

**WEB SERVER APPLICATIONS DEVELOPMENT 1**

**420-DW3-AS**

**FINAL PROJECT**

Statement - Winter 2023

**OBJECTIVE**

Create a simple PHP Web app, including a registration/login form and a kid's game.

**CONDITIONS**

* Team of **3 students**.
* Create a GitHub **repository** in the beginning of the project and regularly update it to include all the files of your Web app at the end.
* Weighting: **30%** of the final grade.

**TABLE OF CONTENTS**

* Components (Registration, Login and Modification forms, Game forms, components, Database, History page)
* Additional technical specifications
* Evaluation grid

**1-COMPONENTS**

* 1. **Registration, Login and Modification forms**

Use the super global variable POST or GET to send the form data.

**A-Login form**

Create a login form (Sign-In), including:

* 2 input fields named “Username” and “Password”
* 2 submit buttons named “Connect” and “Sign-Up” that allow players to submit the form or to access to the registration form.

This login form is the front page (home page) of the Web app.

How the login form works?

* After a successful authentication (correct username and password, as saved in the database) a session must start and the browser must be redirected to the first level game.
* When the username or the password is wrong, a significant error message is shown, such as “*Sorry, you entered a wrong username!*”, as well as a link that invite to modify the password, such as “*Forgotten? Please*, *change your password.*”.

**B-Registration form**

Create a registration form (Sign-Up), including:

* 3 input fields named “Username”, “Password”, and “Confirm Password”
* 2 input fields named “First name” and “Last name”
* 2 submit buttons named “Create” and “Sign-In” that allows players to submit the form or to access to the login form.

Players access this registration form when they clicked the button “Sign-Up” in the front page.

How the registration form works?

* When the username written within the registration form is similar to a username already stored in the database, a significant error message is shown, such as: “*Sorry, this username* already exists. *Please, choose another one.*”
* When the 2 passwords written (“Password”, and “Confirm Password”) and within the registration form are not the same, a significant error message is shown, such as: “*Sorry, you entered 2 different passwords.*”
* When the first name or last name does not correspond to correct information (i.e. empty, or start with a number) a significant error message is shown, such as: “*Sorry, your first name cannot be empty” or “Sorry, your last name cannot start with a digit or number.”*

**C-Password modification form**

Create a password modification form, including:

* 3 input fields named “Existing Username”, “New Password”, and “Confirm New Password”
* 2 submit buttons named “Modify” and “Sign-In” that allows players to submit the form or to access to the login form.

Players access this modification form when they enter a wrong username or/and password and clicked the hyperlinks “Forgot Password” that appears.

How the password modification form works?

* + Based on the previous indications for the login and registration forms, decide how this form will work, but in a way it makes sense.
  1. **Game forms**

**A-Game forms, buttons, and result**

Create a form for each level of the game indicated below that allows players to play with numbers from 0 to 100 and alphabet letters from a to z, lower case or/and uppercase.

When the user submits the form of a game,

1. If it’s not the last level, a result message is shown, including a button “Sign Out” and “Stop this Session” (incomplete playing session) and:
   1. If the user wins the game, a button “Go the Next Level” is shown
   2. If the user wins the game a button “Try Again this Level”, is shown
2. If it’s the last level, a result message is shown (congratulations for winners or encouragement for game over), including a button “Play Again”, “Home Page” and “Sign Out”
3. If:
   1. All the numbers or letters entered are different (not similar) than the ones displayed, an appropriate message is shown.

For example.

Our numbers: 9, 11, 54, 62, 36, 41

Instructions: order these numbers in ascending order

Your numbers: 10,12, 55, 63, 64, 65

Result: Incorrect – All your numbers are different than ours

* 1. Some of the numbers or letters entered are different (not similar) than the ones displayed, an appropriate message is shown.

For example.

Our numbers: 9, 11, 54, 62, 36, 41

Instructions: order these numbers in ascending order

Your numbers: 9,11, 41, 54, 62, 65

Result: Incorrect – Some of your numbers are different than ours

* 1. All the numbers or letters entered are the same displayed but their order is incorrect, an appropriate message is shown.

For example.

Our numbers: 9, 11, 54, 62, 36, 41

Instructions: order these numbers in ascending order

Your numbers: 9,11, 41, 54, 62, 36

Result: Incorrect – Your numbers have not been correctly ordered in ascending order

* 1. All the numbers or letters entered are the same displayed and their order is correct, an appropriate message is shown.

For example.

Our numbers: 9, 11, 54, 62, 36, 41

Instructions: order these numbers in ascending order

Your numbers: 9,11, 36, 41, 54, 62

Result: Correct – Your numbers have been correctly ordered in ascending order

**B-Lives and Game Over**

Each session allows 6 lives to a user, and when these lives are used and the user doesn’t win, it’s game over. To win, a user must win the 6 levels of the game, that are described below:

**C-Game Levels**

**Game Level 1: Order letters in ascending order**

A set of 6 different letters generated randomly is shown and the user must use the form available to write them in ascending order (from a to z).

**Game Level 2: Order letters in descending order**

A set of 6 different letters generated randomly is shown and the user must use the form available to write them in descending order (from z to a).

**Game Level 3: Order numbers in ascending order**

A set of 6 different numbers generated randomly is shown and the user must use the form available to write them in ascending order (from 0 to 100).

**Game Level 4: Order numbers in descending order**

A set of 6 different numbers generated randomly is shown and the user must use the form available to write them in descending order (from 100 to 0).

