WEB SERVER APPLICATION DEVELOPMENT 1

420-DW3-AS

FINAL PROJECT

Statement – Winter 2024

Patrick Saint-Louis

SECTION 1

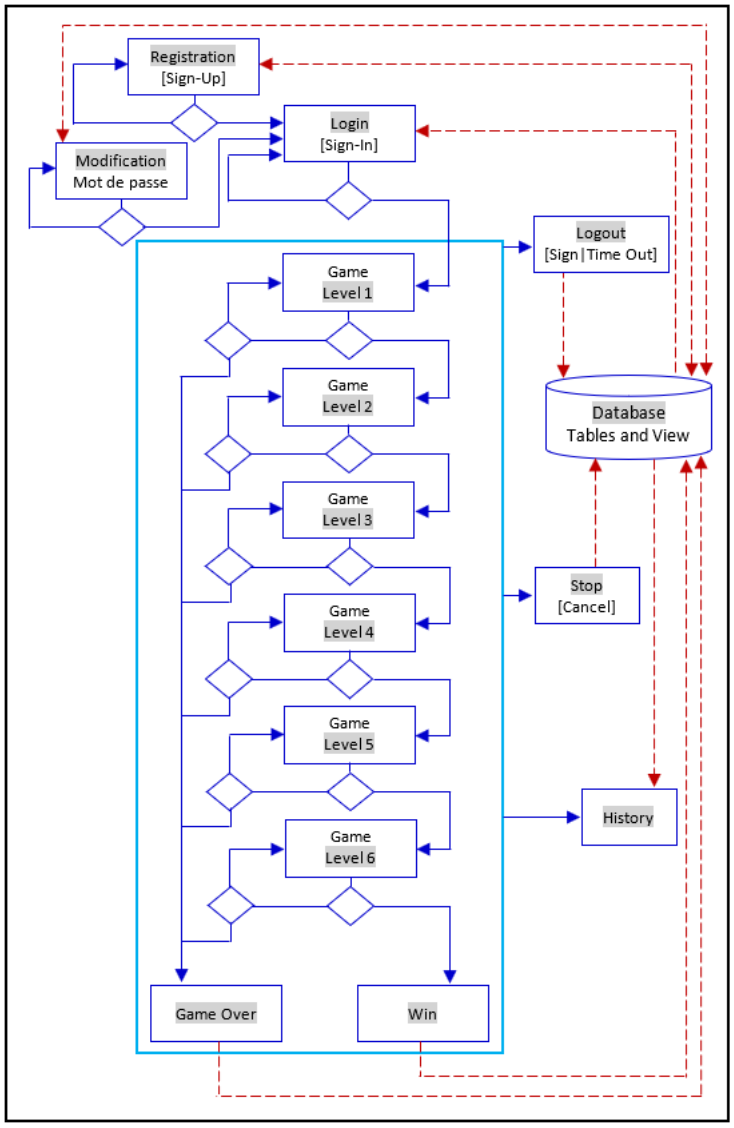
WORK TO BE DONE AND PROCESS OF REALIZATION

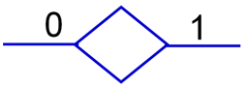
1.1. Work to be done

Create a PHP children's games website, which includes the following features:

1. Creation of user accounts or registration (Sign-Up).
2. Real-time validation of the information entered by the user in the form for creating user accounts with AJAX.
3. Sign-In to an existing user account.
4. Logout of a connected user account (Sign-Out and Time-Out).
5. Changing the password for an existing user account.
6. Management of several levels of a question-and-answer game that follow one another.
7. Abandonment of a game in progress.
8. Display of the history of the results of all the games ended.
9. Manipulation of a MySQL database to save, select, and update data collected and generated by the previous features.
10. Custom display of features and interaction of additional interactivity and attraction features.
11. Integration of any other optional functionality to complete, customize or optimize the website with PHP.

The block diagram below shows the relationship between the features listed above.





2 different states

Ex.

1: Successful feature execution

0: Failed feature execution

Communication in one direction only (sending)

Communication in 2 directions (sending and receiving)

Figure 1-Functional diagram of website features

1.2. Project delivery process

1. Create a team of developers, each with specific tasks to perform, indicated in a work plan developed and shared at the beginning of the project.
2. Create a GitHub repository at the beginning of the project, accessible by all teammates, to store and regularly update website codes and features.
3. Create the directory and file structure diagram that identifies all directories and files contained in the program and their relationships, as well as the features contained in each of these files.
4. Create the technical solution plan that identifies all user-defined functions and/or classes (and other object-oriented programming entities, e.g. subclasses) contained in the program and the features to be performed by each of them.
5. Collaborate to create program unit modules, perform unit functional tests, integrate unit modules, perform integration functional tests, and perform validation functional tests.
6. Submit a first version of the website, including all features, for functional testing and evaluation by another team and in parallel, perform functional tests and other evaluations of a similar website developed by another team.
7. Modify the website as required, based on recommendations provided by evaluators and knowledge gained from the evaluation of another team's website.
8. Submit the latest version of the website for evaluation.
9. Present orally the latest version of the website submitted.

