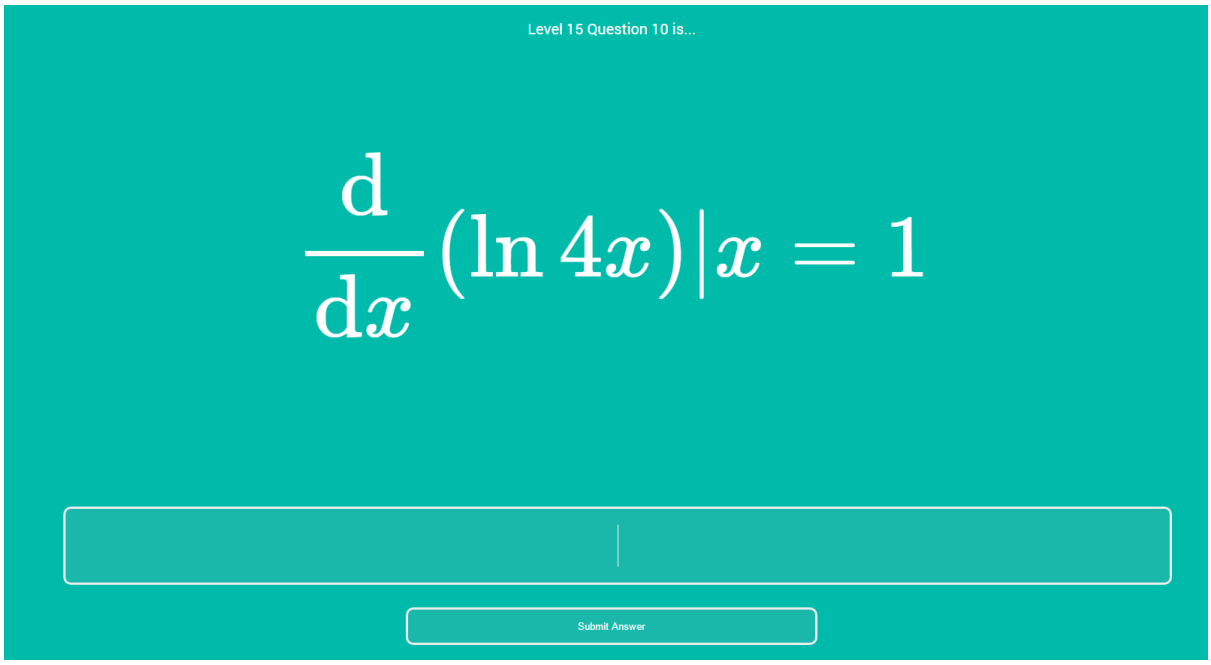
Screenshot 9



As is evident, I am playing a normal-mode test and the AI has generated the question 3 squared, I will enter 6 because I intend to be wrong, since the answer is 9. As you can see in the screenshot on the following page, the server redirected me to home.php where I was told that I had aswered the question incorrectly.

The prompt contained insightful information, such as what the user entered, their score and the correct answer to the question.

Screenshot 10



This was the generated last question by the AI, having answered correctly I was redirected back to home.php and shown this prompt containing my score:

Conclusion

I fully believe that my website ticks all of the objectives outlined in analysis, although some of these are down to opinion. For example, whether or not the website is aesthetically pleasing is up to opinion. However, I find that it is aesthetically sufficient. I believe that while developing the website I went far and beyond some objectives, also creating functionalities that weren't outlined initially such as the statistics page, with all of its insightful and interesting figures.

On the other hand, I feel like with most things, there is room for improvement. I believe that a functionality that allows users to directly connect with other users and assess each other's scores, custom tests, etc. would only be beneficial and would encourage interactivity. To build on this, certain users over the age of 18 could be allowed to link their account to a Facebook or Twitter account so that the website can access various benefits from their API's such as additional demographics statistics and an increased social outreach which would contribute to a healthy SEO.

