**PRACTICAL WORK :**

1. **Introduction**

In order to create a community space on their website, most webmasters use a member management system. This allows them to retain their visitors, who can then participate more easily in the life of the site.

Interested? Building your member space can not be improvised, there are a number of things to know. We will discover all you need to know about this in this lab.

This practical work will be slightly different from those you have read so far: indeed, we will design together and step by step the member space. Rather than focusing on the source code itself, I'll show you the method, what you need to know, but I will not give you a ready-to-use PHP code. It would not make sense: at this point, you have the level to write the code yourself using the template that I will offer.

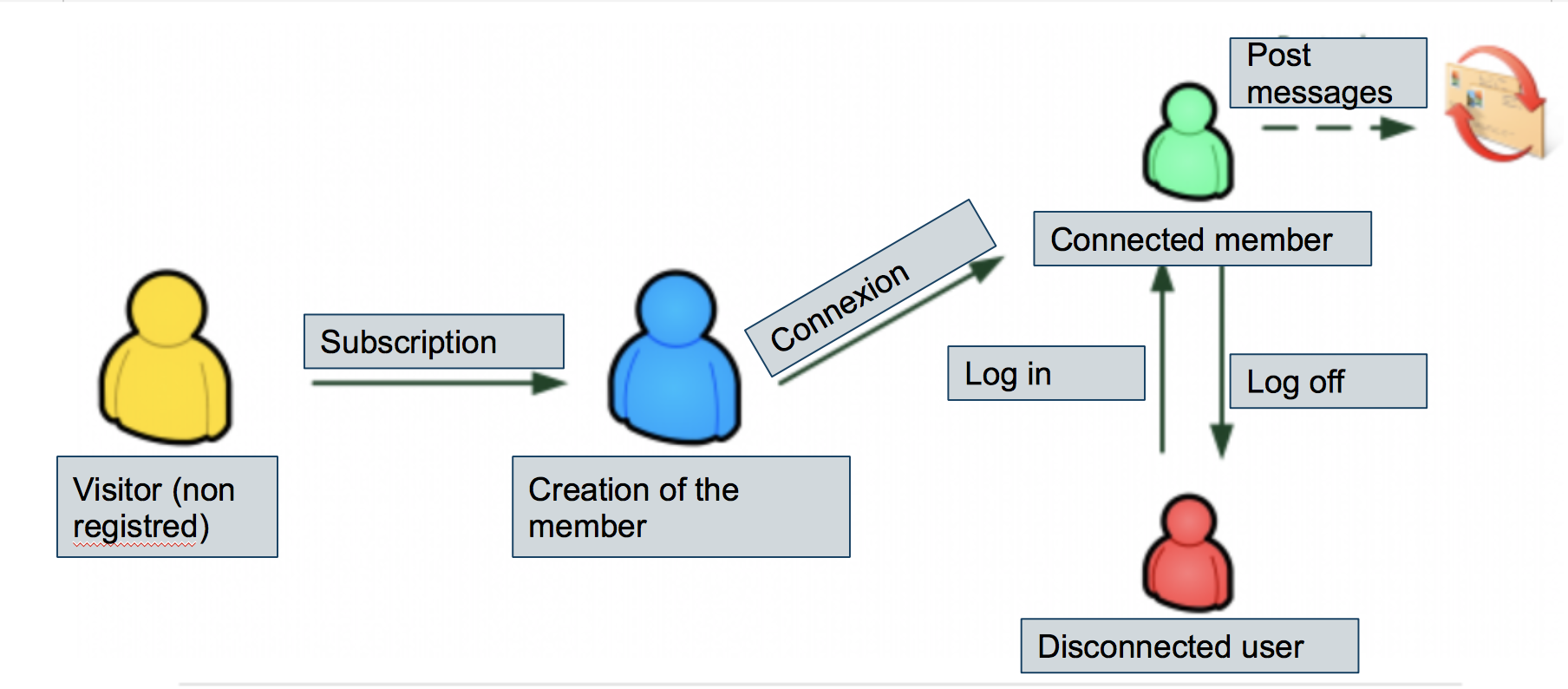
1. **Members Area Design**
   1. What are the features of a member area?

This is the first question we must ask ourselves: what do we want to do concretely? This will also allow us to define what we want to avoid having to conceive, at least initially.

