Intro to authentication - Text lecture

We want our application to allow users to log in using a username and password. The application will check that the user exists in the database and confirm that their password is correct.

We also want to force users to log in. If they try to access pages when they aren't logged in, we need to reroute them to the login page.

Php offers a way to store information that can be accessed across multiple pages call sessions. A session is unlike a cookie in that the information is stored on the server instead of the user's machine.
http://www.w3schools.com/php/php_sessions.asp
Well put sessions to work after we create the MySQL table.
Start by creating a User Definition table

```
CREATE TABLE User_Dfn ( User_ID INT NOT NULL AUTO_INCREMENT
                    ,username varchar(20) NOT NULL
                    ,pwd varchar(40) NOT NULL
                    ,PRIMARY KEY (User_ID));
```

Now create a page to add new users. Create a new file named AddUser.php
This page will simply collect the user name and password that we want to use, and send it in a post action to AddUserSubmit.php.

```
<?php
/* begin our session */
session_start();
?>
<html>
<head>
  <title>Add New User</title>
</head>
<body>
<h2>Add New User</h2>
<form action="AddUserSubmit.php" method="post">
  <fieldset>
    <p>
      <label>Username</label>
      <input type="text" name="username" value="" maxlength="20"
required/>
      <i>(4-20 characters)</i>
    </p>
```

```
    <p>
      <label>Password</label>
      <input type="password" name="pwd" value="" maxlength="20"
required/>
      <i>(4-20 characters)</i>
    </p>
    <p>
      <input type="submit" value="Add User" />
    </p>
  </fieldset>
</form>
</body>
</html>
```

Next, Add a new file titled AddUserSubmit.php

For filter_var FILTER_SANITIZE_STRING function check out the link below:
http://www.w3schools.com/php/filter_sanitize_string.asp
This function removes special characters which can cause many issues including possible security risks.

```
<?php
require_once('SQLFunctions.php');
session_start();
/* Check that username, and password are populated*/
if(!isset( $_POST['username'], $_POST['pwd']))
{
    $message = 'Please enter a valid username and password';
}
/* Check username length is not more than 20 and not less than 4*/
elseif (strlen( $_POST['username']) > 20 || strlen($_POST['username']) < 4)
{
    $message = 'Incorrect Length for Username';
}
/* Check password length is not more than 20 and not less than 4*/
elseif (strlen( $_POST['pwd']) > 20 || strlen($_POST['pwd']) < 4)
{
    $message = 'Incorrect Length for Password';
}
/* Check the username for alpha numeric characters */
elseif (ctype_alnum($_POST['username']) != true)
```

```
{
   $message = "Username must be alpha numeric";
}
/* Check the password for alpha numeric characters */
elseif (ctype_alnum($_POST['pwd']) != true)
{
       $message = "Password must be alpha numeric";
}
else
{
   /* Store username and pwds as variable */
   /*Use filter_var to remove special characters from the inputs*/
   $username = filter_var($_POST['username'], FILTER_SANITIZE_STRING);
   $pwd = filter_var($_POST['pwd'], FILTER_SANITIZE_STRING);
   /* Encrypt with password with sha1, a cryptographic hash function  */
   /* Never store plain text passwords in the database*/
   $pwd = sha1( $pwd );
```

We will continue building this in the next video

Complete user sign-up - Text lecture

Below is the full AddUserSubmit.php file:

```
<?php
require_once('SQLFunctions.php');
session_start();
/* Check that username, and password are populated*/
if(!isset( $_POST['username'], $_POST['pwd']))
{
   $message = 'Please enter a valid username and password';
}
/* Check username length is not more than 20 and not less than 4*/
elseif (strlen( $_POST['username']) > 20 || strlen($_POST['username']) < 4)
{
   $message = 'Incorrect Length for Username';
}
/* Check password length is not more than 20 and not less than 4*/
elseif (strlen( $_POST['pwd']) > 20 || strlen($_POST['pwd']) < 4)
{
```

```
    $message = 'Incorrect Length for Password';
}
/* Check the username for alpha numeric characters */
elseif (ctype_alnum($_POST['username']) != true)
{
    $message = "Username must be alpha numeric";
}
/* Check the password for alpha numeric characters */
elseif (ctype_alnum($_POST['pwd']) != true)
{
        $message = "Password must be alpha numeric";
}
else
{
    /* Store username and pwds as variable */
    /*Use filter_var to remove special characters from the inputs*/
    $username = filter_var($_POST['username'], FILTER_SANITIZE_STRING);
    $pwd = filter_var($_POST['pwd'], FILTER_SANITIZE_STRING);
    /* Encrypt with password with sha1, a cryptographic hash function  */
    /* Never store plain text passwords in the database*/
    $pwd = sha1( $pwd );
    try
    {
         /*Connect to CRUD
Database  mysqli(Server,User,Password,Database)*/
        $link = connectDB();
        /* Check that username does not already exist */
        $sql = "SELECT 1 FROM User_Dfn WHERE username = '".$username."'";
        if($result=mysqli_query($link,$sql))
        {
            if(mysqli_num_rows($result)>=1) {
              $message = "Username already exists";
            } else {
              /* Prepare the sql insert statement */
              $sql = "INSERT INTO User_Dfn (username, pwd ) VALUES
('".$username."', '".$pwd."')";
              if (mysqli_query($link, $sql)) {
                $message = 'New user added';
```

```
            } else { echo  "<br>Error: " . $sql . "<br>" . mysqli_error($link);  }
          }
       }
    }
    catch(Exception $e)
    {
       $message = 'Unable to process request';
    }
}
?>
<html>
  <head>
    <title>Add New User</title>
  </head>
  <body>
    <!-- Message is a variable that was populated previously based on the
php above  -->
    <p><?php echo $message; ?>
  </body>
</html>
```

