“A Multiple Choice Test” Website

Created with **PHP**, **SQL** / **MySQL** and **HTML** / **CSS**

Project and Documentation by Michael Gebhart

Full Code on GitHub

Presentation Video on Google Drive

Project .zip on Google Drive

Before I started with this project, I had already gathered some web design knowledge in HTML, CSS and JavaScript with the creation of static websites that were, for instance, my own portfolio or my third semester project *Underground Berlin* (2020).

Anyway, getting to know the creation of dynamic websites with PHP, SQL and MySQL was a really great addition to that. Even though I have struggled a lot in the beginning with getting the syntax for PHP right - especially in combination with HTML documents - I am really satisfied with the result I have achieved.

The Goal

The task was to create a website that provides **a multiple choice test** to a user who should be able to **login** and **register** with his name and password first. Furthermore, the user should be able to **save** the progress so that it's possible to return to the test to a later time. The files should be written in HTML, PHP and SQL. To offer decent UX, I have decided to stylise the project with CSS.

Project Structure

The final project consists of four different webpages the user can navigate between:

- `index.php` (login)
- `register.php` (registering as a new user)
- `test.php` (the multiple choice test)
- `result.php` (display of the final score)

All the questions including their selectable answers are located in `test.php`. As of now, there are five questions each with three possible answers. Therefore, it was needed to define a structure to make references to every question and answer (more on p.5):

- each **question** is referenced to with its number of order as a written out `string` (***one, two, three,...***)
- each **answer** of every question has an `int value` in accordance to its order (between **1** and **3**; with **1** for the first, **2** for the second and **3** for the final answer)

For the **MySQL** part, there is one database (called '`test`') that contains three different tables:

- `users` (a table where every user is registered with name and password)
- `save` (save data so that each user can save his/her progress of the test)
- `solution` (a table that defines the right answers of every question)

Users

A new entry is created with every user's registration with **username** and **password** of choice and an unique **ID** (auto-incremented):

id	username	password
1	Michael	123
2	Gordon	abcd

Save

A new entry of identical ID is created with every user's registration. Each column's **key** *(one, two, three...)* addresses the **question** while it's **value** represents the respective **answer** (1-3).

In this example, Michael (id: 1) has already finished the test with four out of five right answers while Gordon (id: 2) has just saved his progress after selecting the first answer (value: 1) of the first two questions (key: one; two) and leaving the other questions unedited (default value: 0).

id	one	two	three	four	five	result
1	3	1	3	2	1	4
2	1	1	0	0	0	0

Solution

This table is structured similarly to ***save*** and consists of one entry that defines **the right answers** (value:1-3) to every question (key: one, *two,...*). This table cannot be edited by the user in any way.

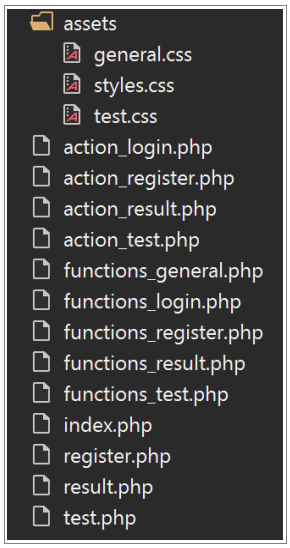
The result in table *save* is calculated by comparing the values to the user's save data once the test is submitted.

id	one	two	three	four	five
1	3	2	3	2	1

Please note: I have been considering to create a second database called 'questionnaire' that contains in one table all questions and answers. Due to time constraints and the task's definition that "*for the content of the test* [I] *can take whatever* [I] *like to*", I have decided not to provide the questions and answers via a database table, but to hard code them with HTML.

If I had gone with the former solution, I would have create one row for each question: In each row, every column would contain either the question, a possible answer or the key to the right answer. In the actual HTML code, this would have been represented by a PHP code block that echoes and iterates through each row.

Web Page Structure



Each page consists of three parts:

1.) **main .php** file combined with HTML code that has a `<form>`
2.) **action_*.php** file which is called by said form or on page load
3.) **functions_*.php** file serving as a function library for the action file

Every functions_.php file is based on a general functions library **functions_general.php** which includes general methods such as checking whether an user or a database exists or getting an user's ID by name.

