# ICS 3104

# Lab 2

**Preliminaries**

This lab is a continuation of Lab 1.

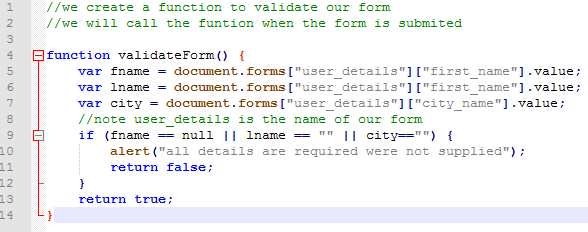
# PART 1

# Form validation

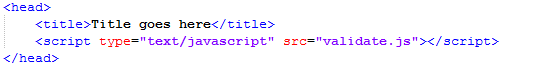
## Client side validation using JavaScript

We create a JS file to validate out form

Create a validate.js file as shown



Include the JS file in the head section of lab1.php as shown



Include this parameter (***onsubmit***) in our form as shown. Note that our JS function is called when our form is submitted.



Test the form by submitting a form that does not have all the details.

## Server side validation

If a user disables JavaScript, then your validation above will not work. We need to perform server side validation

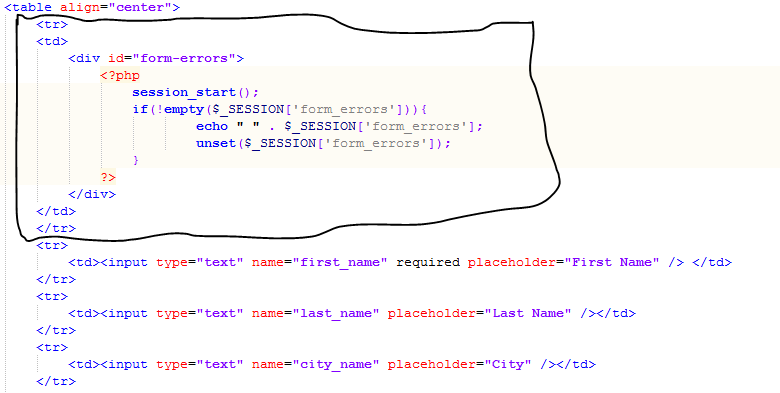
We proceed as follows

Disable JavaScript in your machine for Google chrome check here <http://www.technipages.com/google-chrome-enable-or-disable-javascript> and for Mozilla check here <http://www.wikihow.com/Disable-JavaScript> .

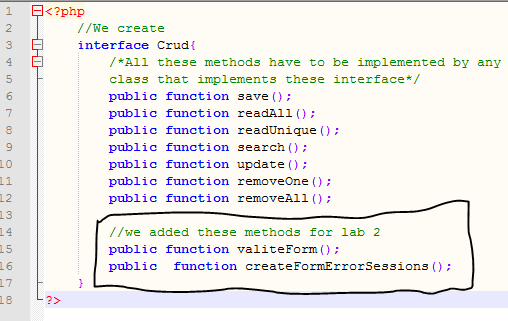
Now try to submit form. What do you notice?

