# Tutorial 9

Editor – Brackets

### *Goals*

Introduction to PHP and MySql.

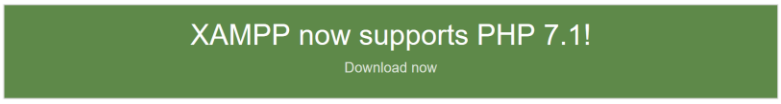
- Set up and configuration of Xampp
- Learning Data flow

Things to note: Each week Xampp will need to be installed. Xampp is Windows software, similar software is available for Mac, called Mamp.

## Installing and configuring Xampp

Go to the website: [https://www.apachefriends.org/index.html](https://www.apachefriends.org/index.html)

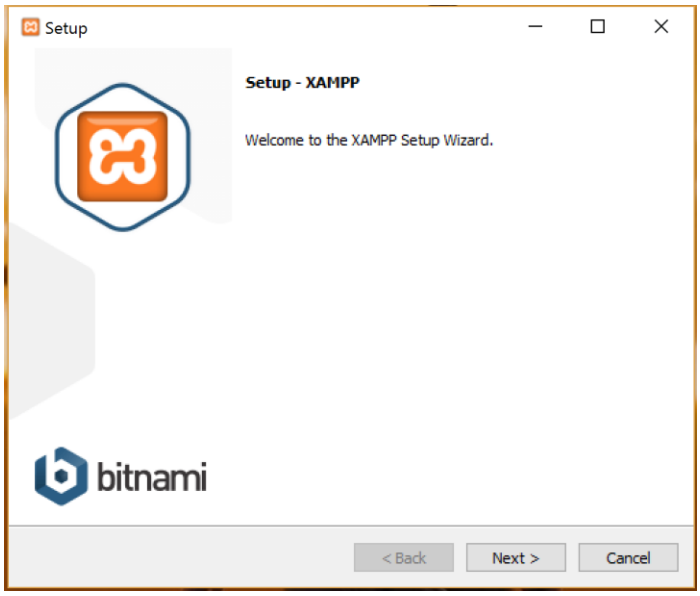

Download the Xampp with PHP 7.1, click on the following link



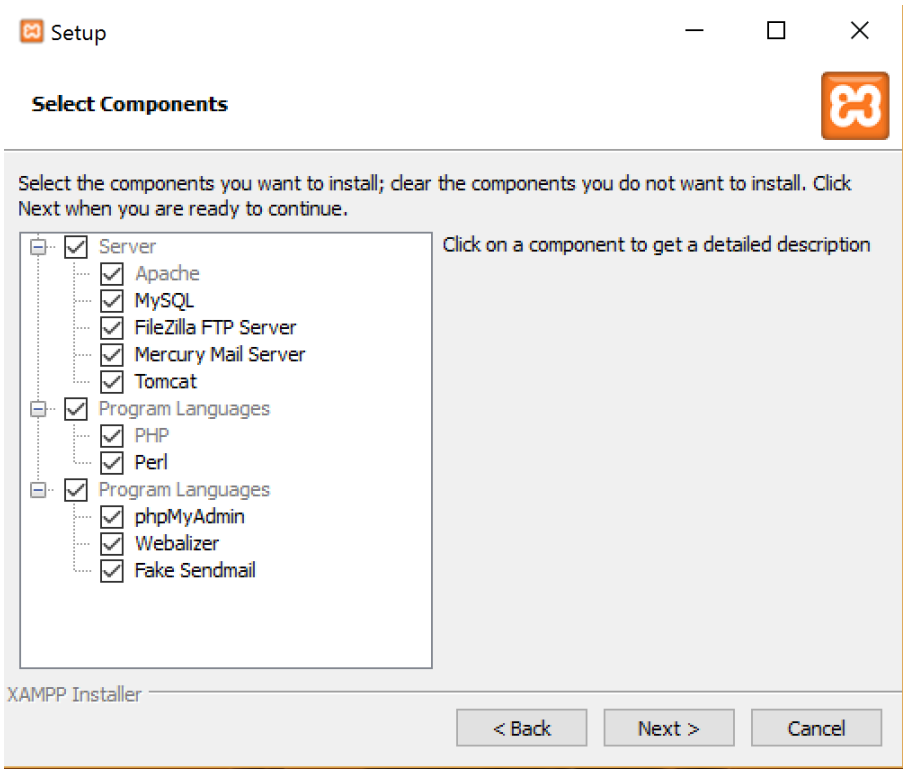

Then click on


Download to the desktop, Once there, double click on the file and following the prompts

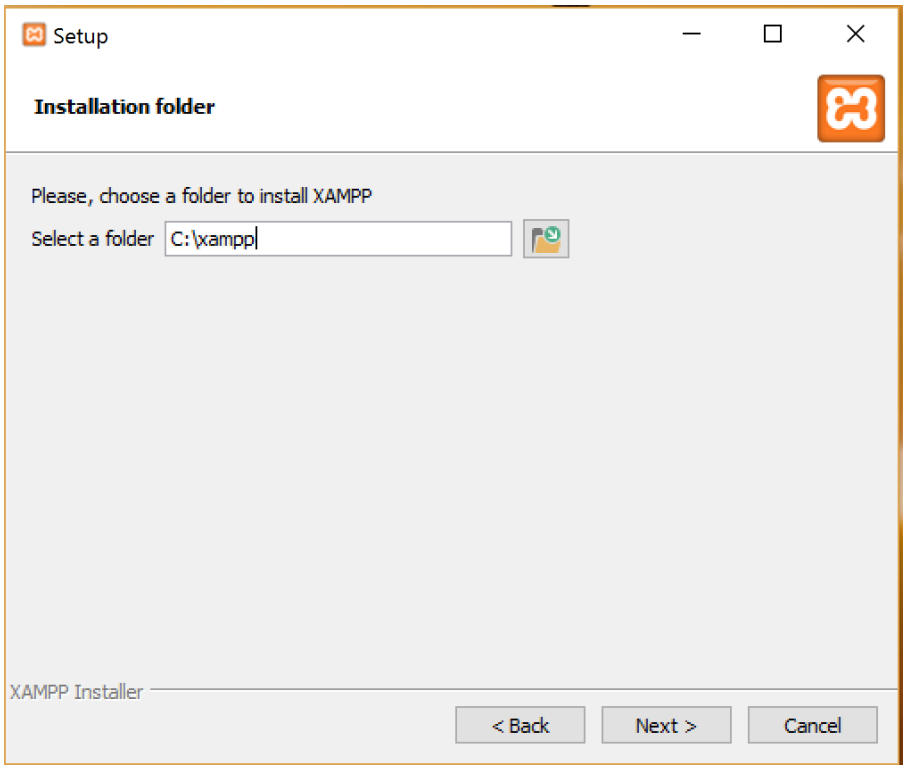

Click Next




Click Next

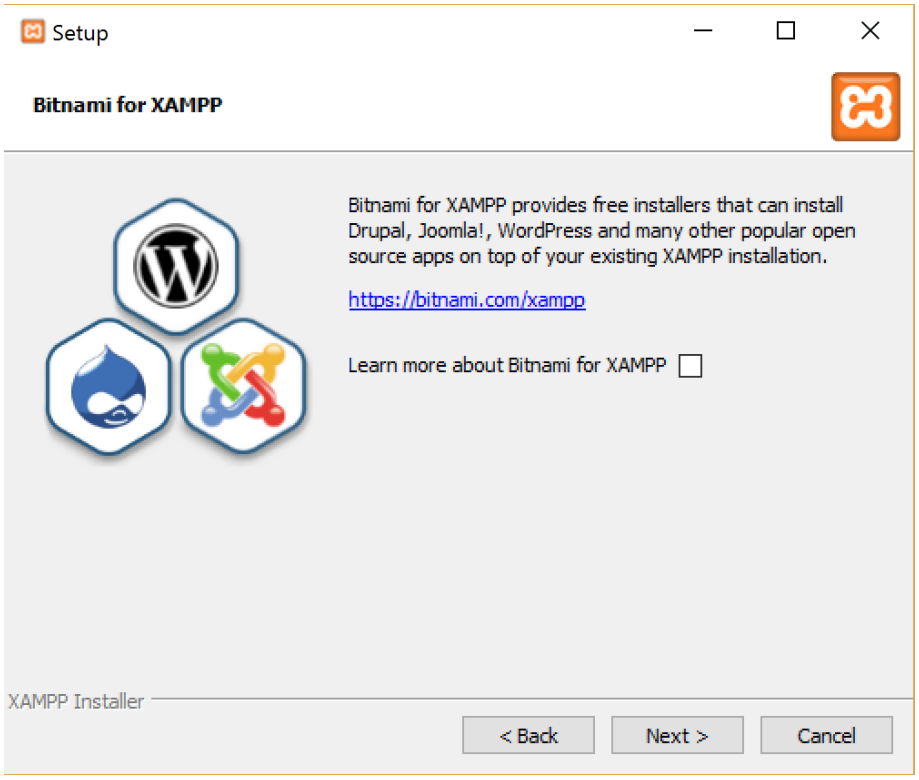


Click Next

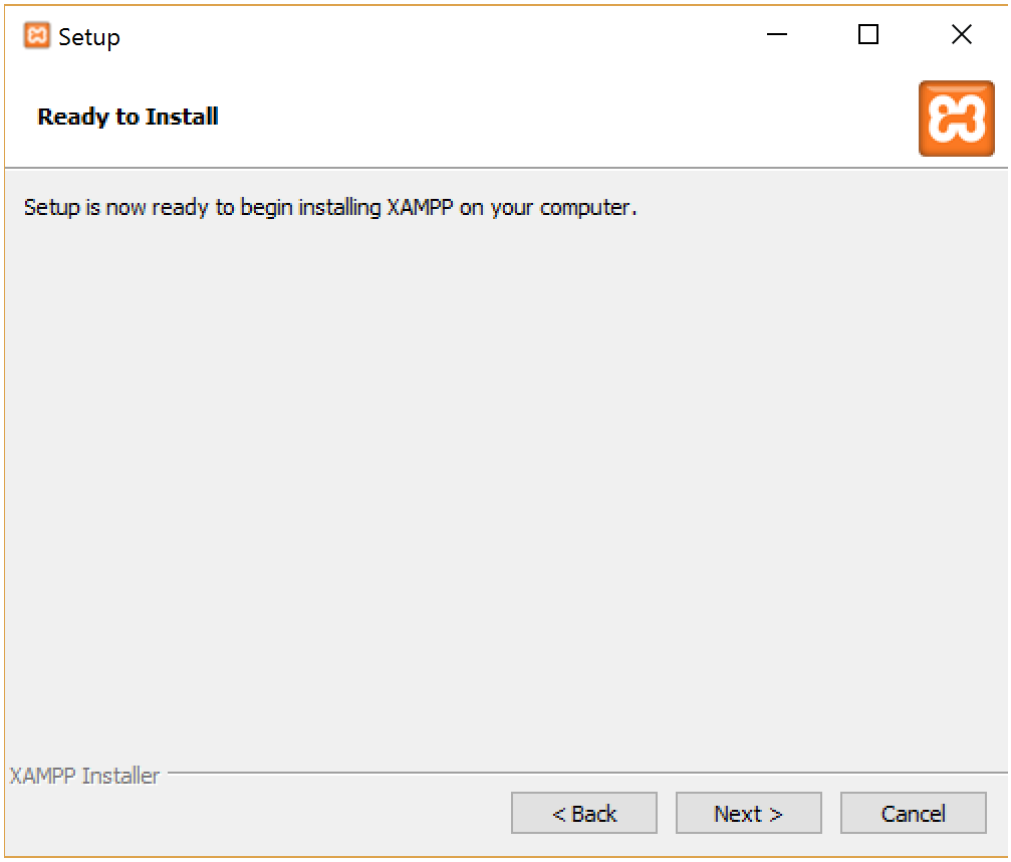



NB: This is the installation folder, all of your web pages will belong in c:\xampp\htdocs