SECTION 2

DETAILS ABOUT FEATURES TO CREATE

2.1. Creation of user accounts or registration (Sign-Up)

2.1.1. Input fields of the registration form

Create a user account creation or registration form that includes:

- 5 input fields named for example "Username", "Password", "Confirm Password", "First Name" and "Last Name".
- 2 submission buttons named for example "Register" and "Login" that allow users to submit this form or access the login form.

2.1.2. Validation of the information entered in the registration form

Perform the following validations **with the server using PHP** (more secure) but do not perform them with the client using just HTML and the web browser (less secure).

- First Name, Last Name, Username, Password and Confirm Password must contain information; Therefore, none of these input fields can be empty.
- First Name, Last Name, and Username must begin with a letter of the alphabet, from a-z or A-Z.
- Username and Password must contain at least 8 characters.
- Password and Confirm Password must be the same.
- The Username cannot be the same as a Username from another account that has already been created in the past, as stored currently in the database. That is, the username is unique.

2.1.3. Processing of information submitted via the registration form

- After the successful creation of an account, (all information entered is correct) the browser is redirected to the home page, while displaying an appropriate message to inform the user.

- When at least one of the information entered by the user is incorrect, the account creation form is displayed again with:
    a. All previously entered information for First Name, Last Name, and Username;
    b. An appropriate error message, for each field, when applicable, such as "*Sorry, this username already exists!"* or *"Sorry, you entered 2 different passwords"*

2.2. Real-time validation of the information entered by the user in the form for creating user accounts with AJAX.

Use AJAX, this means the `XMLHttpRequest()` Javascript class and PHP superglobal variable $_POST, $_GET or $REQUEST, to validate in real time the information entered by the user in the following input fields: First Name, Last Name, Username, Password and Confirm Password.

This means, monitor for example the JavaScript event "onkeyup" (each time a keyboard key is released), in order to validate in the background, the data already written in each input field in order to display error messages when this data is incorrect.

For example, display a message similar with "Must start with a letter a-z or A-Z" when the user starts writing in the field First Name, but typed a digit as the first character.

2.3. Sign-In

2.3.1. Input fields of the login form

Create a user account login form, which is the home page (homepage) of the website and includes:

- 2 input fields named for example "Username" and "Password"
- 2 submission buttons named for example "Login" and "Register" that allow users to submit this form or access the registration form.

2.3.2. Processing of information submitted via the login form

- After successful authentication (correct username and password, as saved in the database), a PHP session (session_start()) starts and the browser is redirected to the top-level game (level 1).
- When the username or password is incorrect, the login form is displayed again with:
    a. The username as previously entered by the user;
    b. An appropriate error message, such as "*Sorry, the username or password is incorrect!* »;
    c. A link that prompts the user to change the password, such as " *Forgot your password ? Change it.* ».

2.4. Disconnection to a logged in user account (Sign-Out and Time-Out)

2.4.1. Sign-Out

After a successful connection, a logout (sign-out) link is displayed in the navigation menu that allows the user to log out and thus close the PHP session open to the connection (session_destroy()).

2.4.2. Time-Out

After a period of inactivity of 15 minutes in a connected user account, an automatic logout (time-out) of said account is performed and thus closes the PHP session opened (session_destroy()).

Upon logout or timeout, the browser is redirected to the home page of the website.

A disconnection results in an incomplete game session being saved in the database.

2.5. Changing the password of an existing user account

Changing the password is only possible before logging in, for example when the user forgets his password.

2.5.1. Password change form input fields

Create a password change form, which includes:

- 3 input fields named for example "Username", "First Name", "Last Name", "New Password" and "Confirm Password"
- 2 submission buttons named for example "Edit" and "Login" that allow users to submit this form or access the login form.

2.5.2. Processing of information submitted via the password change form

- Based on the instructions given above for the forms for creating an account and logging into an existing account, decide how this form works in a logical way that makes sense.