Conclusively, the files call each other's definitions as in the following:

- index.php → action_login.php → functions_login.php → functions_general.php
- register.php → action_register.php → functions_register.php → functions_general.php
- test.php → action_test.php → functions_test.php → functions_general.php
- result.php → action_result.php → functions_result.php → functions_general.php

The Web Pages

Index.php

index.php · action_login.php · functions_login.php · functions_general.php

The very first page the user gets to see is **index.php** with the **log-in screen**.
If the user enters his **username** and **password** into the respective fields and clicks the submit button `[LOG-IN]`, the action form would trigger the `return bool` function **`check_login()`**. That function searches the *user* table and selects all rows with an identical username and password, i.e. it is checked whether the user exists at all:

```
$stmt_checklogin = $sqli->prepare("SELECT * FROM users WHERE users.username = ? AND users.password = ?");
$stmt_checklogin->bind_param("ss", $username, $password); $stmt_checklogin->execute();
```

The number of rows from that statement is then calculated with `'$stmt_checklogin->num_rows'` and if that number is not 0, but 1, the function would return `true`, redirect the now logged-in user to the multiple choice test on **test.php** and save the username as a reference in **`$_SESSION['username']`**.

Otherwise with `false`, an error message is echoed and displayed (as seen in the screenshot) and the user is allowed to try it again with different input data.

A hyperlink at the end of page leads the user to **register.php** where he can create a new entry in the *user* table.

Register.php

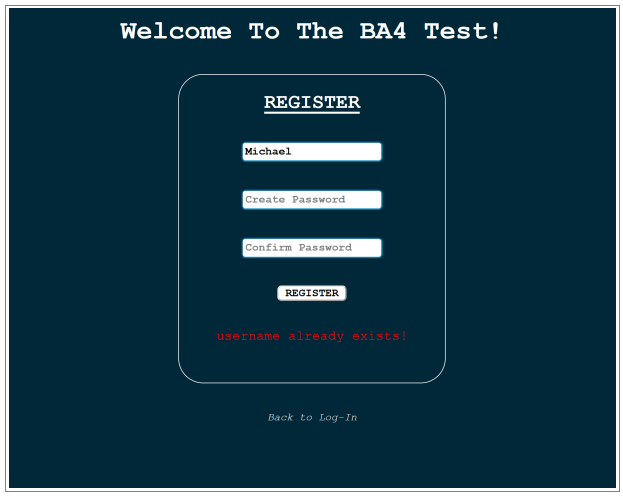
register.php · action_register.php · functions_register.php · functions_general.php

On **register.php** the user can create new entries in the *user* table.

For this, the user who is yet unregistered needs to enter: 1.) an **unique username** and 2.) the **password twice** into the dedicated fields and to click the submit button `[REGISTER]` to trigger the action form which then calls the function **`register_user()`**:

```
$stmt_insert = $sqli->prepare("INSERT INTO users (username, password) VALUES (?, ?)");
if (check_user($sqli, "users", $_POST['username']))
        echo "username already exists!";
else {
        if ($_POST['validated_password'] != $_POST['password'])
                echo "passwords do not match!";
        else {
                $stmt_insert->bind_param("ss", $_POST['username'], $_POST['password']);
                $stmt_insert->execute();
        }
}
```

Here, it is checked whether requirements 1.) and 2.) are met. If not, the function would echo an error message:



If the registration process is successful otherwise, a new user entry would get inserted into the **table *user*** with an `auto_increment` ID, a new username and password. Finally, the user is then redirected back to the log-in screen on **index.php** to re-enter his/her login data.

With every new user, there's not only one new entry in the table *user*, but also a new entry in the **table *save*** with the same ID of the user to keep track of the user's state of progress. To set up a basic system for the multiple choice test, each row represents the value of answers (1-3) to each question as the key (*one*, *two*, *three*,..):

```
$stmt_save = $sqli->prepare("INSERT INTO save (one, two, three, four, five, result) VALUES (?, ?, ?, ?, ?, ?)");
```