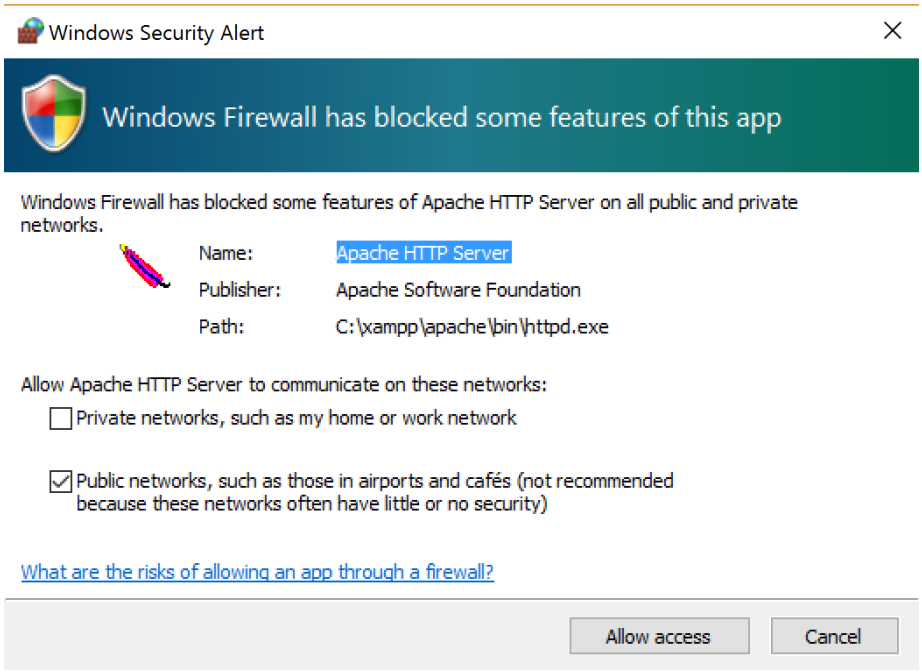
Untick Bitnami and click Next



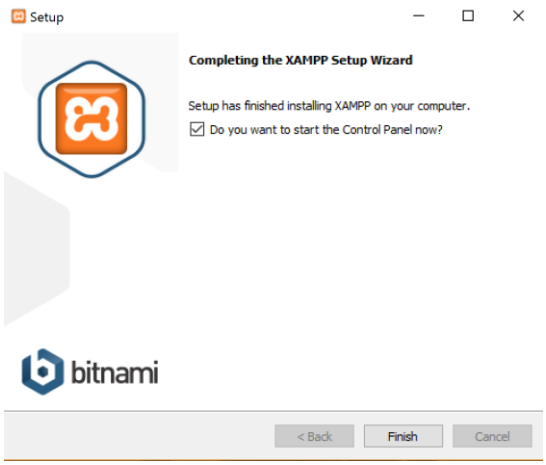
Click next to install




Allow any pop ups such as Apache or MySql

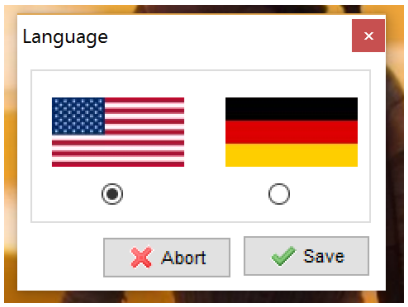


Then click finish



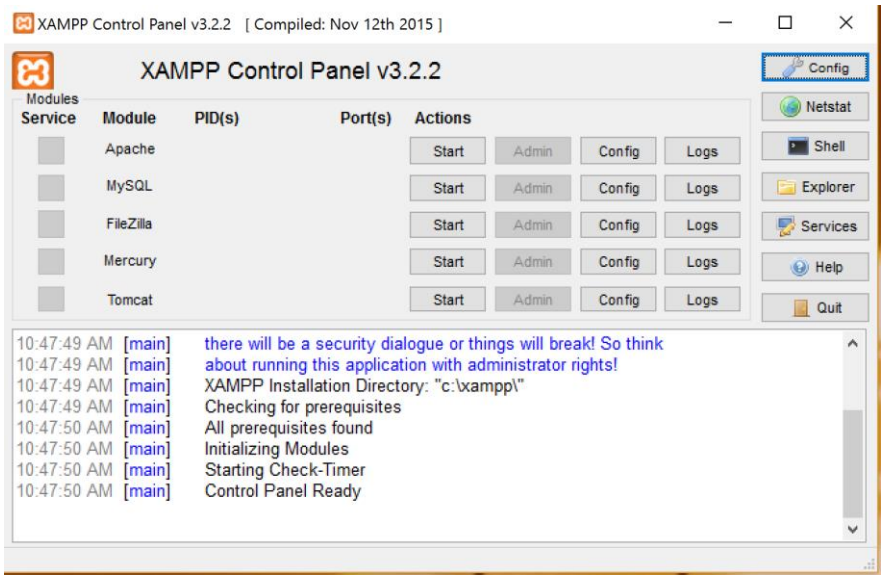

Now we configure Xampp.

Assuming you left the “Do you want to start the Control Panel” ticked, you should see the following on your screen.



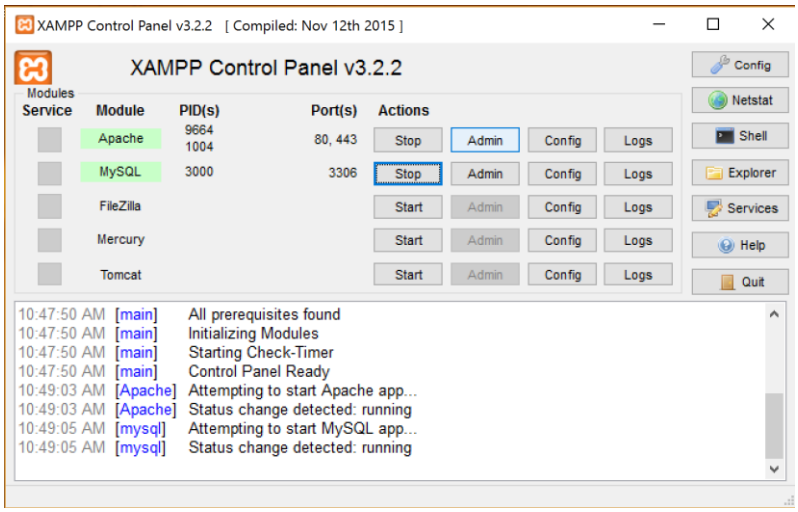

Click on Save

The following should appear:




From here we need to click Start on Apache and MySql.

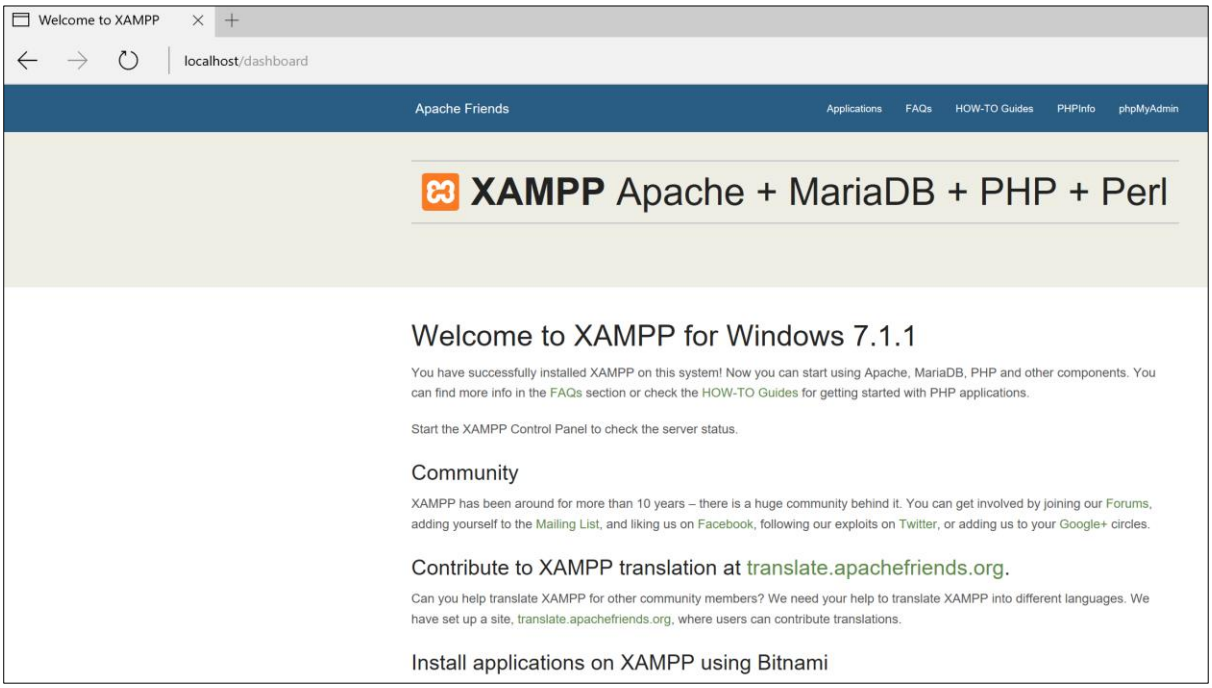
You should see the following.




We now have the Apache webserver up and running with MySql.

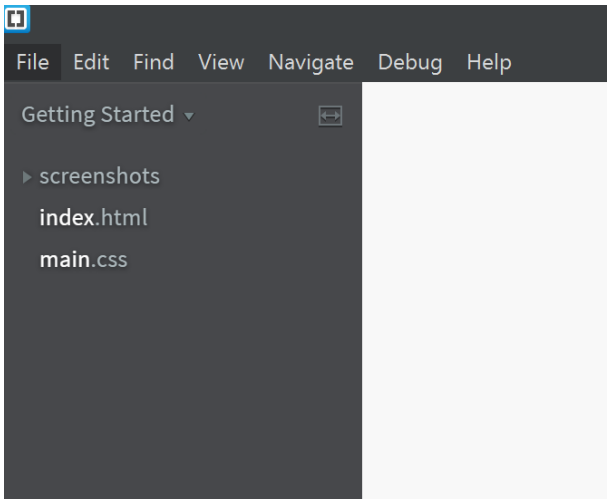
To check that it is working load up a browser and type localhost into the address bar.

You should see the following:

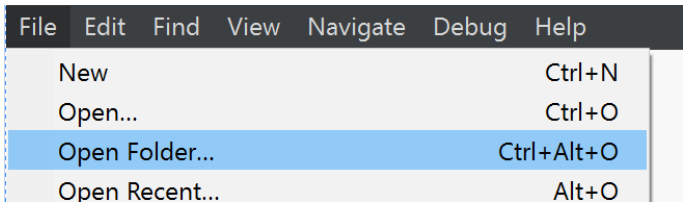



Now, this is all configuration information from Xampp, what we should do is clean out the folder htdocs, to put our own files. To do this, we will use brackets to clean out the folder.

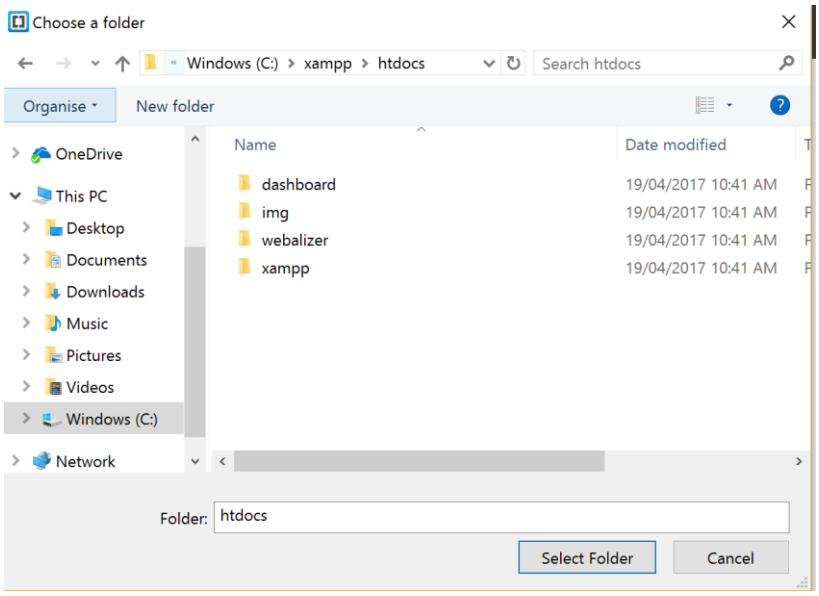
Load up brackets




From here, go to open folder

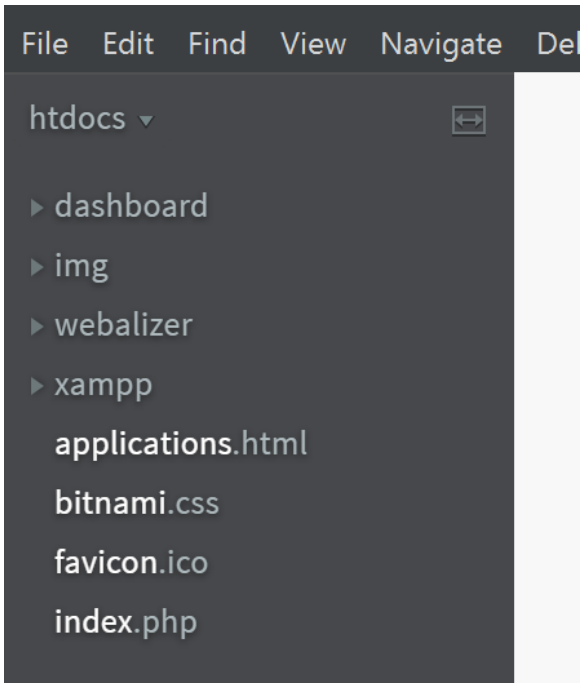



Navigate to c:\xampp\htdocs

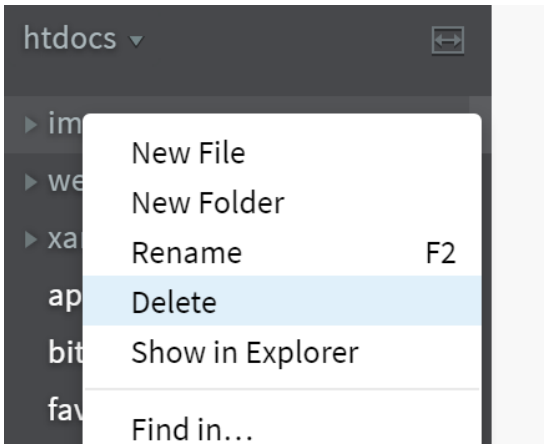



Click Select

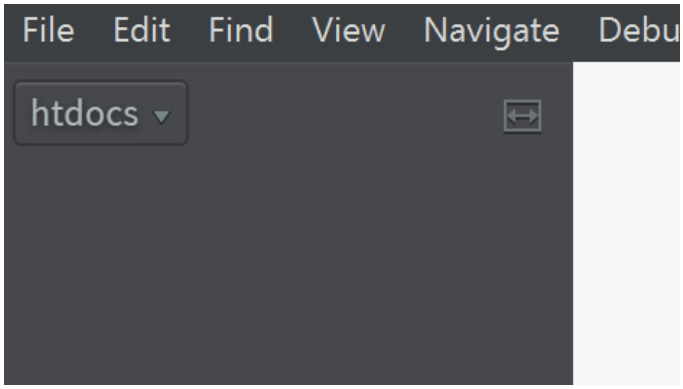
This should give you the following set of files in the navigation pane of brackets:



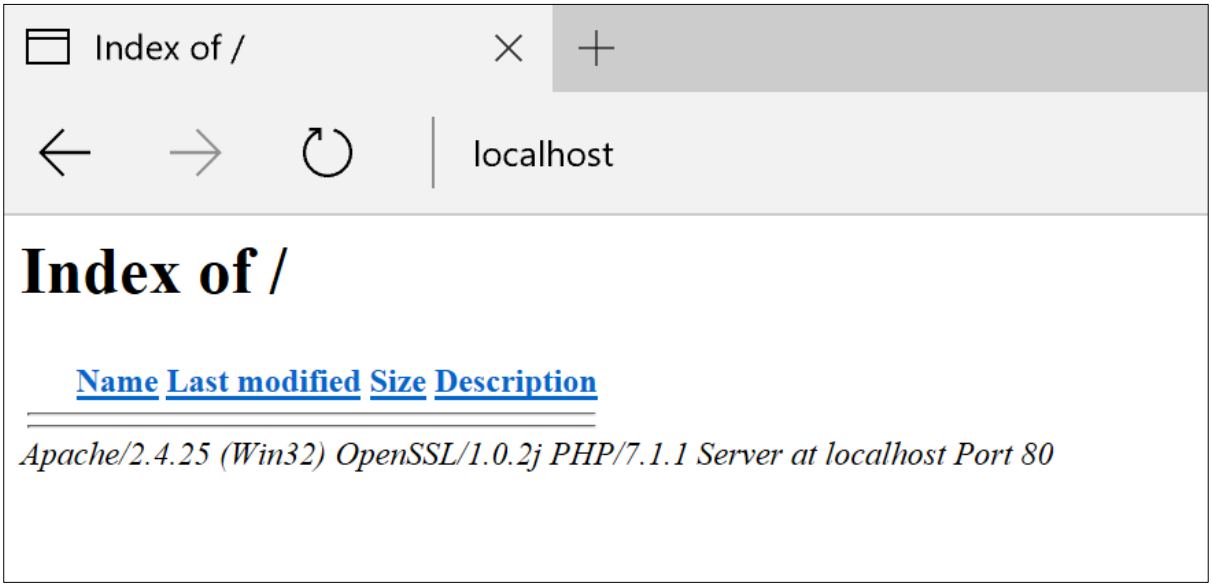

Highlight and right click on the navigation elements and delete everything in the folder




Once you have emptied the folder it should look like



Now, we check to ensure that the system is clean, go to your browser and navigate to http://localhost/ You should see the following




Now we have a clean server to start creating with.