We write some code in out html file, PHP files and a new CSS file

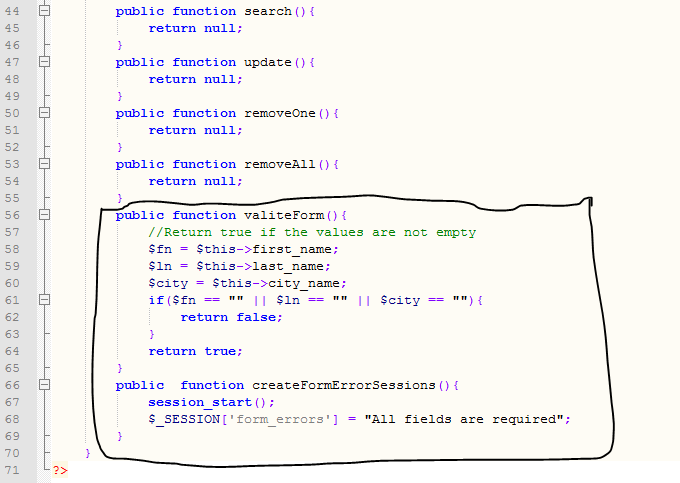
Add the following in the lab1.php



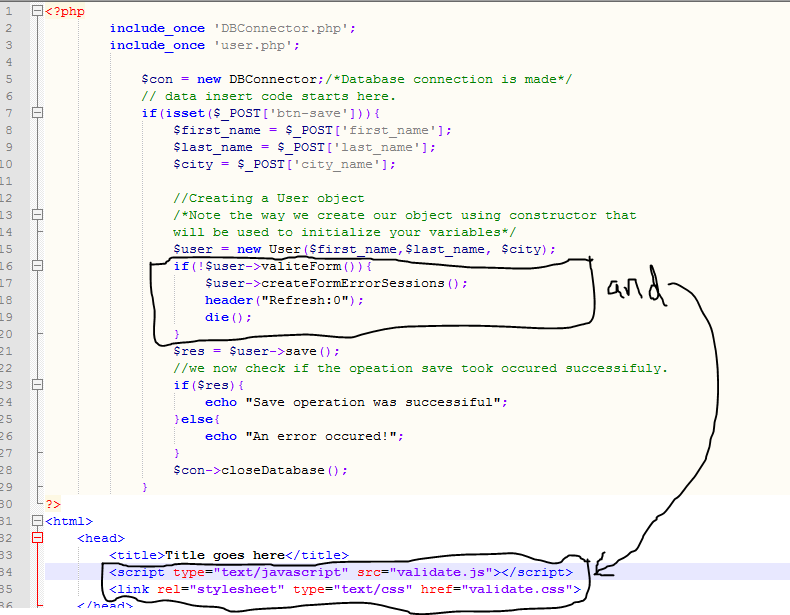
Add following functions in the *Crud interface* and implement them in the User class as shown



Implement the as shown below

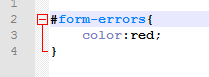


Add the following section the lab1.php



Note how we have included the JS and CSS files in the head sections

Lastly, create a CSS file by name validate.css. Its contents are shown below



Is *#form-error* and *id* or *class*?

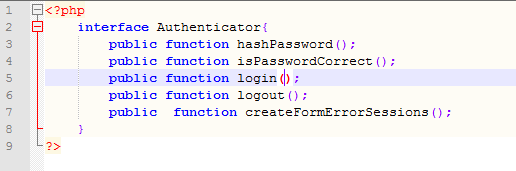
Now with JS disabled in your browser, try submitting an empty file. You should be able to validate your form from server side. (Note that you have to input value for first name since it has been marked *required*; - **an HTML 5 attribute**).

# PART 2

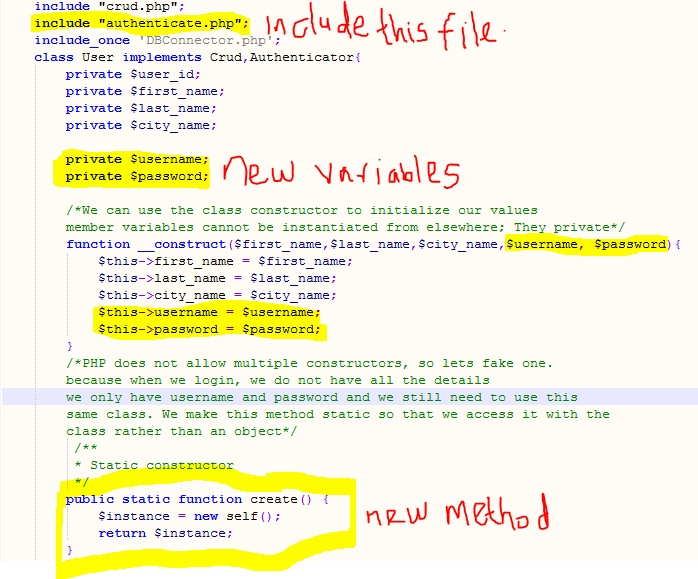
This is a continuation of part 1 of this lab.

Here we look at how we can register a user, login and logout in this system. We also learn more about ***interfaces*** classes.