Try the form out.

There are all sorts of good security practices that we could implement, but for now, this will accomplish basic functionality.

Login users - Text lecture

Now lets create a Login page Login.php

```
<html>
<head>
<title>CRUD Login</title>
</head>
<body>
  <h2>CRUD Login</h2>
  <form action="LoginSubmit.php" method="post">
  <fieldset>
    <p>
      <label>Username</label>
```

```
      <input type="text" name="username" value="" maxlength="20" />
    </p>
    <p>
      <label>Password</label>
      <input type="password"  name="pwd" value="" maxlength="20" />
    </p>
    <p>
      <input type="submit" value="Login" />
    </p>
  </fieldset>
  </form>
</body>
</html>
```

Create another file LoginSubmit.php

```
<?php
require_once('SQLFunctions.php');
session_start();
/* Check if the user is already logged in */
if(isset( $_SESSION['user_id'] ))
{
    $message = 'User is already logged in';
}
/* Check that username and password are populated */
if(!isset( $_POST['username'], $_POST['pwd']))
{
    $message = 'Please enter a valid username and password';
}
/* Check username length */
elseif (strlen( $_POST['username']) > 20 || strlen($_POST['username']) < 4)
{
    $message = 'Incorrect Length for Username';
}
/* Check password length */
elseif (strlen( $_POST['pwd']) > 20 || strlen($_POST['pwd']) < 4)
{
    $message = 'Incorrect Length for Password';
}
/* Check username for alpha numeric characters */
```

```
elseif (ctype_alnum($_POST['username']) != true)
{
    $message = "Username must be alpha numeric";
}
/* Check password for alpha numeric characters */
elseif (ctype_alnum($_POST['pwd']) != true)
{
        $message = "Password must be alpha numeric";
}
else
{
    /* Store username and pwds as variables*/
    $username = filter_var($_POST['username'], FILTER_SANITIZE_STRING);
    $pwd = filter_var($_POST['pwd'], FILTER_SANITIZE_STRING);
    /* Encrypt password with sha1*/
    $pwd = sha1( $pwd );

    try
    {
         /*Connect to CRUD
Database  mysqli(Server,User,Password,Database)*/
        $link = connectDB();
        /* Prep SQL statement which will compare the user credentials with
what is stored in the database*/
        $sql = "SELECT User_ID FROM User_Dfn WHERE username =
'".$username."' AND pwd = '".$pwd."'";
        /*echo $sql."<br>";*/

        /*Run the query*/
        if($result=mysqli_query($link,$sql))
        {
          /*assign the User_id from the database to the session user_id*/
          while($row = mysqli_fetch_assoc($result)) {
            $user_id = $row['User_ID'];
            /*echo "<br>user_id=".$user_id;*/
            /* Set the session user_id parameter */
            $_SESSION['user_id'] = $user_id;
            $_SESSION['timeout'] = time();
```

```
            /*header("Location: ToDoApp.php"); UNCOMMENT this once tested
there will be ins below*/
            $message = 'You are now logged in';
          }
        }
          if($user_id == false)
          {
            $message = 'Login Failed';
          }
    }
    catch(Exception $e)
    {
        $message = 'Unable to process request';
    }
}
?>
<html>
<head>
<title>LoginSubmit</title>
</head>
<body>
<p><?php echo $message; ?>
</body>
</html>
```

Test this by attempting to log in intentionally fail the validations.
Once tested, uncomment the header location row that forwards users to ToDoApp.php once they are logged in. Then you should be able to test logging in with correct credentials
Now lets create a logout page Logout.php

```
<?php
session_start();
// Unset all of the session variables.
session_unset();
// Destroy the session.
session_destroy();
?>
<html>
  <head>
```

```
    <title>Logged Out</title>
  </head>
  <body>
    <h1>You are now logged out.</h1>
  </body>
</html>
```

Test it out and ensure you have closed the session by logging out!