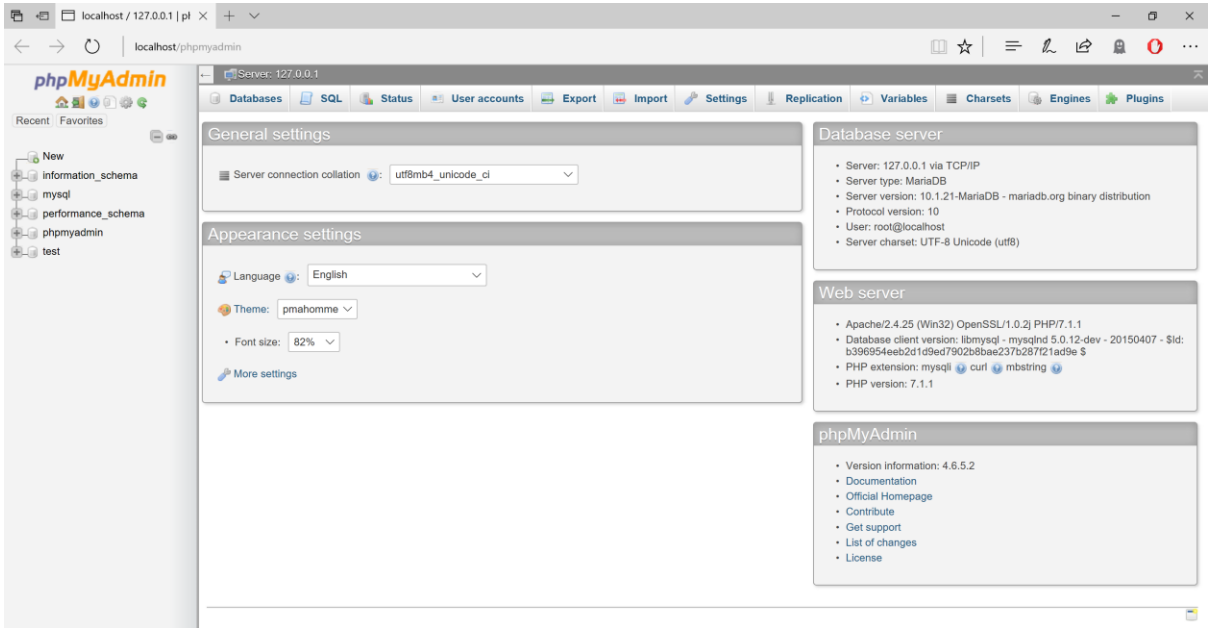
## Creating your php Website

From here, right click and create a new file called index.php and put in the following content:


In this lesson we will be communicating with a database through our php files. There are 4 aspects to database communication; Create, Read, Update and Delete otherwise referred to as CRUD. But before we get into the code, let's look at the structure of a database and how we can access and manipulate the database service.

There are a couple of ways in which we can do this, one is via command prompt and the other is through a web interface. In our case, to make life as easy as possible, we will be using the web interface; this interface is called phpmyadmin.

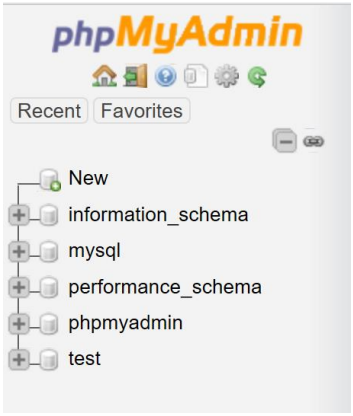
To get access to the web interface, open a browser and go to the address localhost/phpmyadmin and you should see the following:




This is the normal screen in which you can manipulate all of the databases you require for most web projects.

We will run through the interface, to increase your ability to navigate and understand the way it works and how it will work for your code.

- The following pane, contains a list of all currently existing databases within the MySql system.

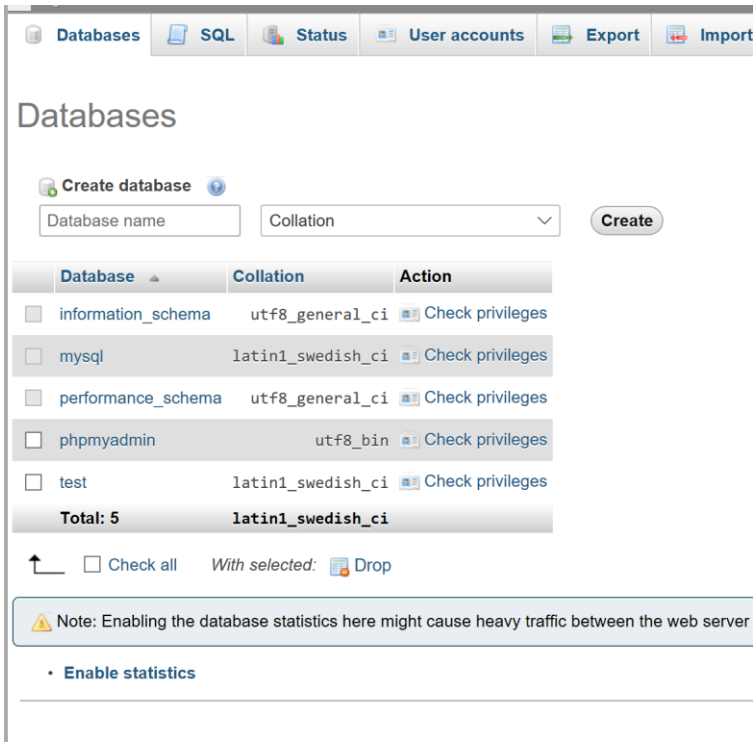



-
- Each of the databases can be viewed in more by expanding via the plus sign. Also note, that there is a home (house) icon under the title, this returns you to this page.
- This part of the interface:


-
- Allows you to access specific elements of MySQL. There are only a few of them that we will require for this lesson.
    - Database – This is where you can create and delete databases
    - User Accounts – This is where you will create/delete user accounts that allow access to the database. This is a security element that is very important, if your server has multiple databases and each database is access via one user account, if there is a security breach then only one database can be compromised. So always create a specific user for a database.
    - Export – This is where you can make a backup of the database. When saving your work/submitting your assignment. You will need to export your database to a .sql file.
    - Import – This is where you bring a .sql file back into the MySQL system. So, if you have worked on a database, exported the work and saved it to USB, if the computer removes all the server details (All university computers are cleaned, so your files will be removed each week) it means that when you return to the computer, you will be able to import all the database information (Tables and data). Thereby removing the potential requirement to recreate and re-enter.

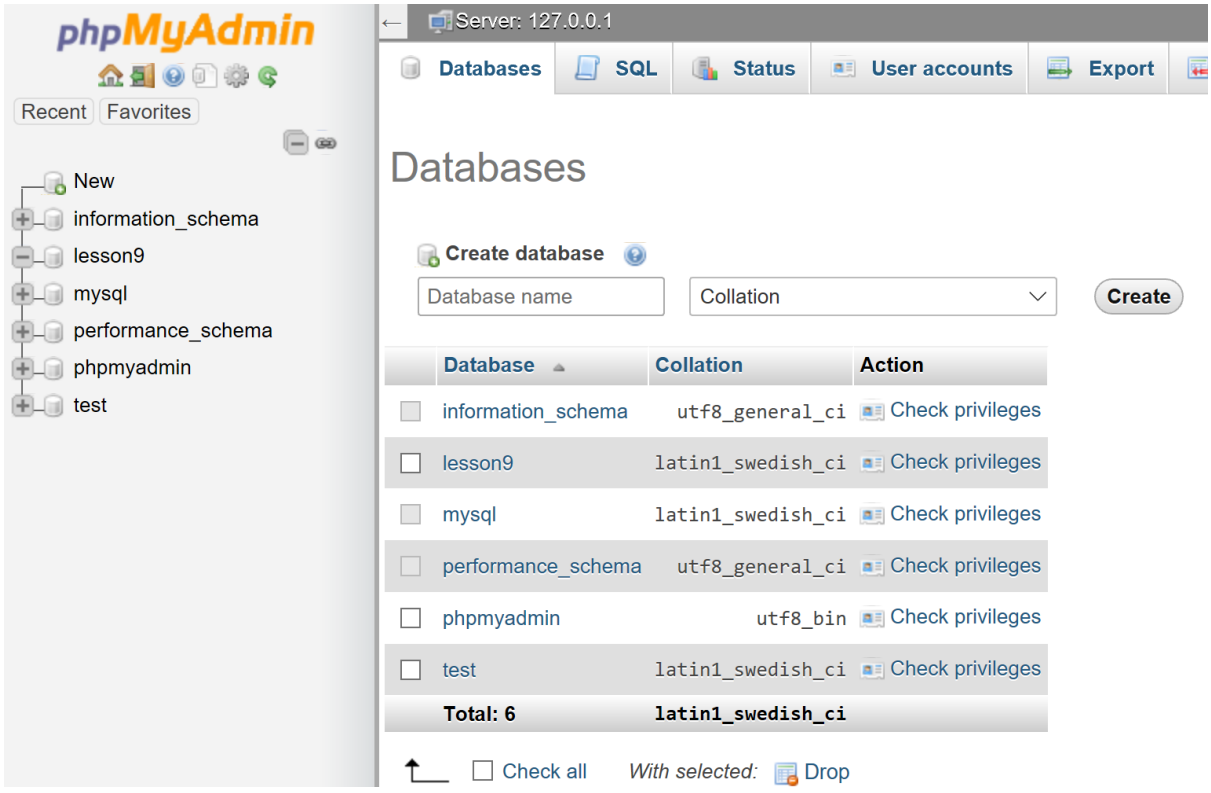
Now that we have a basic overview of the interface, let's create the database structure, account, and tables within the database

Click on the database link, you should see the following:

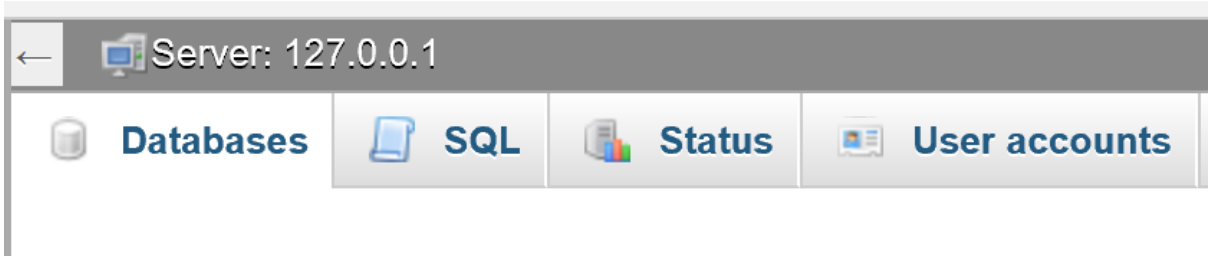



From here enter in the database name and click create. NB When it comes to your assignments, try using your student number as your database name. In this case, we will name the database lesson9.

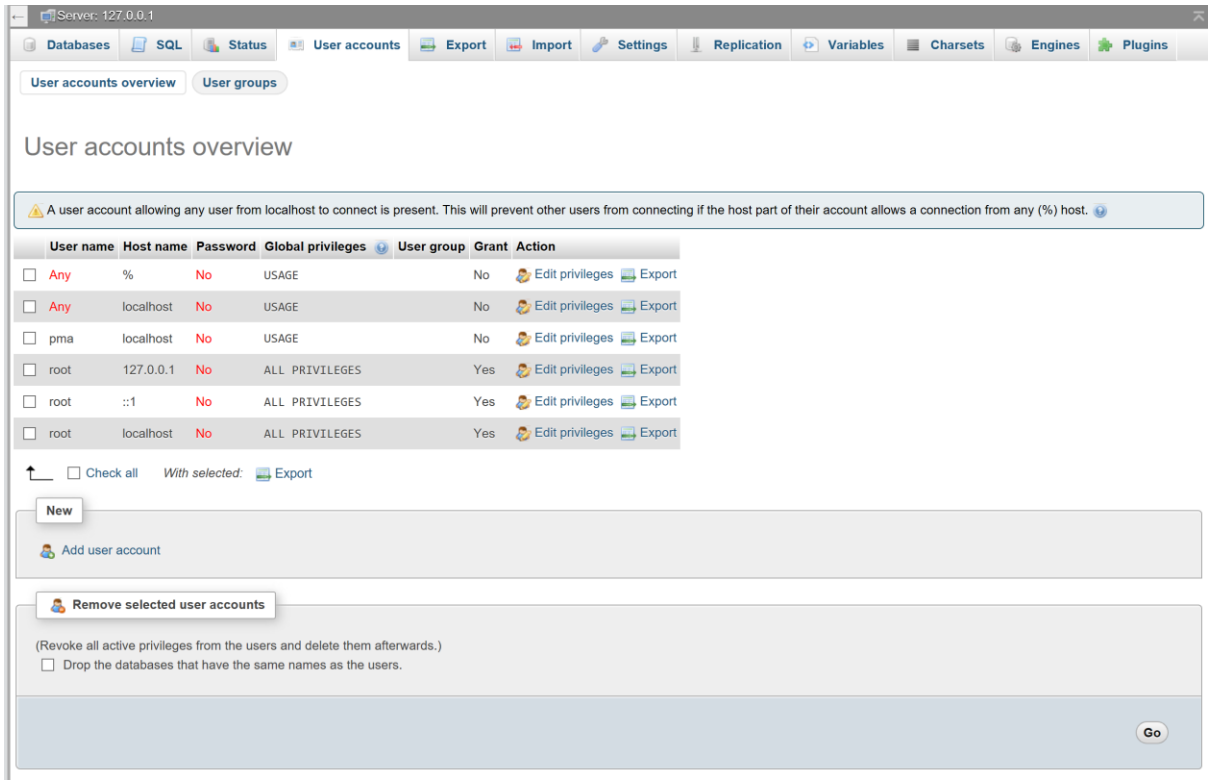
You should have a structure looking like this once done. Once the database was created, you might be automatically taken into the database, click on the home icon and then database link to see the following.



