

CSC 435 Computer Networks, Spring 2014

Instructor: Dr. Natarajan Meghanathan

Term Project Choice # 1: Web Development in a Linux Virtual Machine Environment

Due: April 25, 2014

Max. Points: 100

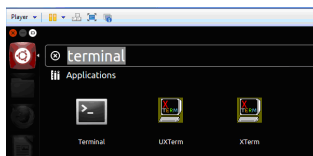
This project is for educational and awareness purposes only. We are not responsible for anyone using this project for any malicious intent. The objective is to illustrate web development in a Linux environment and show how packets reaching the web application can be captured/ viewed at another virtual machine running on the same network as that of the virtual machine hosting the web application. You will do this project involving an Ubuntu virtual machine (hosting your web application), Backtrack 5 virtual machine (on which we will run Wireshark) and the host machine (Windows/Linux or Mac machine) through which you interact with the web application. You will need to download VMware Player which is the virtualization software that will be used for this project. You will also need to download Ubuntu OS and Backtrack 5 to complete this project. If Ubuntu OS and Backtrack 5 are already running on a VMware Player, you could use them.

Installing VMWare Player


Download the latest version of VMware Player for your Operating System from <http://www.vmware.com/download/player/download.html>

Downloading and Installing Ubuntu OS

1. Download Ubuntu OS <http://www.ubuntu.com/download/desktop> (32-bit version should be sufficient and is recommended) and save it somewhere on your computer
2. Open up VMWare Player
3. Click on **Create a New Virtual Machine**
4. Select Installer disc image file (iso): browse for your Ubuntu .iso file and click **Next**
5. Type in your full name in the space provided. Use your J-number as Username (with a lowercase j). For your password, Select a password of your choice (easy to remember; but, difficult to find out by others). Click **Next** after entering the information.
6. Next, type in a name for your virtual machine (use your J-number again). Click **Next**.
7. On the next page, select **Store virtual disk as a single file**, and click **Next**.
8. Click **Finish** on the next page and wait for the OS to be installed.
9. Next, log into Ubuntu OS with your password and press **Enter**.
10. Click the **Player** menu, and go to **Manage** then **Virtual Machine settings**.
11. When the settings come up, make sure that the **Network Adapter** is set to **NAT**, and click **OK**.
12. Launch a terminal by clicking the **Dash Home** (indicated in the picture below) and typing **terminal** in the box provided. Then click the **Terminal** icon.



Installing XAMPP in the Ubuntu VM

1. In your virtual machine, open the Firefox web browser (Applications → Internet → Firefox) and navigate to the XAMPP for Linux webpage (<http://www.apachefriends.org/en/xampp-linux.html>). Follow the directions on this page to download and setup LAMPP.
2. Scroll down the page to the “XAMPP for Linux” section and click on the 32-bit version of XAMPP Linux 1.8.2 (or 1.8.3 or any available higher version) link to download the installer file. The file will be downloaded to the ‘Downloads’ folder in your VM. **Note that if you have a 64-bit Ubuntu Virtual machine, you should download the 64-bit version of XAMPP instead of the 32-bit version.**
3. After the download is over, launch a user terminal window (Click on Dash home  and type **term**). You need to have ‘super-user’ root permissions to install and run LAMPP. Now you need to set a root password using the ‘sudo passwd root’ command. It may ask for a password; enter your password to login to the Ubuntu VM. Or (as shown in the screenshot below), you may be directly prompted to select a UNIX password, enter a password of your choice and confirm that. Make sure your UNIX root password can be easily remembered (but difficult to be guessed!!; to keep it easy to remember, I would suggest having both the root password and the password you use to login to the Ubuntu VM to be the same). A screenshot of this step is shown below.

```
ubuntu@ubuntu:~$ sudo passwd root
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
ubuntu@ubuntu:~$
```

4. Now, login as root by running the command ‘su root’. You will be prompted to enter the root password that you selected in the previous step. The \$ prompt will now change to display that you are logged in as root as shown below.

```
ubuntu@ubuntu:~$ su root
Password:
root@ubuntu:/home/ubuntu#
```