2.6. Management of several levels of a question-and-answer game that follow one another

2.6.1. Game levels and description of their content

Create a form for each level of the game listed below, including the appropriate number of input fields (6 or 2) to allow users to type only (not select or drag because they don’t require the same level of difficulty and learning for the player) numbers and letters displayed as indicated.

The numbers displayed each time:

- Vary from 0 to 100.
- Are 6 in terms of quantity.
- Are different (the same digit is generated only once).
- Are randomly generated.

The letters of the alphabet displayed each time:

- Vary from a to z (lowercase) or A to Z (uppercase), not both lowercase and uppercase at the same time.
- Are 6 in terms of quantity.
- Are different (the same digit is generated only once).
- Are randomly generated.

The game content for each of the game levels is described below. The game always starts with the first level (level 1) and to move to a higher level (e.g. from level 1 to 2, from level 2 to 3…, from level 5 to 6), a player must obligatory succeed the previous level first.

Game Level 1: Order 6 letters in ascending order
Game Level 2: Order 6 letters in descending order
Game Level 3: Order 6 numbers in ascending order
Game Level 4: Order 6 numbers in descending order
Game Level 5: Identify the first (smallest) and last letter (largest) in a set of 6 letters
Game Level 6: Identify the smallest and the largest number in a set of 6 numbers

2.6.2. Possible outcomes of a game that ended

Each game allocates 6 lives to a player. The outcome of a game is determined and recorded in the database when the game ends. A game ends in any of the circumstances described below.

1. Game Over ─ When the quantity of lives given has been exhausted (used) without the player having won the last level of the game, the game ends and the result is a "*game over"*. This means, the user has failed the entire game.

2. Incomplete (abandoned game or user account disconnected: cancel, time-out or sign-out) ─ When the player decides to abandon a game in progress (cancel), is automatically disconnected after a certain amount of inactivity (time-out), or voluntarily chooses to log out during a game in progress(sign-out), the game ends and the result is "*incomplete".*

3. Win (successfully completed game) ─ When the quantity of lives given has not been exhausted (used) but the player has won the last level of the game, the game ends and the result is "*win*".

2.6.3. Processing of information submitted via the game form

When the user submits the form of a game,

1. If it is not the last level of play (level 6), nor the last part (5 lives already exhausted):
    a. An appropriate message is displayed indicating the result;
    b. If the user wins the game, a "Go to next level" button is displayed or the form for the next game level is displayed directly;
    c. If the user fails the game, a "Try again" button is displayed or the form of the same game level is displayed directly.

2. If it is the last level of play (level 6) or the last game (5 lives already exhausted):
    a. An appropriate message is displayed indicating the result (congratulations for the winners and encouragement for the losers);
    b. A "Play again" button appears.

3. Calculation of the result when submitting a game form. If:
    a. All numbers or letters entered are different (not identical) from those displayed, an appropriate message is displayed.

Like what.
Our numbers: 9, 11, 54, 62, 36, 41
Instructions: Arrange these numbers in ascending order.
Your numbers: 10,12, 55, 63, 64, 65
Result: Incorrect – All your numbers are different from ours.

b. Some of the numbers or letters entered are different (not similar) than the ones displayed, an appropriate message is displayed.
Like what.
Our numbers: 9, 11, 54, 62, 36, 41
Instructions: Arrange these numbers in ascending order.
Your numbers: 9,11, 41, 54, 62, 65
Result: Incorrect – Some of your numbers are different from ours.

c. All numbers or letters entered are the same displayed, but their order is incorrect, an appropriate message is displayed.
Like what.
Our numbers: 9, 11, 54, 62, 36, 41
Instructions: Arrange these numbers in ascending order.
Your numbers: 9,11, 41, 54, 62, 36
Result: Incorrect – Your numbers were not correctly arranged in ascending order.

d. All digits or letters entered are the same displayed and their order is correct, an appropriate message is displayed.
Like what.
Our numbers: 9, 11, 54, 62, 36, 41
Instructions: Arrange these numbers in ascending order.
Your numbers: 9,11, 36, 41, 54, 62
Result: Correct – Your numbers have been correctly ordered in ascending order.

2.7. Abandonment of a game in progress

A “Cancel" button always appears in the game form to allow the user to abandon the game to be able to start a new fresh game if needed.

An abandoned game leads to the recording of an incomplete game in the database.

2.8. Display of the history of the result of all game games

This feature is not available until a successful login (authentication) to a user account. For example, it is not accessible on the registration page or on the login page.

This functionality is accessed through a navigation menu item displayed to any user logged into their user account and displays an HTML table (<table> </table>) that contains one row for each game ended ─ i.e. either win, game over, or incomplete ─ that includes the information (columns) shown below.