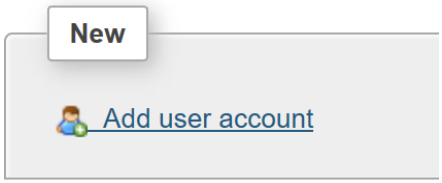
As you can see, the database is listed in the left navigation bar as well as in the list of databases. Next click on the user accounts link.




You should see the following:




This is a list of all current user accounts within the MySQL system. To ensure that we work securely with our database, we need to create a new account.  As such, click on Add User Account:

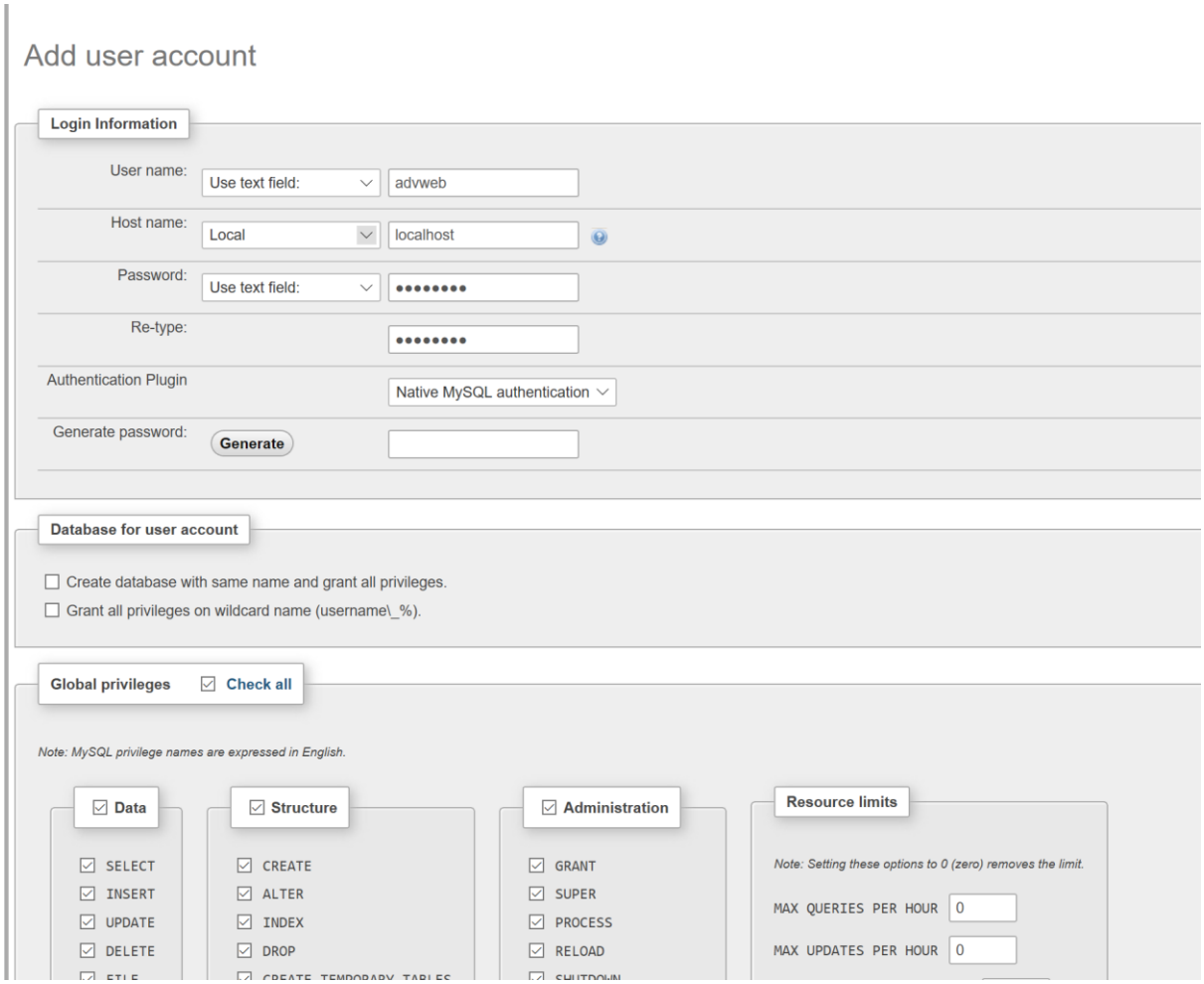



Here you will be prompted for details to enter. User the following:

- Username: advweb
- Host: localhost
- Password: password

Also, ensure that you click the check all part. Once the details a re filled out scroll down and click go. This will create the user and apply it's permissions.

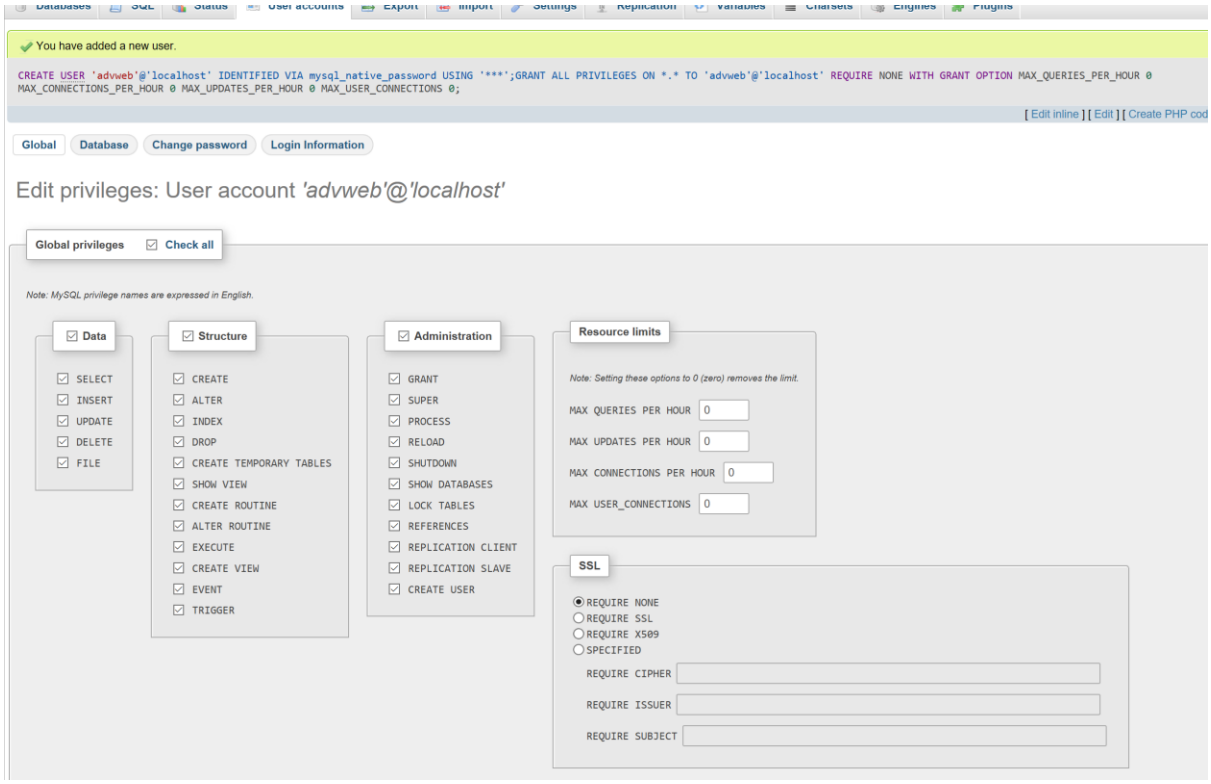
It should look like the following:



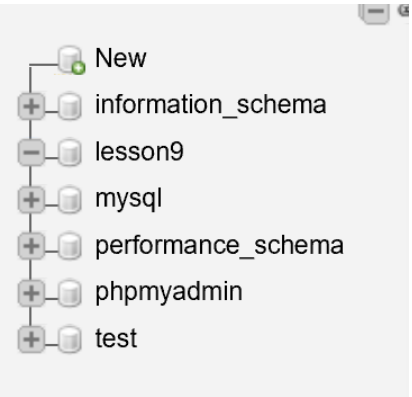

Once you have clicked go, you should be redirected to a page that indicates a successful user addition. If there is an error message that appears regards c:\xampp\mysql\lib\plugins and the message states dir not found. You will have to create the folder structure. Open file explorer and navigate to c:\xampp. Once there, go inside the mysql folder and create a new folder called lib. Once this has been done, go inside the newly created lib folder and create the folder called plugins.

After the creation of these folders, go back to the web browser and re-create the user. It should work successfully this time.

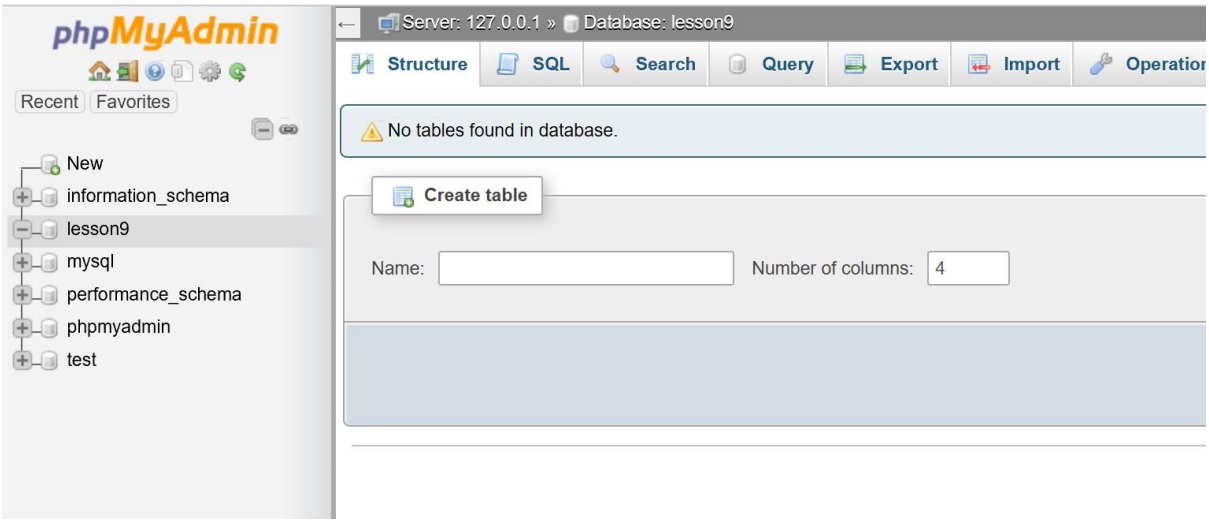
You should be directed to the following page:




Now that the user and database have been created, we need to add some tables to the database, to do this, we need to access the database. On the left hand navigation panel, click on the database called lesson9.




This will bring you to the following screen:




Now that we are inside the database we need to create a table to hold information.

A table can be thought of as a spreadsheet, with each column containing a different type of information. Overall, the table should relate all the containing data, for example; if we made a table called users, we would expect column such as first name, last name, email, phone, mobile, address and so forth. What you wouldn't expect is to find items such as grades, classes, marks and such. So, when designing tables, keep the information related.

On tables, it is best to create a column called a primary key that is auto-incrementing. This means that each time you add information to that table, the system will increase the primary key. This is often referred to as an id field. Your student number is your id within the universities computer network and it is used to link all relevant information about you.

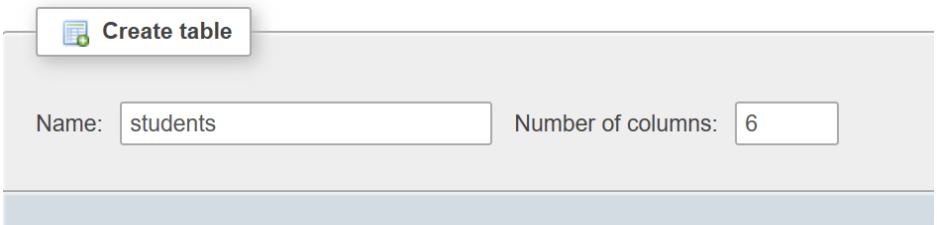
What we will do is create a database that holds students and classes, in this manner you will learn to see how we can keep information separate and yet link them through a database.

To start with, we will create the user table. This table will be designed to hold relevant information in regards to a student.