5. You can now get to the Downloads folder using do a ‘cd Downloads’ from your current folder.
6. Find out the file name for your xampp file (in my case, it is **xampp-linux-1.8.2-5-installer.run**). You need to set the executable permissions for this file using the chmod +x command as shown in the screenshot below. After that simply start the installer by running it as **./xampp-linux-1.8.2-5-installer.run** Note that you should put the ./ (i.e., a . followed by /) in front of the file name to execute.
7. Go through the GUI-based XAMPP installation process; the XAMPP web server would start by default and a screen would appear showing the services running. [If the XAMPP web server screen does not show after installation is completed, you could go type **/opt/lampp/lampp start** and have XAMPP for Linux start (see screenshot below).

See screenshots in the next page.

```

root@ubuntu:/home/meghanathan# cd Downloads/
root@ubuntu:/home/meghanathan/Downloads# ls
xampp-linux-1.8.2-5-installer.run
root@ubuntu:/home/meghanathan/Downloads# ls
xampp-linux-1.8.2-5-installer.run
root@ubuntu:/home/meghanathan/Downloads# ls -l
total 106796
-rw-rw-r-- 1 meghanathan meghanathan 109351907 2014-04-20 12:43 xampp-lin
ux-1.8.2-5-installer.run
root@ubuntu:/home/meghanathan/Downloads# chmod +x xampp-linux-1.8.2-5-ins
taller.run
root@ubuntu:/home/meghanathan/Downloads# ls -l
total 106796
-rwxrwxr-x 1 meghanathan meghanathan 109351907 2014-04-20 12:43 xampp-lin
ux-1.8.2-5-installer.run
root@ubuntu:/home/meghanathan/Downloads# ./xampp-linux-1.8.2-5-installer.
run

```

```

root@ubuntu:~# /opt/lampp/lampp start
Starting XAMPP for Linux 1.8.2-5...
XAMPP: Starting Apache...already running.
XAMPP: Starting MySQL...already running.
XAMPP: Starting ProFTPD...already running.
root@ubuntu:~#

```

8. To test your XAMPP installation, open the Firefox browser and go to "<http://localhost>". If you see the XAMPP logo, your installation is complete.
9. We now need to create a folder to store your web application files. Open a Terminal window and type `su` to login as the root user. Once logged in, type `cd /opt/lampp/htdocs` to navigate to the htdocs folder on your VM. Once inside the htdocs folder, type `mkdir webapp`. This will create a new directory "webapp" inside the htdocs folder.
10. We also need to test whether you could connect to the website from your host machine. On the Ubuntu VM terminal, run the `ifconfig` command. You should be able to see the IP address of the VM. In my case (as seen in the screenshot below, it is 192.168.159.131). Open a web browser in the host machine and go to: `http://192.168.159.131/`

```

root@ubuntu:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:50:56:3e:d3:24
          inet addr:192.168.159.131  Bcast:192.168.159.255  Mask:255.255.255.0
          inet6 addr: fe80::250:56ff:fe3e:d324/64  Scope:Link

```

Tutorial on Web Development

We will now look at developing a sample web application with HTML and PHP. The application will be a simple calculator which will determine a student's grade based on several inputs accepted from a user. If you are already familiar with HTML and PHP, you may skip much this section and move on to the next section to create the HTML and PHP files for the web application needed for this project. We will use the HTML POST method to pass parameters.

HTML using POST

HTML code is used by a web browser to determine how a webpage should appear and what content should be displayed. HTML code is simple and quick and does not require any great amount of time by a programmer to become functional in its use.