1. id
2. First name
3. Last name
4. Outcome of the game
5. Number of lives used
6. Date and time (when the game ended)

The history page contains information about all players who have played at least one game, but not only information about the current logged in user account (logging in).

The provided database contains a view that already contains all the columns that include the information to be displayed in the history page.

2.9. Manipulating a MySQL database to insert, select, and update collected and generated data

2.9.1. Data to be collected, generated and manipulated

Use the provided SQL code to create in MySQL via the mysqli or pdo extension and a PHP script a database and its components (e.g., tables, views, columns of tables and views), in order to collect and manipulate the following data:

A- User identity information, including:

1. First name
2. Last name
3. ID (automatically generated)

B- The user's current credentials, including:

1. Username
2. Password

C- Information about the user's results for each game ended:

1. Date/time when the game ended
2. Result (win, game over, or incomplete)
3. The quantity of lives used

2.9.2. Database components and structure

The SQL code to create the structure of the database, including dummy data and queries for testing, is provided in the file named: 5-database-entity.sql.

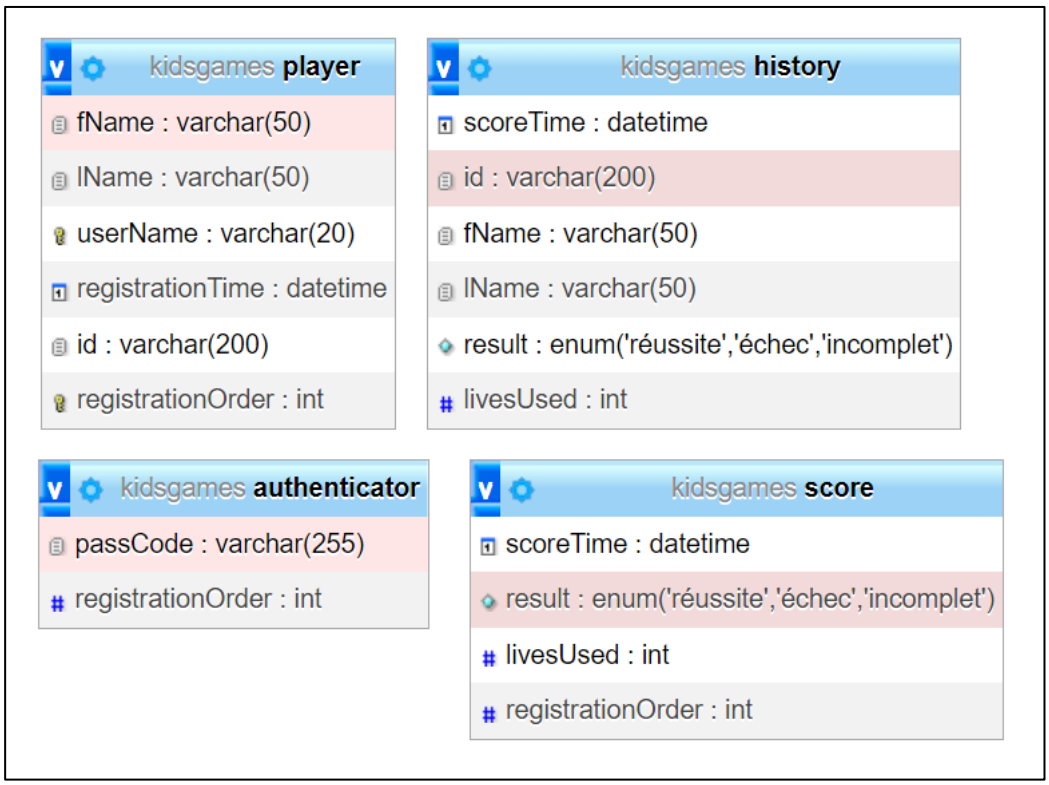


Figure 2-Table, views, columns, and data type of the kidsgame database

2.9.3. Automatic creation of the database and its components and security

- Use the SQL code provided without modification to ensure and facilitate external testers to perform validation functional tests and other assessments of your website.
- Automatically create the database and its components when they are not already created, via a PHP script each time the website is accessed for the first time.
- Use built-in functions, such as stripslashes(), strip_tags(), and htmlentities() to:
    - Sanitize the information entered by the user before storing it in the database and retrieved from the database before displaying it.
- Secure passwords by storing only modified versions in the database, for example, by using built-in functions such as password_hash() and password_verify().