The table will consist of the following columns

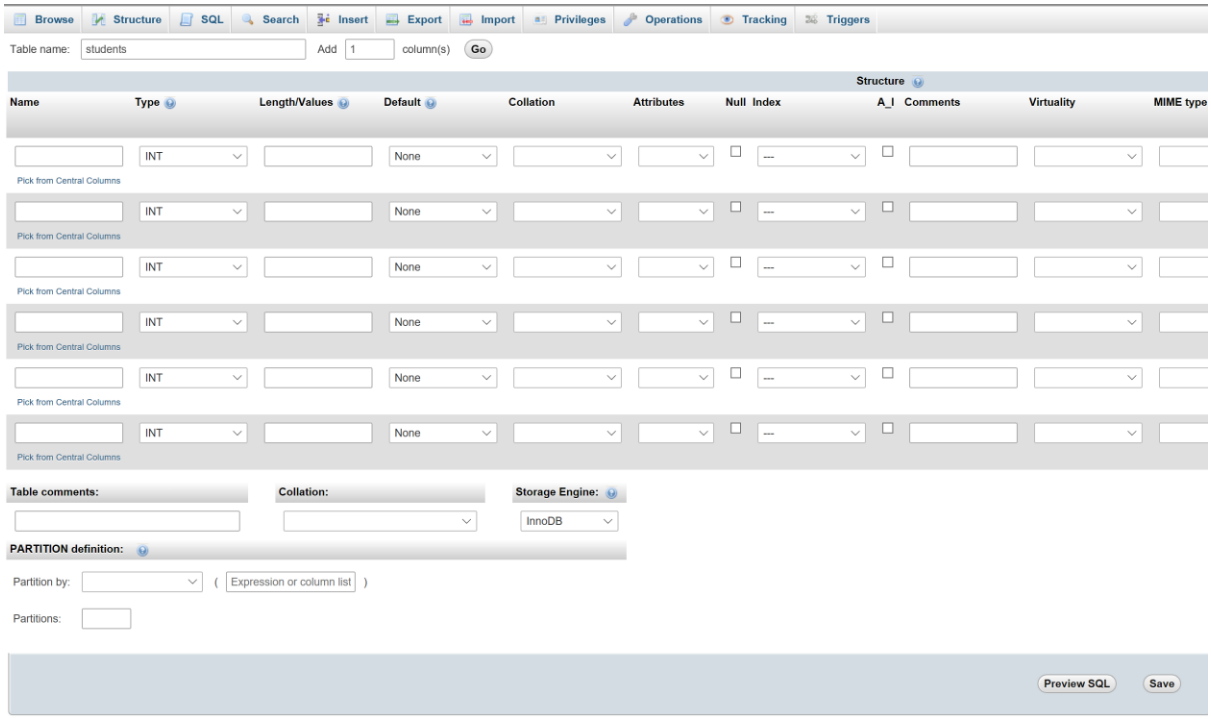
- Id
- First name
- Last name
- Username
- Password
- Email

So, now we need to create the table.




Once entered, click go

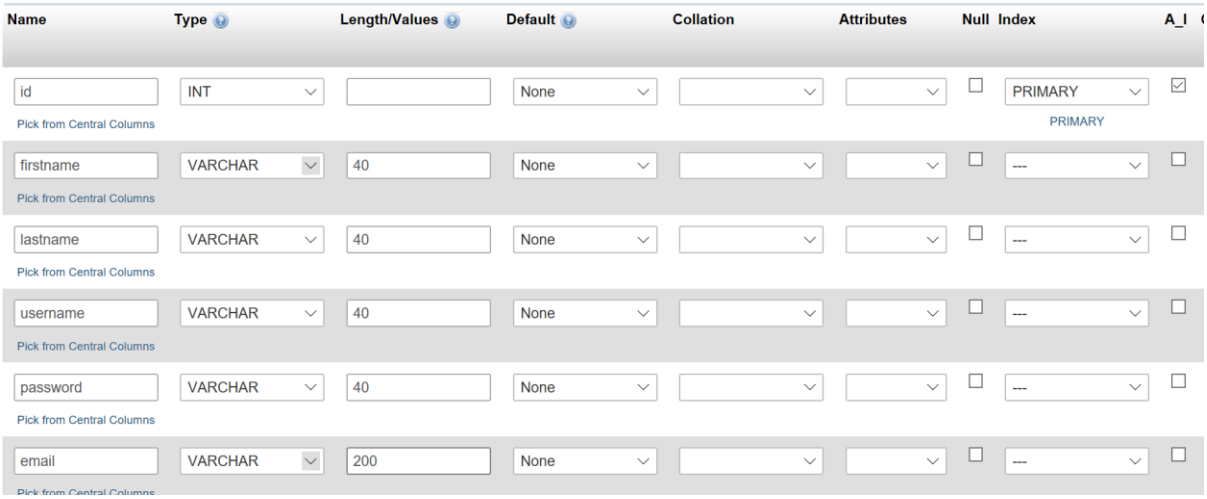
You should see the following screen:



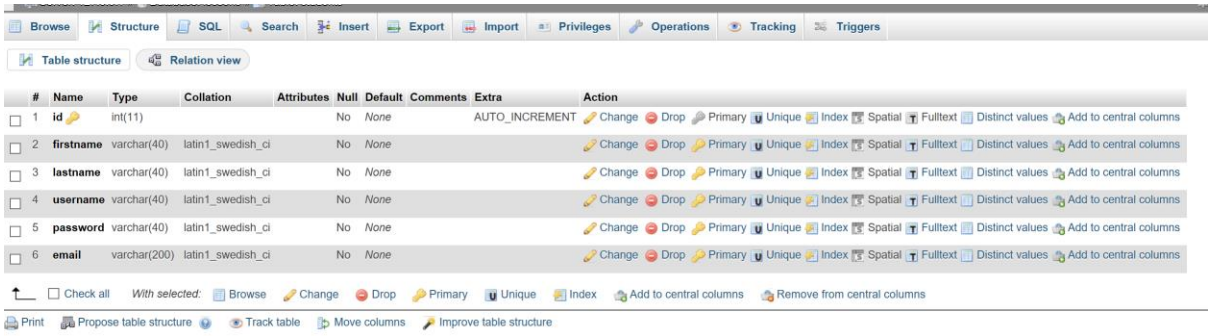

This screen is where you create each column of information, it looks daunting, but we only need to modify a few aspects of it, these are:

- Name – this is the name of the column, so it will be elements like id, firstname, lastname and so forth
- Type – this is where you state what type of element is being stored, the main ones to be concerned with are:
    - INT – Whole numbers, ie 1,2,345,678
    - VARCHAR – This is any typed in information, so all alphanumeric and symbol details, the difference between this and text is that with varchar you state how many characters you wish to store; ie varchar of length 4 could store aaaa 4 characters but it is unable to store aaaaa as this is 5 characters in length
    - TEXT – This allows you to store large amounts of text, no need to indicate a length on characters
    - DATE – Allows you to store a date into the system
    - BOOLEAN – This is a true/false scenario, so the data is one character of either 1 or 0
    - BLOB – Binary Objects, this is how you can store elements such as images, pdfs or other items of digital format that are not straight strings.
- Length – This is where you can state how many characters of a specific type is used. For example int(6) would indicate 6 characters, so the range of numbers is 000000 to 999999
- Index – This is for determining primary keys, so normally only selected for the id field
- A_I – This is shorthand for auto increment. This is normally selected for the primary key and allows the system to increase the primary key on each new addition to the table.

Now we fill out the fields for the student table; it should look like this:



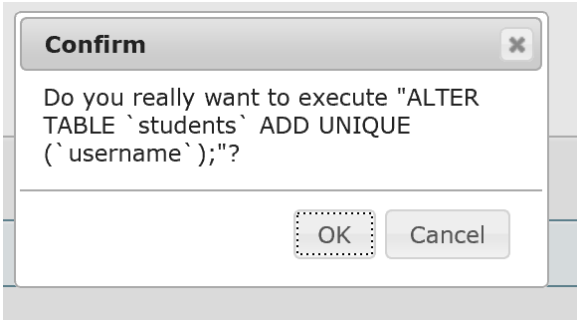

Once the fields have been filled out, click on save. This will take you to the following page:



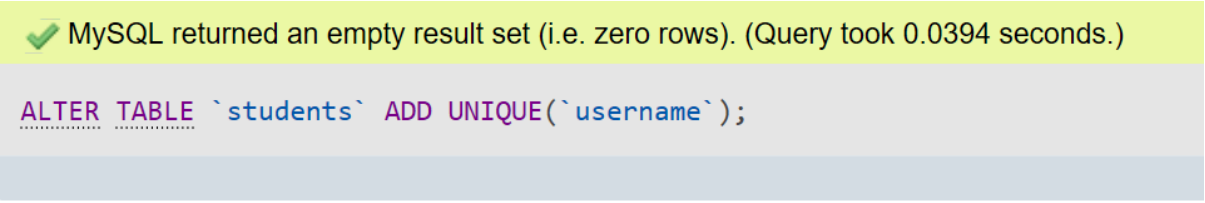

This is the structure page. This page is designed to give you information about the elements of the database table. Now, the table is almost complete, there is one thing left to do, and this is to make the username unique. After all, we don't want anyone to access other peoples details so we eliminate the abilty to have the same username.

On the username row, you will see


Click on unique, this will pop up

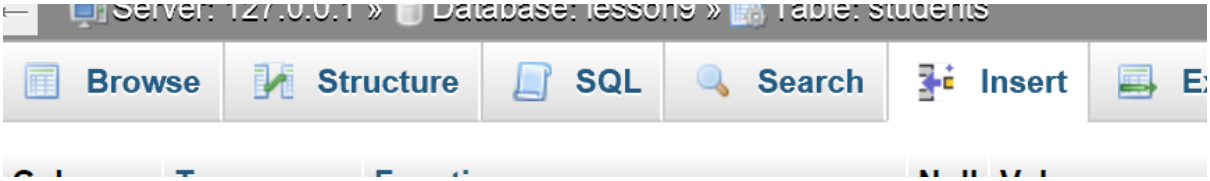


Click on ok. You should this this:




This indicates that it is complete.

There are a couple of ways to now put information into the database, we can do it manually, by going through the phpyadmin, this uses the insert tab:

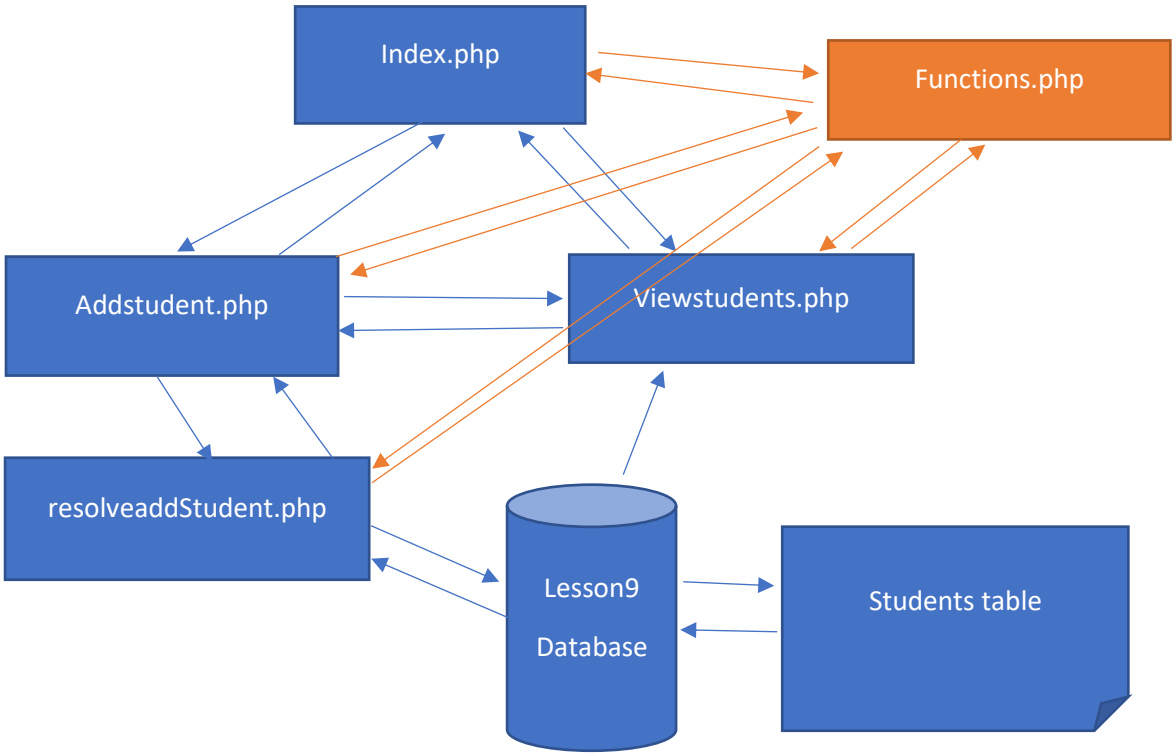



But, this is not how we want to do it. As such, we will now create php pages to add information into the database and read the data from the database.

So, let's load up brackets and start writing up these web files.

## Brackets PHP – MySQL

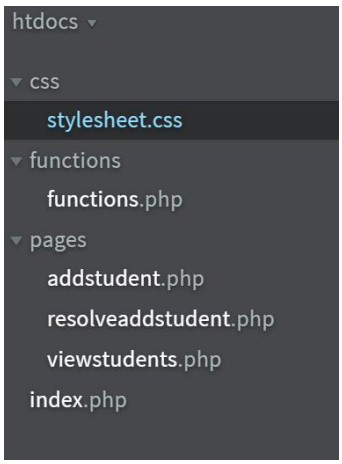
To start with, we will first think about and plan the navigation of our website. From a hierarchical structure the site, for create and read, is very simple, it looks like this:



As you can see, each page is linked, we will create and store functions in the functions.php file for access. Navigation will be based around a very simple linking method.

Now, we will create the php files with a working navigation. Index.php will mainly be used for navigation, so the design is very simple. But, as we want to make coding very easy, we will turn the navigation into a php function.