We create a new interface by name Authenticator in a file called authenticator.php as shown below



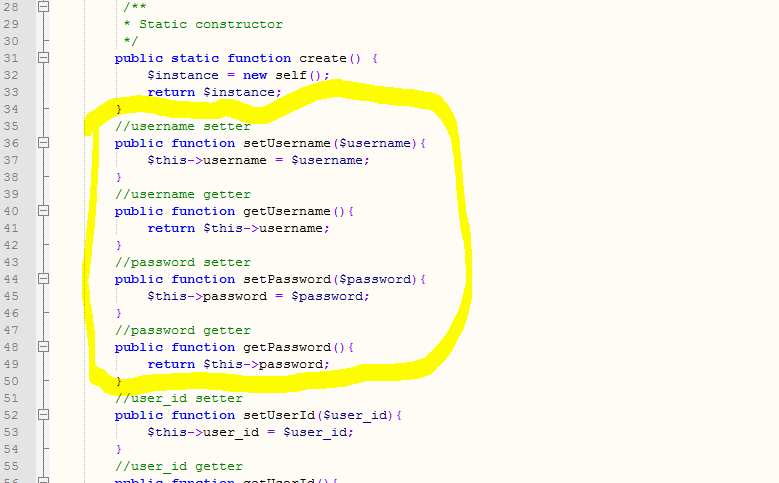
Our new User class will implement 2 interfaces now as shown and we add two new member variables. Also note that our ne constructor should have 2 more parameters plus 2 new getters and setters for the new member variables.



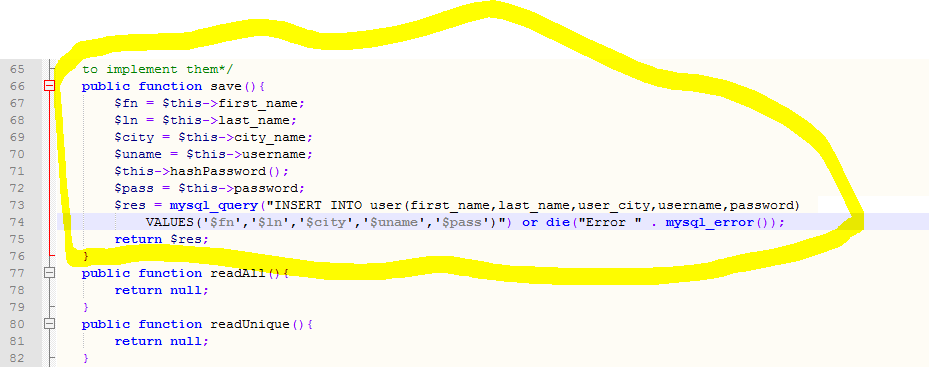
We were able to implement 2 interfaces and we have to implement all the methods of the implemented interfaces

PHP does not support multiple constructors. So we fake the creation of an object using the method *create* (). This method returns the object of the class without necessarily using its constructor! Sounds interesting?

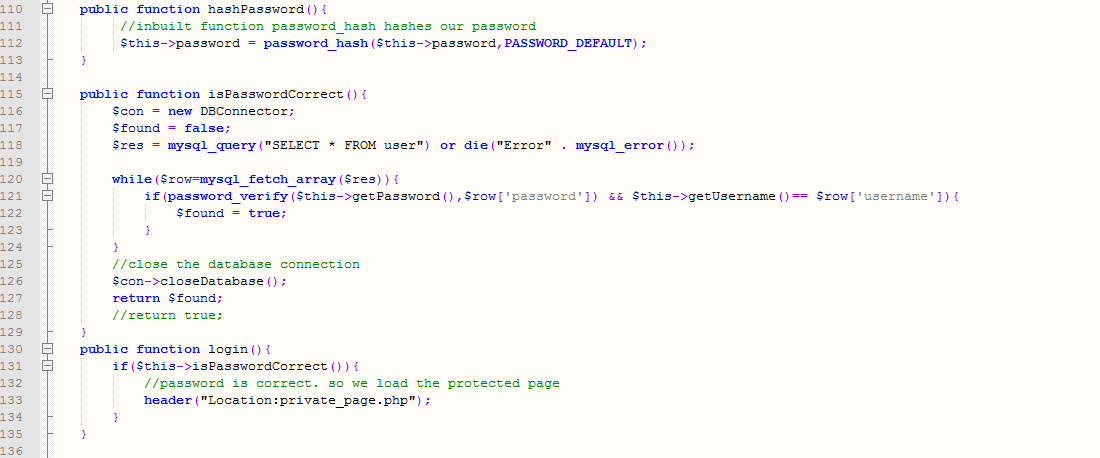
Now we add getters and setters for the new variables username and password. Why do you think we need them now that they can also be initialized by our constructor?