Here is a table of all of the outlined objectives followed by their completion status according to me and my third party (Mr. Charles):

Objective	**Status (According to me)**	**Status (according to Mr.Charles)**
Allow people who access the website to make an account and thus be given special permissions.	Fully Complete	Fully Complete
Securely store user's data such as passwords and personal information.	Fully Complete	Fully Complete
Be up to date legally (Privacy Policy, Terms and Conditions).	**Mostly Complete**	**Mostly Complete**
Give users the option to securely store a minimalistic cookie on their machine so that their account can be remembered and automatically logged in.	Fully Complete	Fully Complete
Make sure every page is aesthetically pleasing.	Fully Complete	Fully Complete
Run advertisements on some pages to gain an income.	Fully Complete	Fully Complete
Develop an Artificial Intelligence system that determines the nature of the next question given the state of the user's answer to the last, and the numbers to be used pseudo-randomly.	Fully Complete	Fully Complete
Allow people who aren't registered to play a trial test.	Fully Complete	Fully Complete
Derive a scoring system that accurately represents the user's performance on any test.	Fully Complete	Fully Complete
Include up to A2-Level further mathematics in the main test.	Fully Complete	Fully Complete

Allow users to create their own tests.	Fully Complete	Fully Complete
Allow these custom tests to be played by other users.	Fully Complete	**Partially Complete**
Allow users to make their own worded questions in the custom tests.	**Mostly Complete**	**Mostly Complete**
Have a "Leaderboard" page to track the scores submitted on a given test.	Fully Complete	Fully Complete
Ensure the website is fully functional on 99% of devices (responsive).	Fully Complete	Fully Complete
Have healthy SEO.	Fully Complete	Fully Complete
Register the website under a suitable domain.	Fully Complete	Fully Complete
Have a low page load time across the domain.	Fully Complete	Fully Complete
Establish a mode whereby the questions go on forever until a wrong answer is given and difficulty increases constantly.	Fully Complete	Fully Complete

Having received Mr. Charles' verdict on the completeness of my project, I questioned him on some of his choices.

Both Mr. Charles and I agreed that the legal documentation was mostly complete, since you can never be 100% with terms and conditions, privacy policy etc. We aren't lawyers and hiring one seemed out of the question.

Mr. Charles believed that the creation of custom tests was partially complete, when I asked he told me this was due to the fact that he would like to see online, interactive, multiplayer games, similar to those on "Kahoot!", whereby an entire class can join a session and partake in a test. This, however, is clearly out of reach of A-Level syllabus, but as I outlined earlier on in this document, I am currently developing this functionality outside of coursework on the website. This is because I believe it will take longer than the allocated amount of time to completely develop this feature.

We both agreed that the worded questions feature was mostly complete rather than fully complete because it doesn't allow users to create a worded question with multiple answers. Although this requirement is rare, it isn't non-existent and again, would be very doable given time, but I'd have to rewrite the custom question detection algorithm.

All in all I see this project as a huge success, I have accomplished what Mr. Charles and I had intended to do in providing an online, interactive platform to test, track and improve any user's mental arithmetic ability, while collecting and providing harmless data analysis at the same time.

If you'd like to see this project running or like to test it out yourself, go to http://mentalmathstest.com.

Sources Table

Source	Link	Description
jQuery	https://jquery.com/	JavaScript Plugin that enables increased functionality at a low latency cost.
MathJax	https://www.mathjax.org/	JavaScript Plugin that allows complex mathematics to be rendered among a large portion of browsers efficiently.
Pace	http://github.hubspot.com/pace/docs/welcome/	JavaScript Plugin that adds a customisable loading bar to the top of the page.
Chart	http://www.chartjs.org/	JavaScript Plugin that allows pages to render accurate and lightweight graphs.
Unslider	http://unslider.com/	JavaScript Plugin that acts as a carousel, for any type of element.
nouislider	https://refreshless.com/nouislider/	JavaScript Plugin, used to implement the slider on leaderboard.php. However I did fiddle with the CSS of this plugin to make it suitable.
Roboto	https://fonts.google.com/specimen/Roboto	Font, commercially available and widely used on my website.
Lunch	http://www.dafont.com/lunch.font	Font, only used once as h1 tag on index.php, commercially available.
Sublime Text 3	https://www.sublimetext.com/	IDE used for the entirety of the website.
HostGator	https://www.hostgator.com/	Hosting service used to host my website.