The bulk of HTML code consists of tags. Tags tell the browser what elements are present in a page, where they are located, and what they contain. There are many different HTML tags, but we will only be using a couple:

- `<html>`
- `<body>`
- `<p>`
- `<form>`
- `<input>`
- `<table>`
- `<tr>`
- `<td>`
- `<a>`

In general, to open a tag, you type the name of the tag within `<>`'s, as in the list above. For most of the tags we will be using, it is also important to close the tag when you are done using it. This is done by inserting a `/` character after the `<` character. Closing some of the above tags would look like:

- `</html>`
- `</body>`
- `</p>`
- `</form>`
- `</table>`
- `<tr>`
- `<td>`
- ``

Note that the `<input>` tag does not have to be closed. Other tags that do not need to be closed are `
` (line break) and `<hr>` (horizontal rule, makes a horizontal divider).

To begin creating your HTML document, open a terminal window and type `sudo gedit` and press enter. Type in your root password and press enter to log in as the root user. In order to save things to your webapp folder, you must launch the gedit program as the root user.

On the first line, type `<html>`, then skip a few lines and type `</html>`. These tags will be the beginning and the end of our HTML document. Now, on the second line, type `<body>` and on the second-to-last line, type `</body>`. These will be the beginning and end of our body segment within the HTML document. Make sure that `</body>` comes before `</html>`. In HTML it is important that your tags are

properly nested. If you open <tag b> within <tag a>, you must close </tag b> before closing </tag a>. For example:

```
<html>
<body>
<p>
A paragraph goes between the 'p' tags.
</body>
</p>
</html>
```

is incorrect. The <p> tag should have been closed within the <body> tag since it was opened within the <body> tag. The correct syntax is shown below:

```
<html>
<body>
<p>
A paragraph goes between the 'p' tags.
</p>
</body>
</html>
```

Using indentation is an easy way to visually confirm that you have nested your tags properly. Whenever you open a new tag within an existing tag, indent once more. This way it will be obvious which tags should be closed first. For example:

```
<html>
  <body>
    <p>
      A paragraph goes between the 'p' tags.
    </p>
  </body>
</html>
```

Here it is easy to see where tags should be closed.

Once you have your <html> and <body> tags set up, we will insert some text into our webpage. Text is generally placed in between <p> and </p> tags to denote paragraphs.

Inside the body of the page, create 3 sets of <p></p> tags. Insert the following three paragraphs into the three sets of paragraph tags, respectively.

Paragraph 1:

We will create an app that will calculate a student's grade for a course when provided with the total number of points a student has earned in a category, and the percentage of the total grade that each category is worth.

Paragraph 2:

Each course has three areas in which a student receives credit: homework, quizzes, and tests. The amount of the grade that a category comprises for each class differs, however.

Paragraph 3:

For instance: In History 204, there are a total of 500 possible points for homework, 300 possible points for quizzes, and 1000 possible points for tests.

These paragraphs will be displayed on the web page and describe to the user what the page is and how the calculator works.

At this point, save the document into your webapp folder (/opt/lampp/htdocs/webapp/) as index.html. Now open a browser window and navigate to http://localhost/webapp/index.html and you should see the page you just created. Read the page and try to understand what our web application will do.

The example we have provided on the page still doesn't seem complete though, so we will add a little more information. We want the information to be included in the same paragraph as the example given, but we want it to start on its own line. To accomplish this, insert a
 tag after the period at the end of the sentence in paragraph 3. Now insert the following line between the
 and the </p>:

Additionally, homework is worth 25% of the final grade, quizzes are worth 25% of the final grade, and tests are worth 50% of the final grade.

Now save the file, and refresh the browser window containing your document. You will see that the sentence has been added on its own line at the end of paragraph 3.

At this point you should now create a paragraph before the first paragraph that contains your name and student id#. Save your document, and view it in the browser.

Now we will create the form. A form provides a place for users to enter data and make selections. A form contains several attributes, but the only ones we will be using are the method and action attributes. The “method” attribute tells the HTML page how the form will be submitted. There are several ways of submitting a form, but in this exercise, we will only be concerned with the “GET” method and the “POST” method. The “action” attribute tells the HTML page where to submit the form.

In HTML, attributes are enclosed within the opening tag and the values of attributes are enclosed within quotation marks.