Enforce log in - Text lecture

Create a file titled TestLoginStatus.php

```
<?php
require_once('SQLFunctions.php');
session_start();
if(!isset($_SESSION['user_id']))
{
    $message = 'You must be logged in to access this page';
}
else
{
    /*copy the session user_id to a local variable*/
    $user_id = $_SESSION['user_id'];
    /*echo "<br>user_id=".$user_id;*/

    try
    {
        /*Connect to CRUD Database*/
        $link = connectDB();
        /* Prep SQL statement to find the user name based on the user_id */
        $sql = "SELECT username FROM User_Dfn WHERE User_ID =
".$user_id;
        /*echo "<br>".$sql."<br>";*/

        /*execute the sql statement*/
        if($result=mysqli_query($link,$sql))
        {
```

```
        /*from the sql results, assign the username that returned to the
$username variable*/
        while($row = mysqli_fetch_assoc($result)) {
          $username = $row['username'];
          /*echo "<br>username=".$username;*/
        }
      }
      /* Return Status to User*/
      if($username == false)
      {
         $message = 'Access Error';
      }
      else
      {
         $message = 'Welcome '.$username;
      }
   }
   /*if something goes wrong, return the following error*/
   catch (Exception $e)
   {
      $message = 'Unable to process request.';
   }
}
?>

<html>
<head>
<title>Test Login Status</title>
</head>
<body>
<h2><?php echo $message; ?></h2>
</body>
</html>
```

Now test out this page -> If you aren't logged in, you will see a message that you must be logged in to access this page

If you are logged in, you will see a welcome message with your username

Now that we can log in and out of the site using session, we need to enforce this which we will start in the next video

Sessions - Text lecture

First, let's create a new page titled session.php

This page will load at the beginning of the other pages. It will redirect users to the login page if they haven't yet authenticated.

```
<?php
session_start();
if(!isset($_SESSION['user_id']))
{
    /* Redirect If Not Logged In */
    header("Location: Login.php");
    exit; /* prevent other code from being executed*/
} else {
  /*we are going to start tracking a new session variable we will call timeout.
   by comparing the session timeout plus 600 seconds to the current time,
   we can force users to the logout page when they attempt to access the
page, after 10 mins of inaction*/
  if ($_SESSION['timeout'] + 10 * 60 < time()) {
    /* session timed out */
    header("Location: Logout.php");
  } else {
    /*if the user isn't timed out, update the session timeout variable to the
current time.*/
     $_SESSION['timeout'] = time();
  }
}
?>
<!-- The following navigation links are for convenience  -->
<div align="right">
  <a href="ToDoApp.php">Home</a>
  <a href="AddUser.php">New User</a>
  <a href="Login.php">Log On</a>
  <a href="Logout.php">Log Off</a>
```

```
  <a href="TestLoginStatus.php">Test Login Status</a>
</div>
```
The timeout we are enforcing is for 10 mins (10*60), you will want to change this to one min when testing.

Add The following code to the beginning of each for the following php pages:
```
<?php
include('session.php');
?>
```
AddUser.php
AddUserSubmit.php
CreateToDo.php
CreateToDoSubmit.php
DeleteToDo.php
Logout.php
TestLoginStatus.php
ToDoApp.php
UpdateToDo.php
UpdateToDoCommit.php
Do not add to Login.php, we wouldn't want to be redirected prior to completing the login action.

Now that we have implemented logins for multiple users, we should make sure users see their own todo list and not other's items.
Let's accomplish this by adding an additional column in the ToDos table for User_ID (foreign key):
```
ALTER TABLE ToDos
ADD COLUMN User_ID INT AFTER ToDoID;
```
Then in the CreateToDoSubmit.php page, let's add the additional column

Add a variable for user_id under the other ones (title, description etc.)
```
$user_id = $_SESSION['user_id'];
```
Update the SQL for the new User_ID field
```
$sql = "INSERT INTO ToDos (User_ID, ToDoTitle, ToDoDescription, ToDueDate, EntryTS) VALUES
(".$user_id.",'".$tdTitle."','".$tdDescr."','".$tdDate."', NOW());";
```
This will make the rows keep the user_id associated with the ToDos.
Now let's update the read, update ToDoApp.php as follows:

Update the top to

```
<?php
  include('SQLFunctions.php');
  $user_id = $_SESSION['user_id'];
?>
```

Update the sql to

```
  $sql="SELECT ToDoTitle
              ,ToDoDescription
              ,DATE_FORMAT(ToDueDate,'%m-%d-%Y')
              ,ToDoID
          FROM ToDos
          WHERE User_ID = ".$user_id;
```

Now test this out.

You should notice on the main page ToDoApp.php, that you only see tasks under the current user. You can update and delete only the records from the logged in user.

Try logging in as different users. Try updating ToDo items.