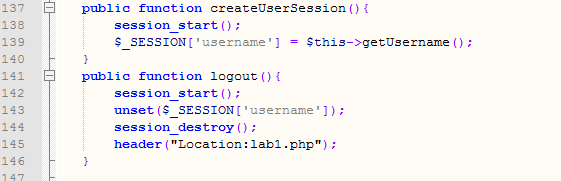


Our save () method new some change now that the user has to save details plus user name and password as shown below



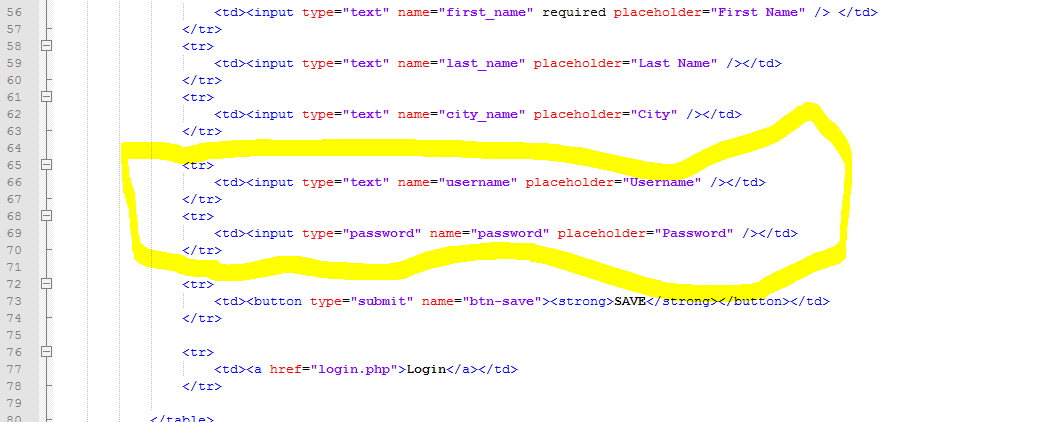
Now let’s add and implement the methods from Authenticator as shown below. Note that because our User class implemented them, all these methods will be implemented in the User class as shown

erro



Now our class logic is complete. We now add more controls to our user details form, we create a login form (login.php), a private page (private\_page.php) which we need to protect and a logout processing file (logout.php)

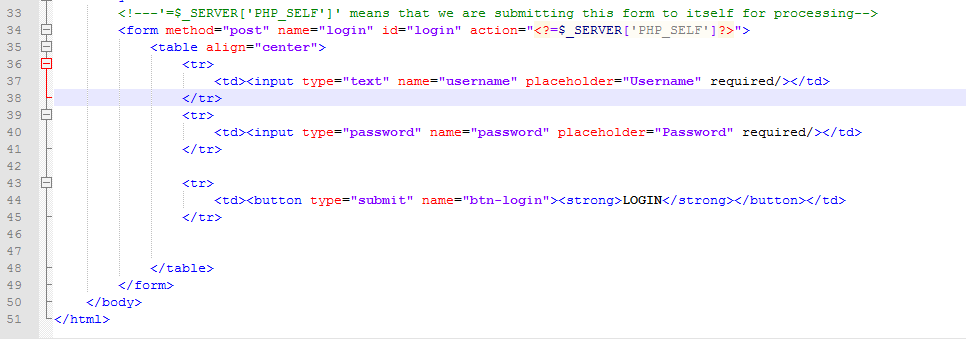
Adding more controls to lab1.php



Note line 77, we have added a link that will take a user to login.php

We create a login.php file plus the processing logic for that form. Note that we wrote all the codes in the User class. Now we are just implementing the class.





We now create our private\_page.php and its contents

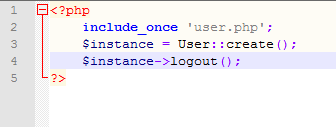
From line 1-6. Note the way we call method start\_session() and we proceed by checking if the session is set. If session is not set, then direct the user to login page to login, otherwise we allow the user to view the page.

In line 16, we create a link for a user to logout. We direct the user to logout.php.



No, what happens when the user clicks logout?

See logout.php below



Explain what is happening in line 3. Remember that we are using the functionality we put in method logout () in class User.

Finally, we need to alter our database table, accommodate our new application.

Execute this query against your table user



Now test your application. Check if you can register a user, login and logout. Also check if you can view the private\_page.php without the user logged in.

Also view your records in the database, notice how your passwords look like.

## Your task

You know that 2 users cannot have the same username.

Create a method in the class User called isUserExist() and write the logic that would prevent the user from using a username that is already in use. You should be able to report if such an attempt occurs.