You are probably used to member areas on other sites. Who has never created an account on a website? On Twitter or Facebook? You have already seen a member space, even if the website does not call it exactly like that. You should know that a member space requires at least the following:

* a registration page;
* a login page;
* a logout page.

You can then add other pages, for example to view and edit your member profile. However, you must at least have created the pages I just mentioned. The following figure should give you a good overview of the stages of a member's life, registration and login to the site.



Once this database is ready, it will then be possible to create all the participative space of your site which will be based on the members: the forums, the comments of the news, etc. They are shown in dashed lines in the following figure. We will use the member number (his "id") to link his messages to his account, using the SQL joins that you are familiar with now.

To begin, we will create the MySQL table that will store the members of our site. This is the first step that will allow us to go further and study the creation of the main pages we talked about: registration, login and logout.

* 1. The table of the members

What characterizes a member? Let's try to see what we need to store at least to create the table:

* a pseudonym ;
* a password ;
* an e-mail address;
* a registration date.

Of course, you could add other fields, such as signature, date of birth, or IM address. We will, however, make it simple to start, knowing that it is always possible to add fields to the table later as we have learned.

I propose you to create a table named members with the following fields:

id (int, primary, auto\_increment);

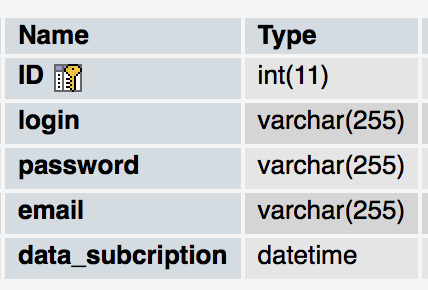
pseudo (varchar 255);

pass (varchar 255);

email (varchar 255);

date\_subscription (datetime).

These fields are summarized in the following figure which presents the table once created under phpMyAdmin.



If you want your members to belong to different groups, it might be interesting to create a table groups listing all groups (member, administrator, moderator ...). You would add a group named group\_name to the membership table, which would allow you to join the two tables as we did with video games and their owners earlier in the class.

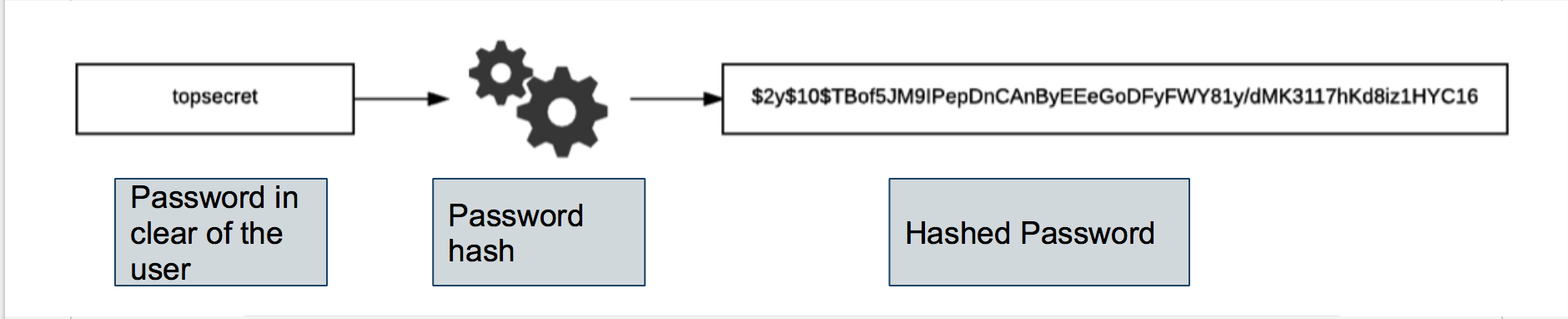
* 1. The problem of the password

One of these fields deserves special attention: the one that stores the password. When they sign up, your visitors will confidently send a password to your site. It is very likely that they use the same password on many other sites. Although this is a very bad security habit (ideally, you would have to use a different password per site), this is unfortunately extremely common.

Knowing this, you have some moral and ethical obligation as webmasters: you should not store the passwords of your visitors in the database. If it falls into the wrong hands (this could happen in a critical case, such as hacking your site, which I do not wish you), someone would have access to all the passwords of your members and could use it to steal their accounts on other sites!

However, I must store the password of my members if I want to be able to ensure that they are the right people! It's an impossible problem to solve!