Use the following file structure:




Index.php

```php
<?php
    include_once('functions/functions.php');
?>
<!DOCTYPE html>
<html>
    <head>
        <title></title>
        <style type="text/css">
        </style>
        <link href="css/stylesheet.css" rel="stylesheet" type="text/css">
    </head>
    <body>
        <div id="navigation">
        <?php
            nav('index');
        ?>
        </div>
    </body>
</html>
```

Stylesheet.css

```css
#navigation{width: 90%; background-color: gray;padding-top: 10px; padding-bottom: 10px;}
#navigation a{font-size: 1.2em; text-decoration: none; margin-left: 50px;color: white;}
#navigation a:hover{text-decoration: underline;}
```

Functions.php

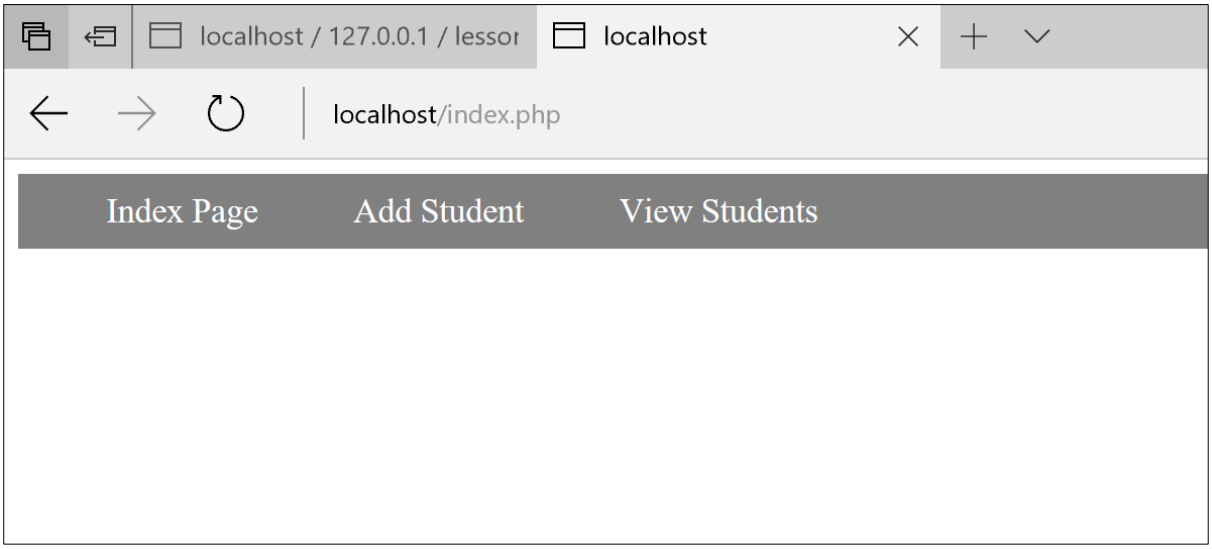
```php
<?php

// Navigation function
function nav($page)
{
    if ($page == 'index')
    {
        echo '
            <a href="index.php">Index Page</a>
            <a href="pages/addstudent.php">Add Student</a>
            <a href="pages/viewstudents.php">View Students</a>
        ';

    }else
    {
        echo '
            <a href="../index.php">Index Page</a>
            <a href="addstudent.php">Add Student</a>
            <a href="viewstudents.php">View Students</a>
        ';
    }
}

?>
```

Result (Remember this is all done on localhost)



Now that we know it works, duplicate the layout for the other pages

Addstudent.php

```php
<?php
    include_once('../functions/functions.php');
?>
<!DOCTYPE html>
<html>
    <head>
        <title></title>
        <style type="text/css">
        </style>
        <link href="../css/stylesheet.css" rel="stylesheet" type="text/css">
    </head>
    <body>
        <div id="navigation">
        <?php
            nav('addstudent');
        ?>
        </div>
    </body>
</html>
```

Viewstudents.php

```php
<?php
    include_once('../functions/functions.php');
?>
<!DOCTYPE html>
<html>
    <head>
        <title></title>
        <style type="text/css">
        </style>
        <link href="../css/stylesheet.css" rel="stylesheet" type="text/css">
    </head>
    <body>
        <div id="navigation">
        <?php
            nav('viewstudents');
        ?>
        </div>
    </body>
</html>
```

Resolveaddstudents.php

```php
<?php
    include_once('../functions/functions.php');
?>
<!DOCTYPE html>
<html>
    <head>
        <title></title>
        <style type="text/css">
        </style>
        <link href="../css/stylesheet.css" rel="stylesheet" type="text/css">
    </head>
    <body>
        <div id="navigation">
        <?php
            nav('resolveadd');
        ?>
        </div>
    </body>
</html>
```

Once this is done, save all the pages and test the navigation. You should be able to bounce between each page using the new navigation page. Next we need to look at adding students to the student table.

To connect to the database, we need to create a function that will allow us to communicate, as this communication needs to work on all pages, we create a function and link it to each page a lot like we did with the navigation.

So, we will create a new function called dbLink() and link it to each page. But first, we need to write it and test it, so we will work on the index.php page first.

Functions.php

```php
//database connection function using PDO
function dbLink()
{
    $db_user = "advweb";
    $db_pass = "password";
    $db_host = "localhost";
    $db = "lesson9";
    try{
        $db = new PDO("mysql:host=$db_host;dbname=$db",$db_user,$db_pass);
    } catch (Exception $e){
        echo 'Unable to access database';
        exit;
    }
    error_reporting(0);
    return $db;
}
```

Notice this is where the username/database name and password that you created through phpmyadmin comes into play. If you look at line 38 'return $db;' this is where we return a true or false to the success of the function talking to the database. This is used in index.php to let us know that our webpages and database are talking to each other.

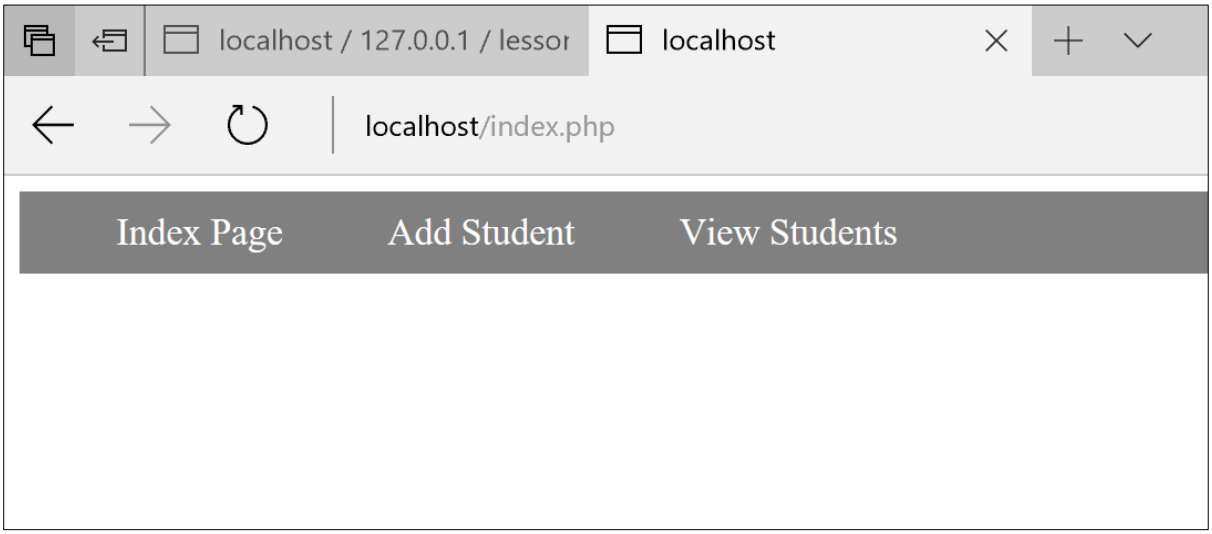
Index.php

```php
<?php
    include_once('functions/functions.php');
    $dbConnect = dbLink();
    if ($dbConnect)
        echo '<!-- database connection established -->';
?>
<!DOCTYPE html>
<html>
    <head>
        <title></title>
        <style type="text/css">
        </style>
        <link href="css/stylesheet.css" rel="stylesheet" type="text/css">
    </head>
    <body>
        <div id="navigation">
        <?php
            nav('index');
        ?>
        </div>
    </body>
</html>
```

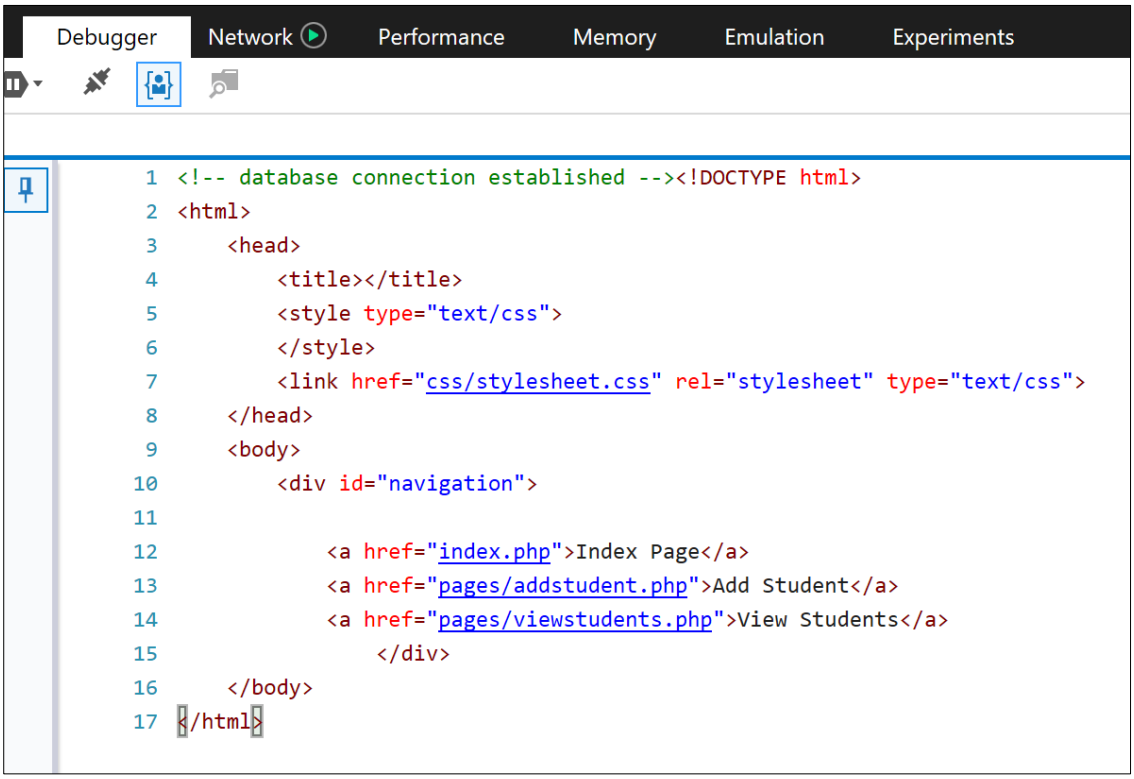
As you can see, just below the line that links index.php to functions.php we call the function dbLink(), but the connection result is feed into a variable called $result. The next if statement writes a html comment to the page to let us know if it worked.

So, to check, we reload the index page and then view the source

Result




Result source code




As you can see, the html comment is written into the source code, so this indicates we a re successfully talking to the database.

We need to do this on the other pages. Add the following code to the top of each page

Addstudent.php

```php
<?php
    include_once('../functions/functions.php');
    $result = dbLink();
    if ($result)
        echo '<!-- database connection established -->';
?>
<!DOCTYPE html>
<html>
```

Resolveaddstudent.php

```php
<?php
    include_once('../functions/functions.php');
    $result = dbLink();
    if ($result)
        echo '<!-- database connection established -->';
?>
<!DOCTYPE html>
<html>
```

Viewstudents.php

```php
<?php
    include_once('../functions/functions.php');
    $result = dbLink();
    if ($result)
        echo '<!-- database connection established -->';
?>
<!DOCTYPE html>
<html>
```

Once done, save all pages and then click through the navigation; on each page, do a view source and confirm that the comment is there.

Next we need to add the data to the database, so to do this we will work on the addstudents.php page. This is just a simple form to collect information. Recall from the database table the following fields:

- Id
- First name
- Last name
- Username

- Password
- Email

Each element we will ask for, except for the id. This will be calculated by the MySQL system as it is an auto-incrementing field.

To collect information, we will need a simple html form:

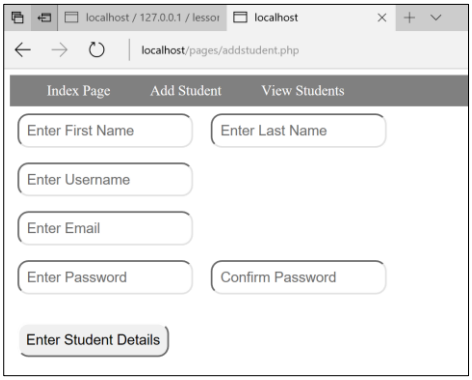
Addstudent.php

```php
<?php
    include_once('../functions/functions.php');
    $dbConnect = dbLink();
    if ($dbConnect)
        echo '<!-- database connection established -->';
?>
<!DOCTYPE html>
<html>
    <head>
        <title></title>
        <style type="text/css">
        </style>
        <link href="../css/stylesheet.css" rel="stylesheet" type="text/css">
    </head>
    <body>
        <div id="navigation">
        <?php
            nav('addstudent');
        ?>
        </div>
        <div class="add-form">
            <form action="resolveaddstudent.php" method="post">
                <input type="text" name="fname" placeholder="Enter First Name">
                <input type="text" name="lname" placeholder="Enter Last Name"><br>
                <input type="text" name="uname" placeholder="Enter Username"><br>
                <input type="text" name="email" placeholder="Enter Email"><br>
                <input type="password" name="pwd" placeholder="Enter Password">
                <input type="password" name="pwd2" placeholder="Confirm Password"><br><br>
                <input type="submit" name="submit" value="Enter Student Details"><br>
            </form>
        </div>
    </body>
</html>
```

Stylesheet.css

```css
#navigation{width: 90%; background-color: gray;padding-top: 10px; padding-bottom: 10px;}
#navigation a{font-size: 1.2em; text-decoration: none; margin-left: 50px;color: white;}
#navigation a:hover{text-decoration: underline;}
input{padding: 10px 10px 10px 10px; font-size: 1.2em; border-radius: 15px;}
.add-form input{margin: 10px 10px 10px 10px;}
```

Result



Now we need to modify the addresolvestudent.php page so that we can test to ensure that data is travelling from addstudent to resolveaddstudent pages.