2.10. Custom display of features and integration of additional interactivity and attraction features

In order to use the knowledge already acquired in the client-side web development course:

1. Add image, video, and audio files, if possible, to increase the level of attraction of your web pages.
2. Add a style sheet to the website (for example, CSS or Bootstrap), if possible, to customize the visual appearance of your web pages and their components, including forms, text, and images.
3. Add additional client-side features (e.g. JavaScript and JQuery), if possible, to optimize the interactive look of your web pages and their components.

However, stay 99% focused on using PHP, including basic syntax, conditional and iterative control structures, numbered and associative arrays, date and time, and connecting a script to a MySQL database.

2.11. Integration of any other optional functionality to complete, customize or optimize the website with PHP

The integration of any other optional functionality to complete, customize or optimize the website with PHP will be appreciated. For example, the use of cookies, reading and writing external file management and real-time date and time management.

However, no optional features added will replace any of the features indicated in the previous sections which themselves are mandatory and noted.

SECTION 3

ADDITIONAL TECHNICAL SPECIFICATIONS

3.1. General structure

3.1.1. Text content

There is a lot of additional information available in this document that you can display in your web pages to improve the user experience. For example, you can tell the user on the registration page about the limitations of the user name, which cannot, for example, start with a number.

Also, you can display a brief description of each game level and the total number of lives granted for each game in the homepage (login) or create an additional FAQ page to do so.

3.1.2. Navigation

Do not create isolated pages, that is, pages that do not contain at least one hyperlink that allows the user to navigate to another page, without manually changing the URL in the address bar of the web browser.

3.2. HTML structure

You are free to decide the complete HTML structure of web pages, but all pages displayed in the web browser must include at least <head>, <header>, <nav>, and <footer>.

- <head> must contain all essential tags, including <title>
- <header> must contain at least the name of the game (choose your name yourself)
- <footer> must contain at least the names of all teammates on your team (including their role, where applicable)
- <nav> (can be placed in <header> must include at least:
    - Before a successful login to an account, a navigation menu item to access: the Sign In feature and the Sign Up feature

- After a successful login to an account, a navigation menu item to access the History feature, the Cancel feature, and the Sign Out feature

3.3. PHP structure

3.3.1. Functional or object-oriented structure

Create most of your code using a structure that allows code reuse and developer collaboration, that is, a structure that includes user-defined functions and/or components of OOP Object-Oriented Programming, including classes, properties, methods, and objects.

3.4. Code and directory structure

3.4.1. Code formatting

To make your website's code easy to review and maintain, it’s mandatory to format it appropriately by:

- Adding indents (a larger left margin indicating what instruction is imbedded inside others);
- Use lowercase letters, except when this is not recommended (e.g. superglobal and constant variables);
- Adding at least 3 significant comments in each file.

For example, you can use an appropriate extension for Visual Studio Code to automatically format your program code.

3.4.2. Formatting directories and files

To stay within the standards, use the following guidelines for directories (folder or folder) and files:

- Use standard file names to name your files, such as index.php and style.css.
- Create multiple files, when necessary, to separate the code instead of creating a single file with a lot of code that has nothing in common. For example, create each class (OOP Object-Oriented Programming) in a separate file, but several functions that have something in common in the same file.

- Use standard directory names to group files, such as css, image, and js.
- When the main directory of your website is saved in the web server (e.g. C:\wamp64\www) and accessed via the web browser (e.g. localhost/website), make sure that the home page is displayed without having to manually write the URL in the address bar of the web browser (e.g. localhost/website/index.php).

3.4.3. Readme file

Create a readme file—a kind of documentation—called READ. ME, README.TXT or README.md containing *markdown markup* or just plain text. Place it in the main directory of your Website. This file should contain the following information:

1. The full name of the developers (teammates of the team) and the contribution of each to the developed program.
2. The game's release date
3. Enumeration and description of all features of the Website (see section 1), including additional features added
4. How the game works, such as the description of levels and the number of lives allocated.
5. Any other relevant technical information on the programme developed, including the programming languages and database management system used, their version and other relevant information.

SECTION 4

ADDITIONAL DOCUMENTS

See the additional documents below, provided separately as an appendix, for more details on the work to be carried out and its evaluation.

1. **Functional diagram and File structure**
    - Functional diagram of website features
    - Solution Plan Diagram (Architectural pattern)
    - Directory and File Structure
    - Directory and File Structure and Description
2. **Project management**
    - Task breakdown and assignment
    - Timeline
3. **Database structure**
    - SQL code for creating database, tables and view
    - SQL code for inserting dummy data
4. **Functional Test Plan**
    - Functional test book
5. **Evaluation grid**
    - Weighting by tasks and tests

Table of Contents