NumericALL

PHP / SQL

Final practical assessment

28th August 2022

Time allowed : 4h00

Marking

At the end of the allotted time, you must deliver all of your files in a Github repository, and a score out of 20 will be given based on to the following scale:

Exercice 1 : points
Exercice 2 : points
Exercice 3 : points
Exercice 4 : points
Other : 2 points ○ Indentation and readability ○ Code Comments ○ Variables name

NB: it is not permitted to copy and use code obtained from other students of this training or previous students of this training - doing so is considered cheating and will be sanctioned.

Tips

If the task seems overwhelming, don't worry! Each step is detailed to move forward little by little.

- Read the entire statement from the start, to know where you are going.
- Take the time to code, commenting on your code as soon as it is necessary, why not by copying the instructions in comment. The proofreader must be able to understand what you have done!
- Keep functions simple, which do only one thing, the better to navigate.
- Focus on PHP rather than CSS/HTML !
- The more your code will be indented and readable, the easier the rest will be!
- Each exercise is independent, be sure to separate them into different files or folders.

Exercise 1: Query

Given this database structure :

INSTRUMENTS(id, name, type, price)
BANDS(id, name, bio)
INSTRUMENTS_IN_BAND(#band_id, #instrument_id)

You need, in a separate TXT/SQL file, to write the query to:

1. display all instruments (name) for the band 'metallica'.
2. display number of instruments per band. For each band, display the name of the band + number of instruments.

Note: Only write the SQL query, no PHP.

Exercise 2: Debugging

A trainee, for which you are responsible, must retrieve a list of users from a database but also give the possibility of adding them using a form.

The main script consists of two parts:

- The list of users and their information
- The form for adding a new user

The trainee begins in PHP programming. It does not indent or comment on its code. As an internship supervisor, you must help him correct and make his code work.
Good indentation of the code is also necessary.

Implementation steps:

1. Download the files at this address: Evaluation Docs
2. Deploy the files index.php and connect.php on your local server and import the SQL script users-install.sql. This will create a new database containing a table "users".
3. Help your trainee to debug his code, correct his errors and make it work.

Exercise 3: Candy Shop

We will work for an Online Candy Shop. The owner wants to be able to add its products and display them. We will take care only of saving the products in the database through a form.

In this exercise, you will have to create a new database and a form.

This form will allow you to add a new candy in the database.

Step 1 :

Create a new database called “candy_shop” and create the ‘candy’ table matching this structure :

CATEGORIES(id, title)
CANDY(id, name, price, #categ_id)

Categories are : “Gummies”, “Lollipops”, “Caramel”.
You’ll have to export the database and attach it to your evaluation folder.

Step 2 :

Now, you need to be able to add a new instrument thanks to the form.

You have to follow those recommandations :

- The form will be processed in Ajax. You can use Ajax in a classic way or with the help of jQuery.
- Category should be a dropdown-list to choose from Gummies/Lollipops/Caramel.
- Validate the input coming from the form.
- Insert the candyin the database
- A success or error message will be displayed.

Exercise 4: Save some money

Step 1 :

You work for a bank.

As part of the site redesign, you must create a BankAccount() class which will have the following private properties:

- an account number (string: 8 characters)
- an amount (int)

Make the getters / setters to validate the data type above as well as the constructor allowing to instantiate the class.

Add two methods:

- Withdraw: Withdraw is possible on the account at any time. The withdraw operation checks if you have enough money on the bank account (amount).
- Deposit: The deposit is possible at any time. The deposit is the addition of an amount to the account

Step 2 :

You can choose between two account types : create an account to *save* money or a *professional account*.

Create the matching classes using inheritance.

SavingAccount() will have an 'interest' property that is defined at the creation of the bank account. It will also have a method to retrieve the current interest of the bank account.

For example: if you have an interest of 0.012 and 500€ on your bank account, your interest is 0.012*500€

ProfessionalAccount() will allow you to withdraw even if the amount is negative.

Instead, it has a 'limit' property. This limit has to be defined when creating the bank account.

For example: if you have a limit of 500, it means you can withdraw up to -500€ (0 is not the limit).

Part 3 :

In a new file, instantiate the class so that you can display 2 different bank accounts (one *SavingAccount* and one *ProfesionalAccount*).

For the 2 accounts, try to withdraw money until it is no longer possible using a loop and the previously created functions (withdraw).