Functions.php

```php
//Show the information in memory
function showMem()
{
    echo '<pre>';
    echo '<h3>Post Memory</h3>';
    print_r($_POST);
    echo '<h3>Get Memory</h3>';
    print_r($_GET);
    echo '<h3>Session Memory</h3>';
    print_r($_SESSION);
    echo '</pre>';
}
```

Resolveaddstudent.php

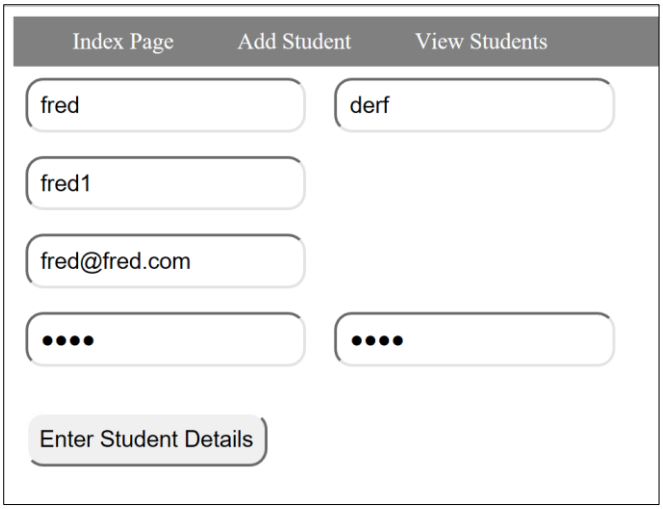
```php
<?php
    include_once('../functions/functions.php');
    $dbConnect = dbLink();
    if ($dbConnect)
        echo '<!-- database connection established -->';
?>
<!DOCTYPE html>
<html>
    <head>
        <title></title>
        <style type="text/css">
        </style>
        <link href="../css/stylesheet.css" rel="stylesheet" type="text/css">
    </head>
    <body>
        <div id="navigation">
        <?php
            nav('resolveadd');
        ?>
        </div>
        <?php
            showMem();

        ?>
    </body>
</html>
```

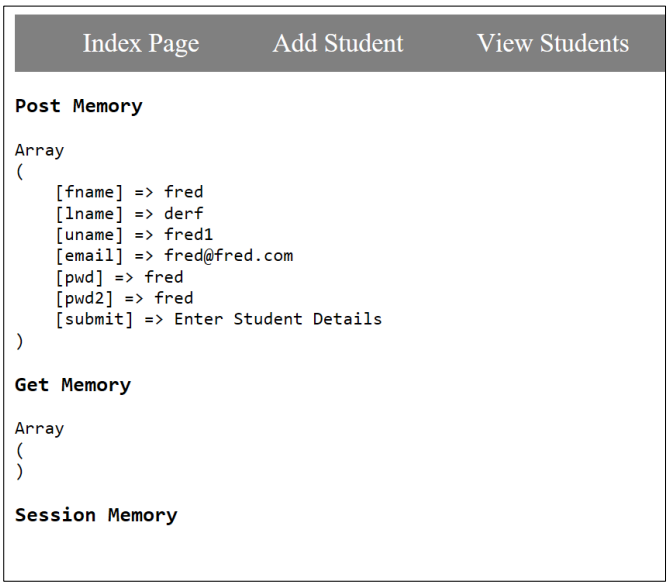
Now that this is ready to be tested. Save the pages and test.

Result – addstudent.php

Enter in any details you want.




Result resolveaddstudent.php




As you can see, the information has transferred over and is now ready to be collected into variables and then inserted into the database. But, before we insert the data into the database we need to check that the passwords match, otherwise, the student might have issues logging in.

To do this, we require a simple validation on the passwords. So we add a new function and implement it on the resolveaddstudent.php page.

Functions.php

```
54  //Validate passwords
55  function validate($pwd1,$pwd2)
56  {
57      if ($pwd1 == $pwd2)
58          return true;
59      else
60          return false;
61
62  }
```

Resolveaddstudent.php

```
15      <body>
16          <div id="navigation">
17          <?php
18              nav('resolveadd');
19          ?>
20          </div>
21          <?php
22              showMem();
23              //Collect the information
24              $fname = $_POST['fname'];
25              $lname = $_POST['lname'];
26              $uname = $_POST['uname'];
27              $email = $_POST['email'];
28              $pwd = $_POST['pwd'];
29              $pwd2 = $_POST['pwd2'];
30              $validate = validate($pwd,$pwd2);
31              if($validate)
32              {
33                  //Store all the information in the database
34              }else
35              {
36                  echo'The passwords did not match.<br>';
37                  echo 'Please <a href="addstudent.php">Try Again</a>!';
38              }
39          ?>
40      </body>
```

Result (Same passwords)

Index Page Add Student View Students

**Post Memory**

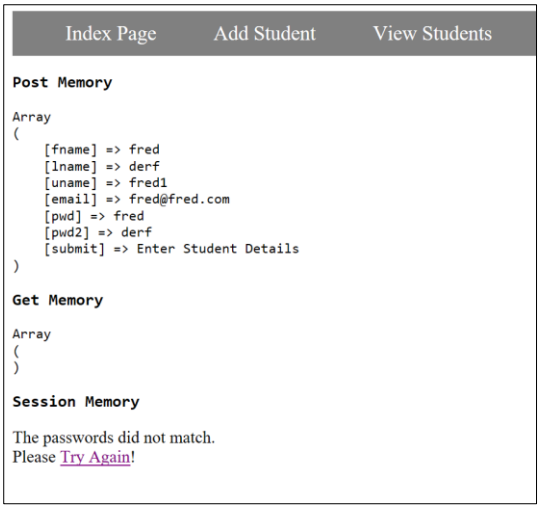
```
Array
(
    [fname] => fred
    [lname] => derf
    [uname] => fred1
    [email] => fred@fred.com
    [pwd] => fred
    [pwd2] => fred
    [submit] => Enter Student Details
)
```

**Get Memory**

```
Array
(
)
```

**Session Memory**

Result (Different passwords)




Now that we know our validation works, we can move onto storing the information into the database. We've already collected all the post information into variables, so now, we transfer this information into a function we will create for insertion.

Functions.php

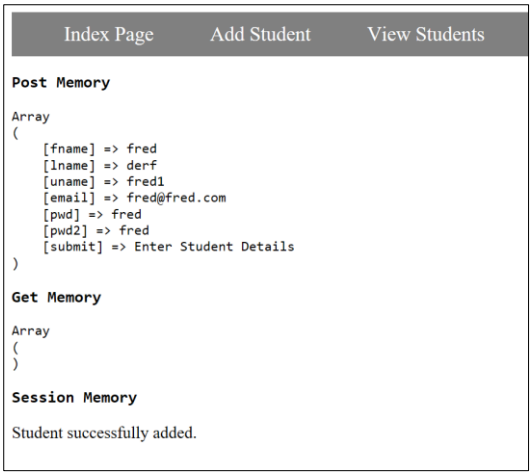
```php
//Insert Student
function insertUser($dbConnect,$firstname,$lastname,$username,$password,$email)
 {
    $q = "INSERT into students (id,firstname,lastname,username,password,email)
    VALUES(NULL,:fname,:lname,:uname,:password,:email);";
    $query = $dbConnect->prepare($q);
    $query->bindParam(":fname",$firstname);
    $query->bindParam(":lname",$lastname);
    $query->bindParam(":uname",$username);
    $query->bindParam(":password",$password);
    $query->bindParam(":email",$email);
    $result = $query->execute();
    return $result;
 } // eo of function
```

Resolveaddstudents.php

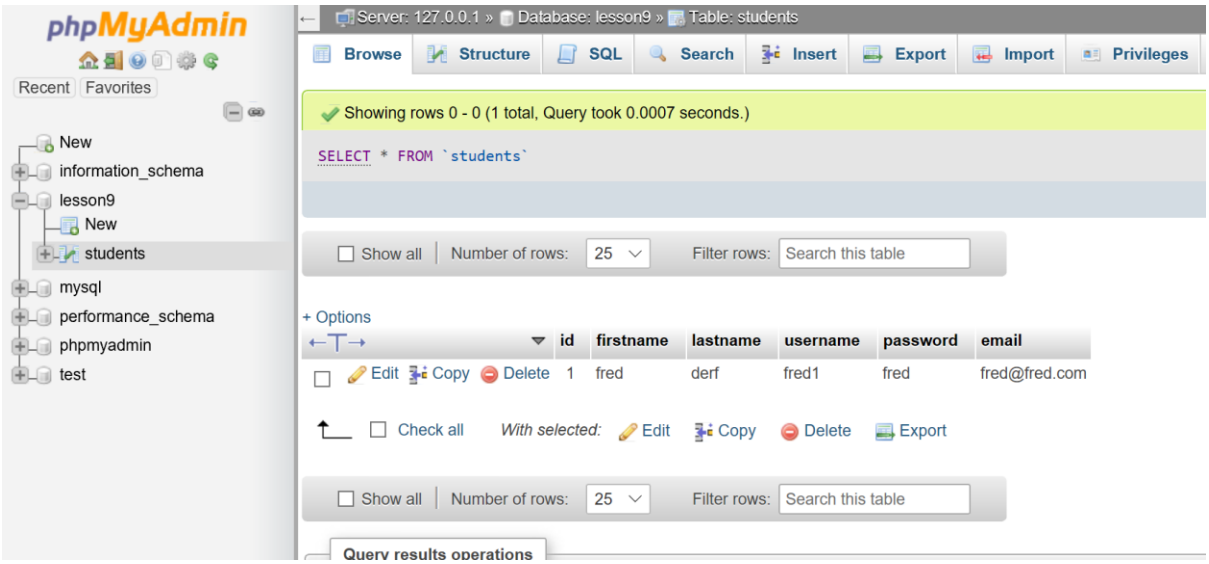
```php
<?php
    showMem();
    //Collect the information
    $fname = $_POST['fname'];
    $lname = $_POST['lname'];
    $uname = $_POST['uname'];
    $email = $_POST['email'];
    $pwd = $_POST['pwd'];
    $pwd2 = $_POST['pwd2'];
    $validate = validate($pwd,$pwd2);
    if($validate)
    {
        //Store all the information in the database
        $resultInsert = insertUser($dbConnect,$fname,$lname,$uname,$pwd,$email);
        if($resultInsert)
        {
            echo 'Student successfully added.';
        }
    }else
    {
        echo'The passwords did not match.<br>';
        echo 'Please <a href="addstudent.php">Try Again</a>!';
    }
?>
```

Run through the addstudent process again, you should see the following

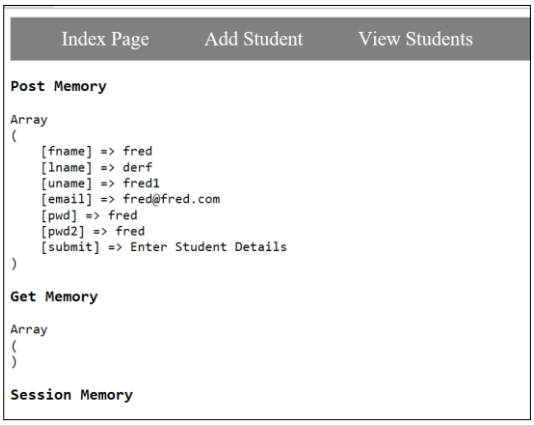
Result (resolveaddstudents.php)




Result (phpmyadmin)



Now, that we have added a single student, let's see if we can add the same one again. Going through addstudent with the same information results in:




As you can see, this is indicative of the password details matching but, if you recall the if statement:

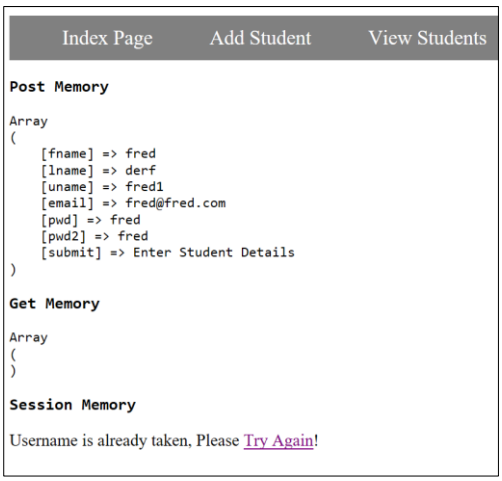
```
$resultInsert = insertUser($dbConnect,$fname,$lname,$uname,$pwd,$email);
if($resultInsert)
{
    echo 'Student successfully added.';
}
```

We didn't say what will occur if there is another issue. Pretty much the only other issue is that the username is already taken, hence why the insert didn't return a successful value. So, let's add an else to that particular if statement and test again.

Resolveaddstudent

```
if($resultInsert)
{
    echo 'Student successfully added.';
} else
{
    echo 'Username is already taken, Please <a href="addstudent.php">Try Again</a>!';
}
```

Result

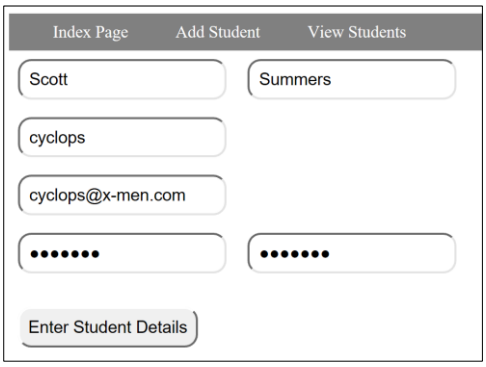



The only thing left for the resovleaddstudent page is to clean up the showMem() function. On that page, comment out the function. And then test by adding a new user.

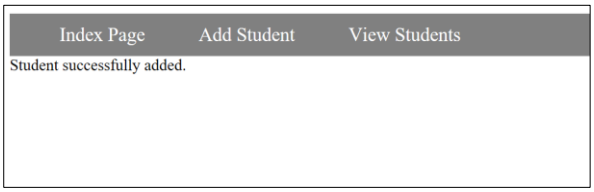
Resolveaddstudent

```php
<?php
   //| showMem();
    //Collect the information
    $fname = $_POST['fname'];
    $lname = $_POST['lname'];
```

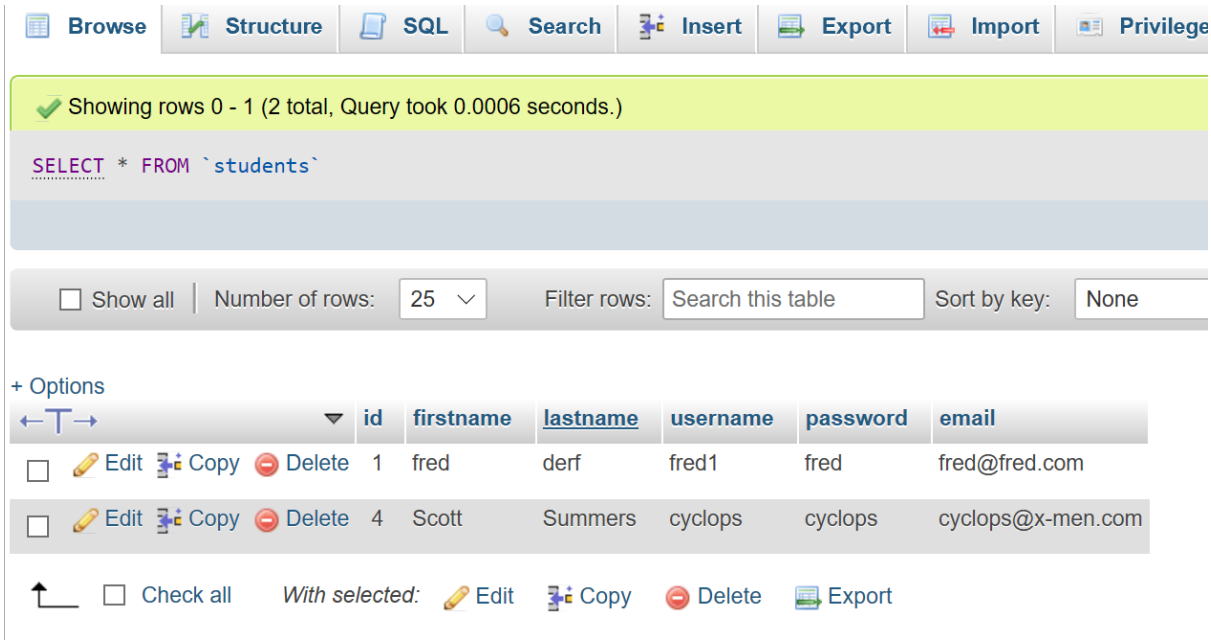
Result Addstudent




Result resolveaddstudent




Result phpmyadmin




Now that we have the ability to add to the database, it is time to make it viewable in a web browser, without the need of using phpmyadmin.

