COMPX322-20X **Assignment One: AJAX-driven Weather Watcher Application**

Due Date: **Thursday 2 April 2020, 5pm**

For this coursework you are to implement a web application that supports a simple weather watcher. Your implementation will require use of the following:

- Validated HTML; *i.e*. without in-line formatting, and with well-formed open and closing tags for HTML elements
- Cascading Style Sheets (CSS)
- JavaScript
- DHTML (modifying web page content using JavaScript)
- Server-side PHP scripting for dynamic generation of web page content and interaction with a MySQL database
- A MySQL database containing appropriate tables to store the data required by the application
- AJAX for partial updating of web page content in an asynchronous manner

Application description

The **minimum functional requirements** for the application are as follows:

- it consists of a **single** HTML page called *index.html* with a separate CSS document to control layout and appearance
- **all** the relevant files for the assignment submission/application are stored in a folder under *htdocs* called *weather*, so that executing the app is done by loading a URL that looks like this: **http://localhost:8888/weather** (replace the 8888 with the port that your server is listening on).
- it allows a registered user to log in (note: you do **not** need to enable new users to register)
- once a user is logged in it displays short weather information for their set of towns
- clicking on one of the town listings will display expanded weather information for that town
- the user can remove existing towns and add new ones from a predefined list (note: you can use the town listings in the provided data table and do **not** need to enable addition of new towns)
- all functionality is achieved within a single page (URL) without the user needing to refresh the page in their browser or navigate to a new page. This is where the use of AJAX is essential. In a non-AJAX implementation the entire page would need to be dynamically constructed on the server and retrieved whenever the display changes. In an AJAX implementation segments of page content can be retrieved from the server and used to update parts of the display.

What you need to do (in whichever order you think appropriate)

a) Import the provided MySQL table containing the town weather information and users. In a real world solution the weather information would be updated regularly to reflect changes – you do not need to handle this but can assume such a process exists.

b) Make any necessary changes to the MySQL tables to hold any remaining information required for your application (you can also add tables if you choose).

c) Populate your database tables with *appropriate* test data that will allow you to **fully** test your application. Do not remove the test user as we will use this for marking.

d) Design and implement the layout of your web page using well-formed and valid HTML in combination with one or more CSS stylesheets which should be stored in separate file(s).

e) Implement PHP scripts to retrieve and add the data needed by your application.

f) Implement any required JavaScript code that is needed to support your application. This includes code to handle asynchronous AJAX requests and responses to and from the server. Keep your JavaScript code in a separate file.

g) **NOTE**: your PHP scripts should return data to your HTML page using AJAX. You should **not** display content directly from your PHP scripts.

Important: For this assignment you must NOT use libraries – make sure you hand code 'raw' AJAX requests.

Guidance

Plan your solution before you begin writing code. Use paper and pencil and make sure you have a clear idea of what is required and how you will solve each aspect of the requirements before you sit down at a computer.

Populating your database tables with suitable test data at an early stage is an important step—it makes you think about the data involved and makes it easier to test your solution as you develop it in an incremental/modular way.

It is probably best to consider a non-AJAX implementation first so that you can determine that your PHP scripts and database operations are working correctly.

Take an incremental/modular approach to implementation... don't try to implement all aspects in one go. You should aim to get the minimum requirements implemented before you think about any extensions you might want to add.

Once you know that your PHP scripts work correctly, modifying your application so that they are called asynchronously using AJAX will be much easier.

Design the appearance of your HTML pages on paper, then transfer the design into implementation. Do not design by writing and modify HTML.

Do not code presentational aspects of the web pages such as fonts, font sizes and colours, background colours and so on into the HTML. Use CSS.

Do **NOT** use HTML tables to format your page. Use **<div>** elements and position them using CSS. You should only use HTML tables to display data in a tabular form.

Ensure that the HTML source of the pages created by your application is valid HTML, using an HTML validator.

Important: Your code **must** use a separate PHP file called `dbconnect.php` to connect to the database, exactly as shown below (but with *username* and *password* replaced accordingly). Do not make any other changes to this file, or your submission may not work when we come to mark it.

```
<?php
     try {
          $con = new PDO('mysql:host=localhost;dbname=COMPX322',
                          'username', 'password');
     } catch (PDOException $e) {
          echo "Database connection error: " . $e->getMessage();
          die();
     }
```

What to submit and how

All of your material for this assignment must be submitted electronically on Blackboard.

Compress (ZIP) the contents of your weather folder (see above) and upload your **weather.zip** file to the **Assignment 1 Submission** area on Blackboard. No other method of submission will be accepted.

When we mark your submission, your `dbconnect.php` file will be replaced by an identical one containing a different username and password, but connecting to a database with the same name, COMPX322.

Make sure all work submitted is your own — you must be familiar with the University regulations on plagiarism. You are free to discuss solutions with your classmates but this is an **individual** assignment so you should prepare all of the code on your own. **You may be asked to explain your solution or parts of your code as part of the assessment process.**

How your work will be assessed

The assignment will be marked out of 50 as follows:

• Application meets minimum functional requirements	15 marks
• AJAX is used to retrieve and display all data without a page reload	10 marks
• PHP scripts correctly handle data and interact with MySQL server	10 marks
• Database has a sound design and includes appropriate test data	5 marks
• HTML/CSS used **appropriately** to create usable layout and clear presentation	7 marks
- correct use of HTML elements	
- CSS used for functional and aesthetic design purposes	
• All code is suitably commented	3 marks
	50 marks = 100%

Note: marks will be deducted if we have to edit your PHP scripts (other than `dbconnect.php`) to correctly connect to the MySQL server so make sure you test your implementation carefully and submit code that will successfully connect.

Optional Extras

These are provided as ideas for things you might also want to do to challenge yourself. They are intended for you to do on your own so lab/tutor assistance will not be provided. They are not worth extra marks, but if you produce something impressive you get the bragging rights of me showing off your application in class.

- Add visual icons representing different types of weather to match the data displayed and make the information more visually appealing (hint – don't hard code this for each town as the weather may change!)
- In a real-world implementation weather would be refreshed at regular intervals (e.g. once ever 30 minutes). Add this refresh to your implementation. If you want to see how it would look in action (where the database being queried is also updating regularly) you can simulate this by adding a randomize function to your PHP script which changes the values coming from the database (or just creates a random value and ignores the database altogether).

  NOTE – if you implement this then do so in a second page so your original application continues to meet the marking requirements.