For our <form> element, we will be using the “POST” method and we will be submitting our form to “app.php” (which we will create later). The syntax for setting up the form will be:

```
<form method="POST" action="app.php">
</form>
```

Again, make sure that you are properly nesting your tags. The <form> tags go (after the last <p></p> tag) inside the <body> tags. Now you should have:

```
<html>
  <body>
    <!-- after all the three paragraphs -->
    <form method="POST" action="app.php">
      </form>
  </body>
</html>
```

The first piece of information to get from the user will be the course name. Inside the form tags, insert an <input> like the following:

```
<input type="text" name="Name"> <br >
```

If you save and refresh your document, you will see that there is now a textbox on the page. The “type” attribute of the <input> tag tells the browser what type of input will be received, and the “name” attribute gives the input a name by which it can be referred to later.

You will notice that there is no text on the page telling what is supposed to go in the box, so enter the text Name: in front of the input tag in your document, save it, and refresh it in the browser.

Now we will create the framework for accepting the rest of the input. There are 9 numbers that need to be retrieved from the user, and the easiest way to neatly display these input boxes is with a table. Tables can be created easily in HTML, so we will insert a table that will look similar to the following table:

Category	Earned Points	Possible Points	Percent
Homework:			
Quizzes:			
Tests:			

To set up a table, inside the form tags and after your input box, insert the <table> and </table> tags. Now we must determine how big our table will be. The table above has 4 rows and 4 columns. In HTML, the process to create a table is to:

- create the table
- create a row in the table
- enter the first item on the row
- enter the second item on the row
- ...
- enter the last item on the row
- create the next row

In this way, the table is created one row at a time. The tag for a table row is <tr>. Inside the table, create four sets of <tr></tr> tags, one for each row of the table.

Now that the rows have been created, each row will have 4 elements. Inside each <tr></tr> set, insert four sets of <td></td> tags for the four elements of each row. When you are finished, you should have:

```
<table>
  <tr>
    <td> </td>
    <td> </td>
    <td> </td>
    <td> </td>
  </tr>
  <tr>
    <td> </td>
    <td> </td>
    <td> </td>
    <td> </td>
  </tr>
  <tr>
    <td> </td>
    <td> </td>
    <td> </td>
    <td> </td>
  </tr>
  <tr>
    <td> </td>
    <td> </td>
    <td> </td>
    <td> </td>
  </tr>
</table>
```

```

<tr>
    <td> </td>
    <td> </td>
    <td> </td>
    <td> </td>
</tr>
</table>

```

The elements of the first row will be the headings for the columns. We would like our headings to be in bold-face, so we will enclose the text of our column headings inside `` tags. The syntax of the first heading will be `Category`. Now fill in the remaining three headings, save, and check your document.

Now we will create the input boxes in the table. For each box, you will create the input tag:

```
<input type="text" name="" ">
```

In the space for the name attribute, enter a unique name for each input field, such as HomeworkEarned, HomeworkPossible, and HomeworkPercent, etc. Each name attribute should NOT contain any spaces, and it should make sense, since we will be using these names in our PHP program to get the values in these fields. Using this naming convention, the first input field would be

```
<input type="text" name="HomeworkEarned">
```

Make sure that you place each box inside the appropriate `<td></td>` tags. Once you have completed creating your input fields, save the document, and view it in your browser to confirm that it looks like the table above.

Now that the table is complete, we need a way to submit our form. After the `</table>` tag, we will create one last `<input>` element, the submit button. Below the `</table>` tag, but before the `</form>` tag, insert the following line:

```
<input type="submit" value="Calculate">
```

This creates a special type of input element, the "submit" type, which is a button which will activate the action attribute of our form. The value attribute of the submit input type determines what the button will say. Now save your document, and view it in the browser. If you have done everything correctly, it should look like the image below. If not, go back through what we have done so far and try to locate and correct any errors.

Natarajan.Meghanathan, J123456

We will create an app that will calculate a student's grade for a course when provided with the total number of points a student has earned in a category, and the percentage of the category is worth.