**Game Level 5: identify first and last letters from a set of letters**

A set of 6 different letters generated randomly is shown and the user must use the form available to write the first letter and the last letter (from the order a to z).

**Game Level 6: identify the minimum and the maximum numbers from a set of numbers**

A set of 6 different letters generated randomly is shown and the user must use the form available to write the minimum number and the maximum number (from the order 0 to 100).

**1.3. Database**

Create a database and components, using the extension mysqli or pdo, to collect and manipulate the following data:

1. The registration profile information of the user, that are:
   1. The first name
   2. The last name
   3. The id (generated automatically)
2. The current authentication information of the user, that are:
   1. The username
   2. The password
3. The result information of the user, that are for each session,
4. The date/time
5. The result (win, fail, or incomplete)
6. The number of lives used

Create the database, tables and columns automatically with the script the first time the app runs.

**1.4. History page**

Retrieves from the data base and displays in a table the information indicated below for each player

1. id
2. first name
3. last name
4. result
5. number of lives
6. and date/time

**2-ADDITIONAL TECHNICAL SPECIFICATIONS**

**2.1. HTML Structure**

A-You are free to decide the html structure of the Web pages, but your pages must include at least a <header>, <nav>, <article>, and <footer>.

B-The navigation menu must include the following elements: Login, Registration, and History that consecutively allow to access:

1. The login page
2. The registration page
3. The history page

C-In addition to the forms and messages, in terms of content, you must choose and display at least a name for the game in the <header> and the name of the developers in the <footer>.

D-There are a lot of additional information available in this statement that you can display to improve the user experience; for example, you can display what each level of the game includes and the number of lives allowed for each session.

**2.2. PHP Structure**

1. Build your code using an appropriate structure, including only user-defined functions and/or OOP (class, property, method, and object).
2. When there’s an error in the information written in a form (validation failed), display both a significant error message and the form with the information written to allow the player to easily modify it and submit the form again, when applicable.
3. Do not create isolated pages, without buttons or hyperlinks (as indicated) that allows the player to go to another page.

**2.3. Code and Directory Structure**

1. Use an extension for Visual Studio Code to be able to automatically format (adding for example indents, lower case letters, and upper case letters for constants) the code of your program to make it easy to be reviewed.
2. Add significant comments to the code of your program to make it easy to be understood and maintained.
3. Use standard file names, such as index.php and style.css.
4. Use standard directory names to group files, such as css, image, js, and font
5. Include a readme.txt file in the root of your program that indicates:
6. The full names of the developers (team members) and the contribution of each to the program built.
7. Any interesting indications about the program built, for example the version of php used, how it works, and how to use it.
8. Each single class can be saved in a file and multiple related functions can be written in the same file… Create multiple files when required instead of written a lot of non-related codes in a single file.

**2.4. Validation and Sanitization**

1. Validate the information entered by the player in the registration form in a way it makes sense. This means the first name, last name, username, and password of the player.
2. Validate the information entered by the player in the game form, in a way it makes sense. For example, numbers cannot be entered in a form where alphabet letters are required.
3. Validate the information retrieved from the database.
4. Sanitize the information entered by the player and retrieved from the database, using built-in functions, such as stripslashes(), strip\_tags(), and htmlentities().
5. Protect the passwords, using built-in functions, such as password\_hash() et password\_verify().

**2.5. Style**

You are free to add style (e.g. CSS and Bootstrap) to customize the visual aspect of the pages and components (e.g. forms, text, images…). It will be appreciated. However, stay focus more on PHP syntax, statements and arrays, functions, OOP (classes, properties, methods, object) and connection of a script to a MySQL Database.

**3-EVALUATION GRID**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **#** | **Elements and tasks evaluated** | **Correct or Highly complete** | **Moderately complete** | **Incomplete** | **Incorrect or Absent** | **Total** |
| 1 | Login form + DB | 10 | From 9 to 5 | | 0 | 10 |
| 2 | Registration form + DB | 10 | From 9 to 5 | | 0 | 10 |
| 3 | Password modification form + DB | 10 | From 9 to 5 | | 0 | 10 |
| 4 | Game Level 1 | 5 | From 4 to 2.5 | | 0 | 5 |
| 5 | Game Level 2 | 5 | From 4 to 2.5 | | 0 | 5 |
| 6 | Game Level 3 | 5 | From 4 to 2.5 | | 0 | 5 |
| 7 | Game Level 4 | 5 | From 4 to 2.5 | | 0 | 5 |
| 8 | Game Level 5 | 5 | From 4 to 2.5 | | 0 | 5 |
| 9 | Game Level 6 | 5 | From 4 to 2.5 | | 0 | 5 |
| 10 | Database - Game - History page | 10 | From 9 to 5 | | 0 | 10 |
| 11 | PHP Structure (functions, OOP) | 10 | From 9 to 5 | | 0 | 10 |
| 12 | PHP Validation and Sanitization | 3 |  |  | 0 | 3 |
| 13 | Code and Directory Structure | 3 |  |  | 0 | 3 |
| 14 | HTML Structure | 3 |  |  | 0 | 3 |
| 15 | Style and other (i.e.; js) | 1 |  |  | 0 | 1 |
| 16 | Pear review | 5 | From 4 to 2.5 | | 0 | 5 |
| 17 | Oral presentation | 5 |  |  | 0 | 5 |
| **TOTAL** | | | | | | **100** |