That's what you believe, and yet the solution exists: it's called hashing. This is a function that transforms any text into a hexadecimal number that represents the password but is unreadable, as shown in the following figure.



To hack a password, there are several functions that rely on different algorithms. The good function to use is **password\_hash**, which will choose for you the best algorithm.

sha1, md5, sha256, sha512 ... are no longer considered safe hash functions today. Even if you see them in some (old) codes, do not use them. Use password\_hash who chooses the best algorithm for you.

The particularity of the hash is that it works in one direction: it is impossible to find the original password once it has been chopped. In addition, a hash (name given to the hashed version of the password) is unique: it corresponds to one and only one password.

You will store the hashed version of the password, which will be passed to the DB by the function password\_hash. When a visitor wants to log in, he will send you his password that you will chop again and compare with the one stored in the database. If the two hashed passwords are the same, then it means that the visitor has entered the same password as when they registered.

1. **Realization of the main pages of the members area**

We have determined a little earlier the list of pages we need at least to manage our members:

registration;

connection;

disconnection.

We will not write the code of these pages but we will review what you need to know to make them correctly.

* 1. The registration page

The registration page usually consists of four fields:

* desired pseudonym;
* password ;
* confirmation of the password (to avoid typing errors);
* E-mail address.

It is recommended to limit as much as possible the amount of information requested. The visitor wishes to be able to register very quickly. If he falls on a page with many fields to fill, he is likely to drop. Let him fill in the other fields (like his signature, his instant messenger and his date of birth) in a second time when he will be registered.

Some sites do not ask for a login and use the e-mail address as their identifier. This is a possibility, especially on e-commerce sites, but if your members will be required to post messages on forums, it is preferable that they register with a pseudonym.

The following figure shows a basic registration form.



The password field is of type password to prevent another person from reading the password on the screen. For this reason, it is strongly recommended that you re-enter the password in case the visitor has made a typo that he / she could not have seen.

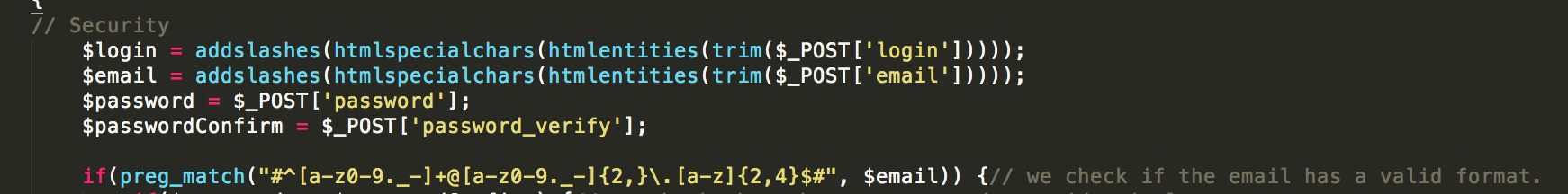
Before registering the user in the database, it will be necessary to make a number of checks.

Is the pseudonym requested by the visitor free? If it is already present in the database, you will have to ask the visitor to choose another one.

Are the two passwords entered the same? If there is an error, invite the visitor to re-enter the password.

Does the email address have a valid form? You can use a regular expression to check it.

Since we saw together the regular expression you will have to use it for the email and other verifications you will need to do for the security of your form :



If all these conditions are met, you can insert the user into the database. As I told you earlier, it is very strongly advised to hash the password before storing it, so that it is no longer "readable". You will have to use the method [password\_hash](http://php.net/manual/en/function.password-hash.php)

We first check the validity of the information as I told you before, then we hash the password and finally we can insert the member in the database. Under phpMyAdmin, we see the member appear as in the following figure.

Its hashed password is not readable and this assures us that we can not "steal" it.

Okay, but then how do we check if the person who connects has the right password ?!

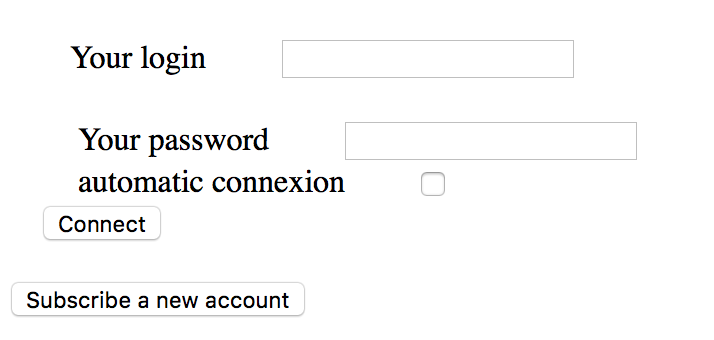
Ah ah! It seems impossible? Yet there is a way, which is stupid when you think about it. We will see during the login step how to check if the member has entered the correct password.