Each course has three areas in which a student receives credit: homework, quizzes, and tests. The amount of the grade that a category comprises for each class differs, however.

For instance: In History 204, there are a total of 500 possible points for homework, 300 possible points for quizzes, and 1000 possible points for tests. Additionally, homework is worth 25% of the final grade, quizzes are worth 25% of the final grade, and tests are worth 50% of the final grade.

Course Name:

Category	Earned Points	Possible Points	Percent
Homework:	<input type="text"/>	<input type="text"/>	<input type="text"/>
Quizzes:	<input type="text"/>	<input type="text"/>	<input type="text"/>
Tests:	<input type="text"/>	<input type="text"/>	<input type="text"/>

PHP using POST

Now that we have a working HTML form that will send data, we need a way to collect and perform some functions on that data. We will do this with PHP.

In the same folder (“/htdocs/webapp/”) where your index.html file is located, create a new file called “app.php” and open it with a text editor (gEdit, nano). Inside this file will be our PHP code. PHP code runs on the server and performs all of its operations before a webpage is rendered. Because all the operations occur before the webpage is created, we can use PHP to dynamically insert HTML into a page depending on the input provided.

Within the app.php file, create open and closing <html> and <body> tags the same way you did in your index.html file. All of our PHP code will be placed inside the <body></body> tags.

Similarly to how HTML code is enclosed within <html></html> tags, PHP code is enclosed within <?php and ?> tags. Insert these into your document. So far your app.php code should look like:

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

Now you are ready to begin writing PHP code. PHP code is similar to many other languages in syntax, but one important difference is that variables do not have to have a type associated with them, but they must start with a ‘\$’ character. The declarations

```
$value = 5;
```

and

```
$value = “five”;
```

have the same syntax, but the first is treated as a number, because that’s what is assigned to it, and the second is treated as a string, because that’s what is assigned to the second one.

The first thing we must do in our PHP code is to retrieve the values from our form. This is done with the \$_POST[‘ ’]; expression, where the content of the ‘ ’ is the value of the name attribute of a form’s input element. Using the naming convention listed above for the input elements’ names, our first three values would be read in by:

```
$hw_earned = $_POST[‘HomeworkEarned’];
$hw_possible = $_POST[‘HomeworkPossible’];
$hw_percent = $_POST[‘HomeworkPercent’];
```

where \$hw_earned, \$hw_possible, and \$hw_percent are the names of our new PHP variables and HomeworkEarned, HomeworkPossible, and HomeworkPercent are the names of our form’s input elements. Now write code that will retrieve the other 6 table elements and the course name from the form and store them as variables.

Now we will perform the mathematical operations that will calculate the grade. The formulas that will be used is

$$\text{Points} = \frac{\text{Earned} * \text{Percent}}{\text{Possible}}$$

where there will be three different “Points” values (HomeworkPoints, QuizPoints, and TestPoints).

Once these three values are calculated, simply adding them together will give the final grade:

```
FinalGrade = HomeworkPoints + QuizPoints + TestPoints
```

Create variables in PHP and perform these calculations. The first one is done for you below:

```
$HomeworkPoints = ( ( $HomeworkEarned * $HomeworkPercent ) / $HomeworkPossible );
```

Now we would like to display the information we have calculated to the user. In PHP, the echo command can be used to print out information to the screen (in this case, to the webpage before it is rendered).

```
echo 'This text will be printed.';
```

We can also include HTML tags within an echo statement:

```
echo '<b>This text will be printed and bold.</b>';
```

First, we will print out the course name that was retrieved from the form. To print out the value of a variable, simply call the echo command without quotation marks:

```
echo $FinalGrade;
```

Write the PHP code to output all the values from your form. Be sure to include the
 tag between lines so that each value will be printed on its own line. The first four values have been written below:

```
echo 'Course Name: ';  
echo $CourseName;  
echo '<br>';
```

```
echo '<b>Homework</b>';  
echo 'Points Earned: ';  
echo $HomeworkEarned;  
echo '<br>';
```

```
echo 'Possible Points: ';  
echo $HomeworkPossible;  
echo '<br>';
```

```
echo 'Percent: ';  
echo $HomeworkPercent;  
echo '<br>';
```

Now, repeat the above procedure to output the values of the rest of the form's elements (3 for Quizzes, 3 for Tests).

