COMP333-18A   Assignment One

Due Date: Friday March 23rd, 5pm

AJAX-driven Chat Application

For this coursework you are to implement a web application that supports a simple chat facility. 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*** requirements for the application are as follows:

- it consists of a **single** html page with a separate CSS document to control layout and appearance
- it displays a list of registered users separated into two groups, those currently logged in and those who are not logged in
- a new user can register
- an existing user can login, once successfully logged in the application shows all messages that have been posted that day
- logged in users can compose and send a message, the new message is then displayed in the list of messages for the day
- all functionality is achieved within a single page (url) without the user needing to refresh the page in their browser. 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.

A correct, well-designed implementation of the minimum requirements described above will earn you up to **96%** of the possible marks.

In addition to these minimum requirements there are several additional components you *could* consider, for example:

- the ability to show all messages posted by a particular user
- the ability for a user to update their account (e.g. change their username whilst retaining all of their previous messages)
- the ability to show all messages posted on a particular date
- …. and many more

If you have the time and motivation to tackle one or more of these (or some other suitable enhancements which use AJAX requests), correct and well-designed implementation will earn you the final **4%.**

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

a) Design and implement the MySQL tables required to hold all the information.
b) Populate your database tables with **appropriate** test data that will allow you to **fully** test your application.
c) 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).
d) Implement PHP scripts to retrieve and add the data needed by your application
e) 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.
f) 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 should 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. Suitable data means that you have added enough variety to fully test your system **and** used data that would be appropriate to show to a client.

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 (eg 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.

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

What to submit and how

All of your material for this assignment must be submitted electronically on Moodle.
Assuming that all parts of your application are within a directory called **comp333assn1** within your **course_html** directory:

Compress the directory into **comp333assn1.tar.gz**

Upload the compressed file to the **Assignment 1 Submission** link on Moodle. No other mechanism for submission will be accepted.

Make sure all work submitted is your own, you should 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.

When you submit a file Moodle will ask you to confirm that what you have submitted is your own work, and will provide you with a 'receipt' that establishes that you have indeed submitted something.

How your work will be assessed

The assignment will be marked out of 50 as follows:

• Application meets minimum functional requirements	20 marks
• AJAX is used to retrieve and display all data without a page reload	15 marks
• PHP scripts correctly handle data and interact with MySQL server	5 marks
• Database has a sound design and appropriate test data	3 marks
• HTML/CSS used appropriately to create usable layout and clear presentation	5 marks
	(48 = 96%)
• Additional feature (using AJAX) correctly implemented	2 marks
	(50 = 100%)

Note: marks will be deducted if we have to edit your PHP scripts to correctly connect to the UoW MySQL server so make sure you test in that environment and submit code that will successfully connect.