To do this, we will create a new function to list all of the students.

Functions.php

```php
function listUsers($dbConnect)
{
    echo'<table>
    <tr>
        <th>ID</th>
        <th>Name</th>
        <th>Username</th>
        <th>Password</th>
        <th>Email</th>
    </tr>
    ';
    $sql = 'SELECT * FROM students ';
    foreach ($dbConnect->query($sql) as $row) {
        echo '<tr>';
        echo '<td>'.$row['id'].'</td>';
        echo '<td>'.$row['firstname'].' '.$row['lastname'].'</td>';
        echo '<td>'.$row['username'].'</td>';
        echo '<td>'.$row['password'].'</td>';
        echo '<td>'.$row['email'].'</td>';
        echo '</tr>';
    }
} //eo function
```

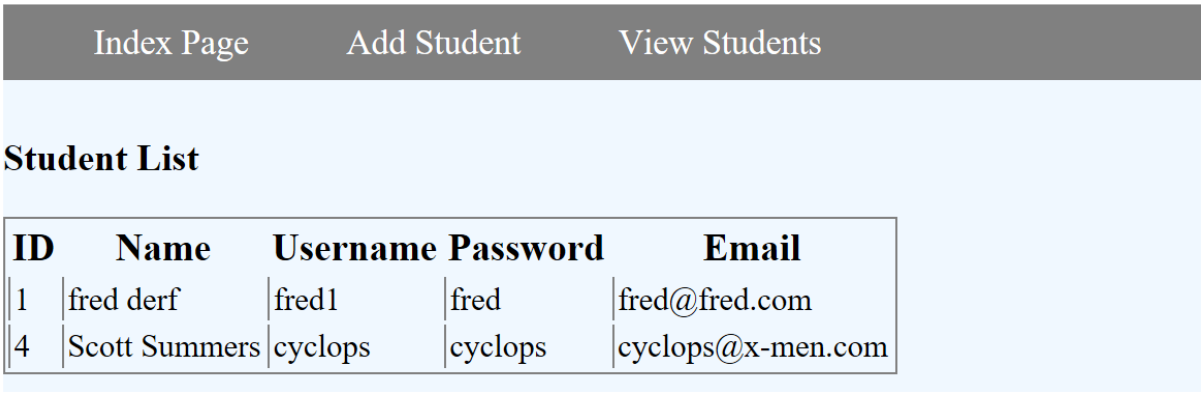
Viewstudents.php

```php
<?php
    include_once('../functions/functions.php');
     $dbConnect = dbLink();
    if ($dbConnect)
        echo '<!-- database connection established -->';
?>
<!DOCTYPE html>
<html>
    <head>
        <title></title>
        <style type="text/css">
        </style>
        <link href="../css/stylesheet.css" rel="stylesheet" type="text/css">
    </head>
    <body>
        <div id="navigation">
        <?php
            nav('viewstudents');
        ?>
        </div>
        <div class="view-students">
            <h3>Student List</h3>
        <?php
            listUsers($dbConnect);
        ?>
        </div>
    </body>
</html>
```

Stylesheet.css

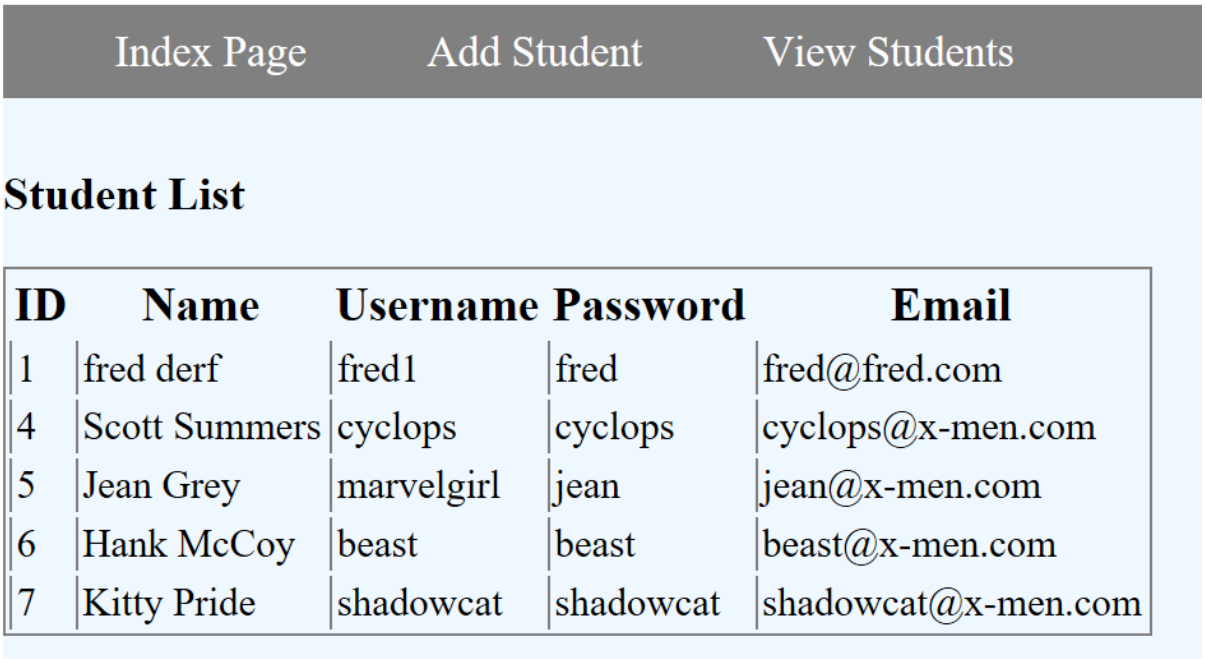
```css
#navigation{width: 90%; background-color: gray;padding-top: 10px; padding-bottom: 10px;}
#navigation a{font-size: 1.2em; text-decoration: none; margin-left: 50px;color: white;}
#navigation a:hover{text-decoration: underline;}
input{padding: 10px 10px 10px 10px; font-size: 1.2em; border-radius: 15px;}
.add-form input{margin: 10px 10px 10px 10px;}
.view-students{width: 90%; background-color: aliceblue; padding-bottom: 10px; padding-top: 10px;font-size: 1.1em;}
table,tr {border: 1px gray solid;}
td{padding: 2px 2px 2px 2px; text-align:left; border-left: 1px gray solid;}
th {font-size: 1.2em; font-weight: bold; text-align: center; padding: 2px 2px 2px 2px;}
```

Result




Test by adding in a few more students. Notice how the table grows and expands based upon the amount of information that is stored within the database.

Example:




This concludes the create and read aspects of databases. Next we will look at update and delete.

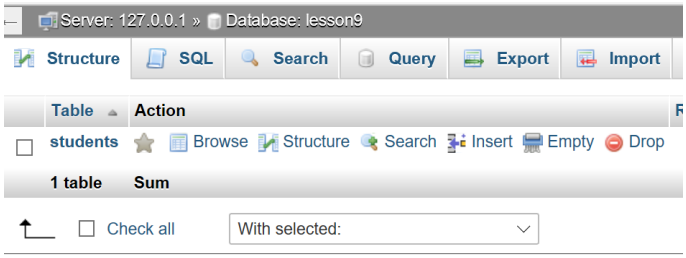
## Backup

This concludes tutorial.

As we are now using servers, and server information, there are two areas to backing up your information.

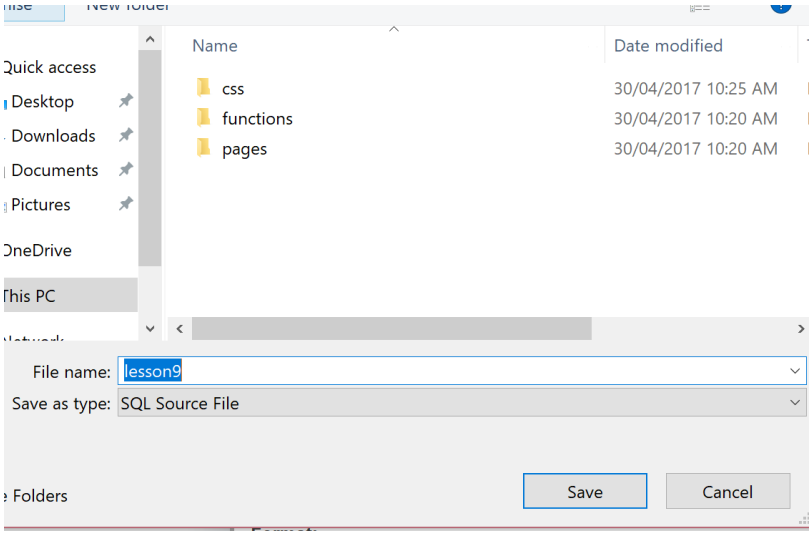
The first area is the htdocs folder. This is all of the php/html/css/javascript files. These are all needed to view the website.

Secondly, the database itself. To back up the database, go to phpmyadmin and navigate to the root area of the database

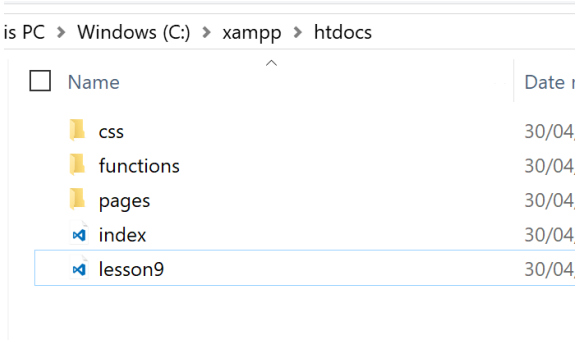



From here, click on export:

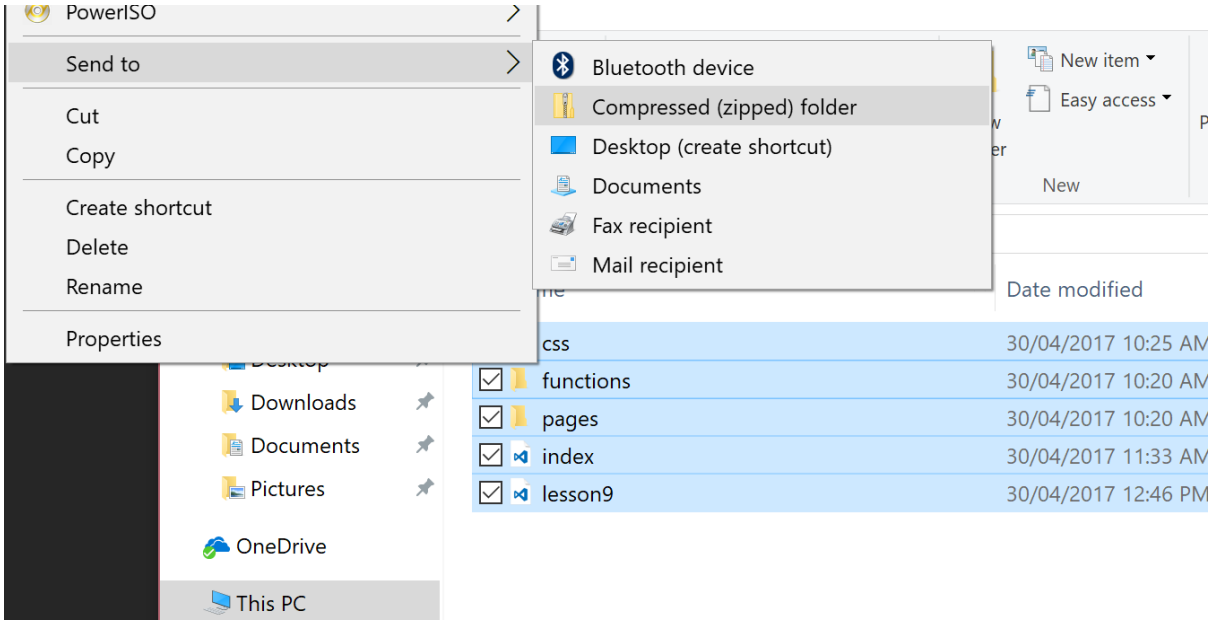

Click on go and save this .sql file into the htdocs folder



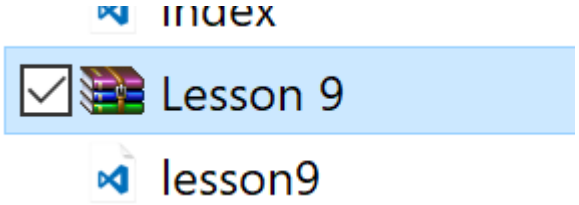
When viewing the htdocs folder you should see




Highlight everything and right click, then select send to compressed file




It will save a zip file, name this file appropriately.




If the icon is different, this is because on the machine this was created on, I use winrar as my compression tool.

Save your work (c:\xampp\htdocs) to USB, student drive, Onedrive, Dropbox or zip and email the folder to yourself.