When you have completed this, insert the code that will print out the rest of the calculated values including the final grade. The first one has been done for you:

```
echo '<br><br>';  
echo '<b>Homework Points: </b>';  
echo $HomeworkPoints;
```

When you save this file, you should be able to go to your browser, enter values in the form, and click the "Calculate" button. You should then be directed to your app.php page where the values you have entered are displayed. If the page does not display correctly, you will need to go back through what we have done so far and try to find and fix your error. If there is an error within your PHP code, the text displayed on the app.php page will tell you what your error is and where to find it. If your app.php page simply

displays the raw code, make sure that you have installed LAMPP correctly, started it, are saving your files to the proper directory, and are accessing your pages through the localhost (http://localhost/webapp/).

Input Validation using PHP

Now that we have seen how to use HTML forms and URLs to pass data to a PHP program, we will see how to use PHP to check the validity of our input. Input validation is an important part of software development. You should never trust that the input provided to your program will be valid, the program should always check the data itself before it performs any operations on the data.

One of the easiest ways to generate an error in our PHP code is to try dividing by zero. In the calculations above, the value of Possible (HomeworkPossible, QuizzesPossible, etc.) is a denominator and a user could enter a value of 0 for these values, which would generate an error. In order to prevent this, we must check the values of our Possible values before we calculate with them.

Your calculations and output will need to be enclosed inside an `if () { }` block which will ensure that all the denominator values passed are valid. The code for this will be as follows:

```
if ( ( $HomeworkPossible != 0 ) && ( $QuizzesPossible != 0 ) && ( $TestPossible != 0 ) )
{
    $HomeworkPoints = ...
    ...
    echo $FinalGrade;
}
```

Now go back to your input form (index.html) and enter a value of 0 for one of the “Possible” fields of your form. When you submit the form, you will see that the values are neither calculated nor are they displayed.

Now, since most schools do not allow anyone to have a grade of higher than an A, and a grade of 100 is usually the highest A you can have, you will determine whether the final calculated grade is higher than 100 or not. Insert code so that once the final grade is calculated, if it is greater than 100, it is reassigned a value of exactly 100. Once you have done that, test your application to make sure that the app.php page will not display a value of 100.

Running the Sample Web Application over a Network

Now that you have your web application working, it is time to test it over the network. In your Ubuntu VM (where you have been writing and testing your web application so far, open a Terminal window and type `ifconfig`. This will give you the network information about the VM you are using. You should see a line that says “inet addr:” and lists four numbers separated by periods. This is the IP address of your virtual machine.

Now, minimize your VM and open a browser window in your host environment (Windows, Mac, Linux). Type “http://” followed by the IP address of the virtual machine, followed by “/webapp/” (for example, if the IP address of your virtual machine is 192.168.159.131, then you would type http://192.168.159.131/webapp/).

Web Application to Develop in this Project

Now that, you have gone through an informative tutorial (hopefully!!) on HTML and PHP, you are required to do the following project using the POST method of retrieving data from HTML forms. I

suggest naming your HTML file as index.html and the PHP file as app.php. Put both your HTML and PHP files inside a folder called **chemapp** (create this folder inside your **htdocs** folder). To test your web application from the web browser in your host machine, you should find out the ip address of the Ubuntu VM and then visit the chemapp folder. In my case, I would do: <http://192.168.159.131/chemapp>.