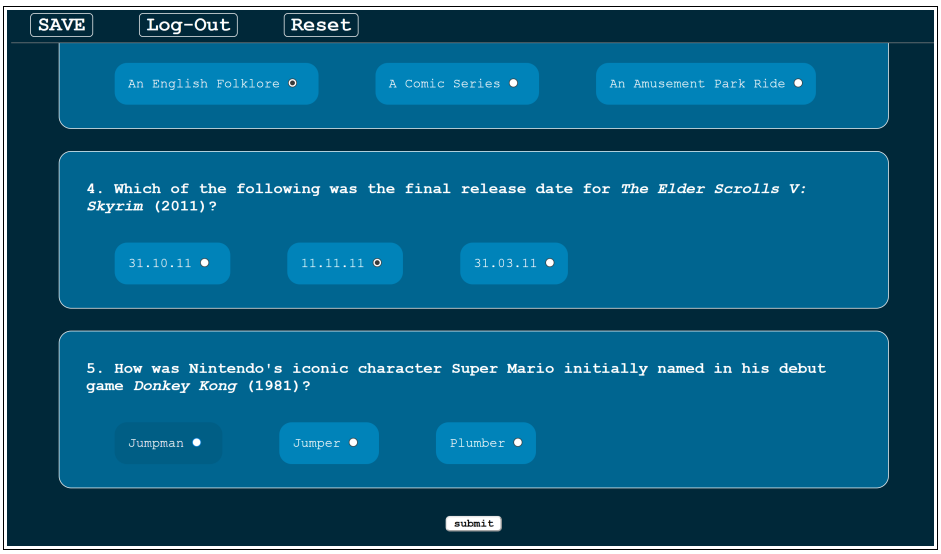
Before executing the prepared statement above, all values `?` are set to the same **`default_value = 0`** which represents that no answer has been chosen yet due to this entry being created before **test.php** has even been opened yet.

Furthermore, the first user to register triggers the function `create_test_database()` that initializes the full database.

Test.php

test.php · action_test.php · functions_test.php · functions_general.php

The multiple choice test on **test.php** consists of one page that can be scrolled through:



Only logged-in users can access this page. With a manually typed URL, for instance, they would get redirected:

```
If (!isset($_SESSION['username'])) Header ('Location: /'); //redirecting back to index.php
```

One code block for a question is written in HTML /PHP as in the following (example of question 4.):

```
<div class="multiple-choice">
        <h3>4. Which of the following was the final release date for <i>The Elder Scrolls V: Skyrim</i> (2011)?</h3>
        <label>31.10.11
                <input type="radio" class="choice-radio" id="4-1" name="four" value="1"
                                                        <?php if(get_save_int('four') == 1) echo "checked"; ?>>
        </label>
        <label>11.11.11
                <input type="radio" class="choice-radio" id="4-2" name="four" value="2"
                                                        <?php if(get_save_int('four') == 2) echo "checked"; ?>>
        </label>
        <label>31.03.11
                <input type="radio" class="choice-radio" id="4-3" name="four" value="3"
                                                        <?php if(get_save_int('four') == 3) echo "checked"; ?>>
        </label>
</div>
```

To get the latest saved progress of the test, the function **`get_save_int([string question_key])`** is repeatedly called, returning from **table *save*** the **answer value** for the question which is referenced to by the given string parameter.
If that returned answer value is the same as the **`value`** in **`<input>`**, the corresponding answer would be be marked as **`checked`**. If that returned answer value is **`0`**, no answer is selected at all.

Furthermore, the web page includes **four buttons** each with different action functions:

- **Save**: updates the **table *save*** with the values of the answers (e.g. '`name="four" value="3"`') the user has selected. With ternary operations, it is checked whether a question is answered and what values should be updated with.
- **Log-Out**: redirects the user back to **index.php** and destroys the `$_SESSION` including all of its variables.
- **Reset**: resets all values in table *save* to **`default_value = 0`**.
- **Submit**: submission of the test which calls the function **`calculate_result()`** that calculates the final score by comparing the answer values of the tables *save* and *solution* with each other; updates the table *save* with not only the current answer values, but also the final score for the key *result*; and redirects the user to **result.php**.

Result.php

result.php · action_result.php · functions_result.php· functions_general.php

The final score is displayed on **result.php**:

HTML calls the final score with `<?php echo get_result() ?>` from the table *save*:

```
$stmt_getresult = $sqli->prepare("SELECT result FROM save WHERE id = ?");
$stmt_getresult->bind_param("i", $_SESSION['user_id']); $stmt_getresult->execute();
$row_result = $stmt_getresult->get_result(); $value = $row_result->fetch_object();
return $value->result;
```

Finally, the user is able to either retry – which resets the *save* progress and redirects to test.php – or to log out and leave the `$_SESSION` by clicking on the button.

Personal Recap

As I have already stated in the beginning, getting to know the creation of dynamic websites including PHP and SQL was a really nice experience. In the end, I was able to finalize the project in a matter of one week. It was also great to try out the concept of ternary operations, a concept some friends of mine introduced me into.

All in all, I am really satisfied with this task and I am excited for future possibilities that might let me build on said experiences in terms of dynamic web design.

I can recommend the following tutorials and sources:

- John Morris via Skillshare (2017)
- Tutorialwork via YouTube (2019) [German]
- php.net [Retrieved: 2020]
- w3schools.com [Retrieved: 2020]

Thank You!