Additional steps could be added to enhance the security of registration. In particular, find out about the Captcha systems that ask the visitor to copy a word from an image to verify that it is not a robot. On the other hand, you could request a confirmation by e-mail to verify that the address is correct.

* 1. The login page

Now that the member is created, he must be able to login to your site. For this, we will use the session system that is made available by PHP and that we learned to use earlier in this course.

Usually, one asks at least the pseudonym (or login) and the password of the member. To make his life easier, he can be offered an automatic connection option that will prevent him from having to log in again each time he visits the site (following figure).



The page that receives the data from the login form must verify the password by comparing it to the one stored in the database with the **password\_verify** function. This function will actually hasher the password of the user who just logged in and compare it to the one that was stored in the database.

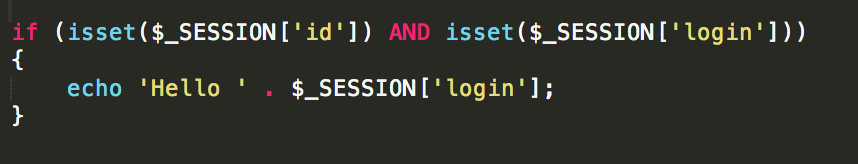
If there is a member with the same pseudonym and hash password, then login is allowed and session variables can be created. Otherwise, we return an error indicating that the username or password is invalid.

We recover the password in the database and compare it with the one sent by the login form thanks to [**password\_verify**](http://php.net/manual/en/function.password-verify.php), which returns us TRUE or FALSE.

If all is good, we can create the session variables and store there for example the id and the pseudonym of the member.

If there is an error, it is better to display a generic message instead of saying "It's a bad username" or "It's a bad password". If someone tries to steal access to a user, the less he knows, the better!

From now on, on all the pages of the site, one will be able to indicate to the member that it is connected thanks to the presence of the variables $ \_SESSION.



If the member wants to be reconnected automatically (which it is advisable to do only on a personal computer, and not on a computer shared with other people!) even when he closes the browser windows, I invite you to create also two cookies that will store respectively:

* the pseudonym;
* the password hashed.

or you can use the id

Thus, if an unconnected visitor who has these two cookies presents himself, you will only have to check if a member corresponds to this information in database and you will be able to connect it automatically, without having to use the login form. Again, we take some security by storing the hashed password in a cookie and not the real password.

Be aware that new centralized login methods have been appearing for some time. They are called OpenID, Facebook Connect, Windows Live ID, etc. By doing a few additional manipulations, you can allow your visitors to connect to your site by entering their Facebook, Windows Live, Twitter or Google, which eliminates the need to provide a password specifically for your site.

* 1. The logout page

After a period of inactivity, the member's session is automatically destroyed and disconnected. If it loads a page of the site again, it will appear disconnected, unless it has activated the automatic connection which will have the effect of reconnecting it immediately and transparently thanks to its cookies.

If the disconnection is automatic after a certain time (the famous timeout), it is still necessary to propose a connection link. The logout page will have to delete the contents of $ \_SESSION, terminate the session system and delete the automatic login cookies if they exist.

1. **Going further**

We have seen together how to set the foundation for a membership management system. Of course, we have done the minimum and now it is up to you to play to improve this script as you see fit. :-)

Here are some ways to complete your member space.

* Set up a member profile page. You can display all sorts of information, such as his email (but it's better to ask him before), his instant messaging address, his date of birth, his passions, his work, the name of the city where he lives, etc. All of this information can be stored in new fields in the member table.
* Propose to the member if he wishes to change his identifiers: his pseudonym and his password. It is common that a member wants to change his pseudonym some time after registering, but above all it is vital that he can change his password at any time in case this one is compromised! Even if the member is already connected, I advise you to ask him again his current password before authorizing him to change, for security reasons.
* Give the member the ability to choose from several navigation options. Everyone does not use your website the same way, maybe some would like to have a menu at the top of the pages rather than another, maybe others would prefer to navigate with a dark design, and so on.

Your member's area should start to be full from there!