The objective is to develop a web application that computes the molecular weight of a chemical compound based on the number of atoms and the atomic weights of each of the constituent elements. For example, the compound C₁₂H₁₅ClO₃ (Clofibrate) has 12 carbon atoms, 15 hydrogen atoms, 1 chlorine atom and 3 oxygen atoms; the atomic weights of carbon, hydrogen, chlorine and oxygen are 12.01, 1.01, 35.45 and 16.00 respectively. Hence, the molecular weight of C₁₂H₁₅ClO₃ is $(12 \times 12.01) + (15 \times 1.01) + (1 \times 35.45) + (3 \times 16.00) = 242.72$. The weight contributed by each element is the product of the number of atoms of the particular element in the compound and the atomic weight of the element.

Create an HTML form that inputs the *Name of the chemical compound, the formula, the number of atoms of each element and the atomic weights of each of the constituent elements*. Your HTML form should have sufficient number of rows (one row for each instance of appearance of an element in a formula) to input information about the constituent elements of a compound. Your PHP code should stop computing the molecular weight and display the total molecular weight once it encounters a blank field or a null value for the Element symbol in a row. **You should have at least one more row than the number of rows needed to input the information for your compound, and you should have a validation (in your PHP code) to test for reading a blank field to decide whether you are done reading the input values.**

Each row of your form should have fields that input the following value:

Element symbol	Number of atoms of element	Atomic weight of element
----------------	----------------------------	--------------------------

All of these data are submitted to a PHP page using the POST method (provide a Submit button at the bottom of the HTML page). In the PHP page, all of the above data are retrieved. In the PHP page, you compute the molecular weight of the compound (similar to the example explained above) and display the value along with the compound name, formula and the total molecular weight. If any of the input values are negative or zero, then your program should display an error message.

Note that if a compound has an element appearing at more than one instance in its formula, you need to input as separate instances. For example, in the case of C₂H₅OH, you should have 4 rows input as follows:

C	2	12.01
H	5	1.01
O	1	16.00
H	1	1.01

The following list shows the atomic weight of some common elements:

Hydrogen (H)	1.01	Aluminium (Al)	26.98
Carbon (C)	12.01	Silicon (Si)	28.09
Nitrogen (N)	14.01	Sulphur (S)	32.07
Oxygen (O)	16.00	Chlorine (Cl)	35.45
Flourine (F)	19.00	Potassium (K)	39.09
Sodium (Na)	22.99	Calcium (Ca)	40.08
Magnesium (Mg)	24.31	Iron (Fe)	55.85

START RECORDING YOUR VIDEO FROM HERE

Record the following:

(1) Display your code and explain how the control flows from an input entered by the user to the output displayed.

(2) Testing with Chemical Compounds

Enter the formula for the chemical compound assigned to you and show that your web application computes the molecular weight of the compound and prints it along with the name. ***Repeat the above testing for two other compounds picked from the above table. Show clearly which two compounds you picked in your video recording and the final report***

Student Name	Compound Name	Compound Formula
Aaron Barker	Baking Soda	NaHCO ₃
Tiffani Gardner	Bleach (Liquid)	NaClO
Gregory Lockett	Cream of Tartar	KHC ₄ H ₄ O ₆
Cameron Taylor	Epsom salt	MgSO ₄ H ₂ O
Carlos Ware	Freon	CF ₂ Cl ₂
Karrington Lewis	Grain alcohol	C ₂ H ₅ OH
Rashad Evans	Hypo	Na ₂ S ₂ O ₃
Deja Knight	Plaster of Paris	CaSO ₄ H ₂ O
Sarah Price	Potash	K ₂ CO ₃
Alain-Daniel Wa-Baguma	Saltpeter	KNO ₃
Deandrea Whinsenton	Sugar	C ₁₂ H ₂₂ O ₁₁
Allison Gray	Washing Soda	Na ₂ CO ₃ H ₂ O

Hard Copy

Your HTML Code and PHP code for the web application, and screenshots showing the submission (entering of input values in the HTML page for the chemical compound assigned to you) and the printing of the molecular weight along with the name of the chemical compound and its formula in the app.php page obtained after clicking the Submit button in the HTML page. Repeat the same for two more compounds.