COMP3322 Modern Technologies on World Wide Web

Assignment Three

Total 18 points

Deadline: 23:59 April 10, 2024

Overview

You are going to develop a Web Chat application using PHP and MySQL database. Users use the @connect email addresses to register and log on to the Chat application. Once logged in, users will be able to exchange real-time messages with other logged-on users via the Chatroom. The system would automatically log out the users when the users haven't participated in chatting for a specific duration.

Objectives

1. A learning activity to support ILO 1 and ILO 2.
2. To practice how to use JavaScript, AJAX, PHP, session control, and MySQL to create a web-based Chat application that supports real-time messaging.

Specification

You develop the application using the course's LAMP docker containers.

You will develop at least four PHP programs – login.php, check.php, chat.php, and chatmsg.php, and the corresponding CSS styling files (login.css and chat.css) and JavaScript files (login.js and chat.js). You should assume all files are placed inside a folder in the public_html directory of the web server docker container.

- login.**php** – a webpage for users to register with the system or log on to the system.
- check.**php** – a PHP program for handling the client AJAX request, which checks whether a user with a specific email address has an account in the system, i.e., has registered already.
- login.**css** – the styling file contains all CSS style rules for the login.php page.
- login.**js** – the scripting file contains all JavaScript code to support the dynamic features of the login.php page.
- chat.**php** – a webpage for users to read and send messages to the chatroom after successfully logging into the chatroom.
- chatmsg.**php** – a PHP program for handling clients' AJAX requests in sending/downloading messages to/from the chatroom.
- chat.**css** – the styling file contains all CSS style rules for the chat.php page.
- chat.**js** – the scripting file contains all JavaScript code to support the messaging function of the chat.php page.

Other than the jQuery library, you are **not allowed to use external libraries** for this assignment.

Requirements

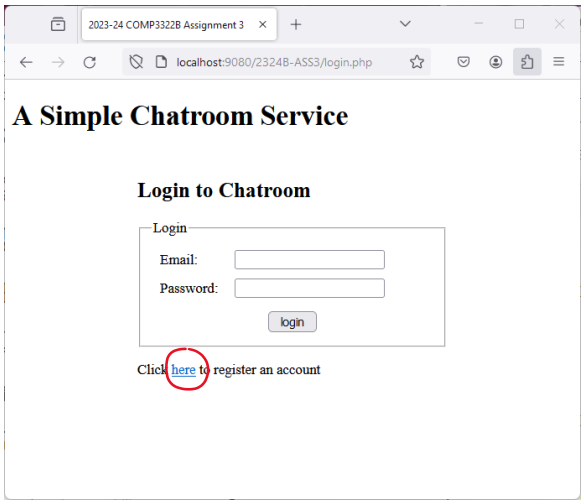
Login and Registration

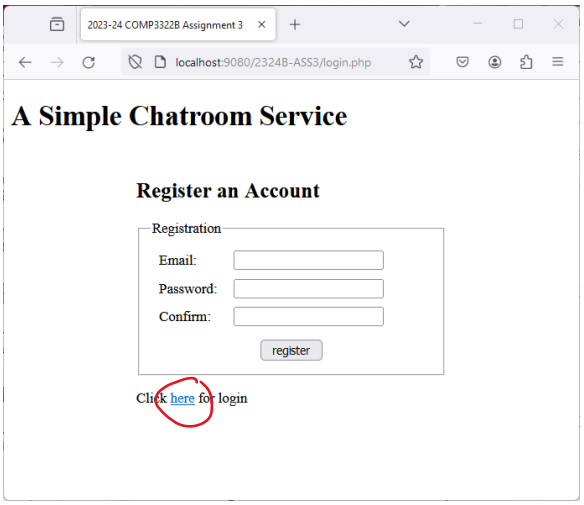
Four programs are involved in providing services for these two functions. They are the login.php, check.php, login.js, and login.css files. Here are the basic services provided by this set of programs. Implement the four programs to handle these requests:

- GET /login.php

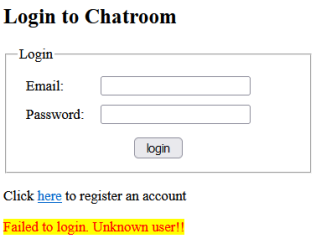
- GET /login.php?action=signout
- POST /login.php with type=login and required user credentials in the message body
- POST /login.php with type=register and required user credentials in the message body

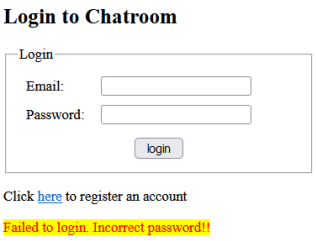
1. Users request the login page by sending a GET request to /login.php. The page **always shows** the login form when completely loaded. Users can click the provided register link (here) to switch to the registration form. Likewise, users can click the login link (here) to switch back to the login form. Therefore, the page should include both forms and utilize JavaScript to enable seamless switching between views.



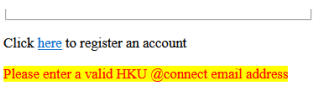


2. Users enter the required fields in the login form for authentication. Once the users click the "login" button, the program sends a POST request to /login.php with the **message body** carries three parameters: type=login, user={email address}, password={password}, where {email address} and {password} are the values entered by the users in the required input fields.
   If successfully authenticated, the server should redirect the browser to the chat.php page. When failed to authenticate, the program should **display a notification below** the form to alert and explain the cause of the failure.

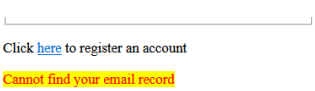




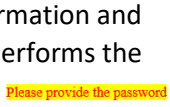
3. The programs also make use of JavaScript to perform client-side validation on the form data.
   a. Check whether the **email address** belongs to the **@connect.hku.hk** domain and display a notification if verification fails. The program performs the checking once the email input field has lost focus.



b. Check whether the email address **has already been registered** and display a notification if no matching record. The program performs the checking once the email input field has lost focus. (**Noted: the user may ignore this alert and be allowed to submit the login request**.)

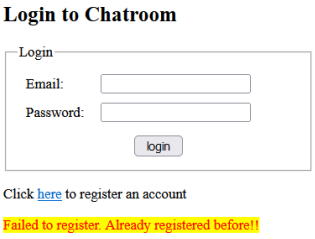


c. Check whether the user has entered all required information and display a notification to alert the user. The program performs the checking when the user clicks the 'login' button.



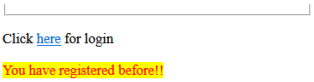
d. The program should not initiate the POST request when fails the above verification steps a or c.

4. Users enter the required fields in the registration form for adding an account and logging in to the chatroom after successful registration. Once the users click the "register" button, the program sends a POST request to /login.php with the **message body** carries three parameters: type=register, user={email address}, password={password}, where {email address} and {password} are the values entered by the users in the required Email and Password fields.
   If successfully registered, the server should **redirect** the browser to the chat.php page.
   When **fails to register**, the program should **redirect** to the login form and **display a notification** below the form to explain the cause of the failure. For example, the user tries to register a new account with an email address matched with an existing account.

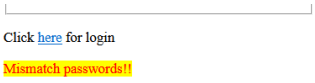


5. The programs also make use of JavaScript to perform client-side validation on the form data.
   a. Check whether the email address belongs to the **@connect.hku.hk** domain and display a notification if verification fails. The program performs the checking once the Email input field has lost focus.

   b. Check whether the email address is **already associated** with an existing account and display a notification if there exists a matched account. The program performs the checking once the email input field has lost focus. (**Noted: the user may ignore this alert and be allowed to submit the registration request.**)



   c. Check whether the two passwords are the same and display a notification if is mismatched. The program performs the checking when the user clicks the 'register' button.



   d. Check whether the user has entered all required information and display a notification to alert the user. The program performs the checking when the user clicks the 'register' button.
   e. The program should not initiate the POST request when fails the above verification steps a, c, or d.
6. The program should **clear all notification messages** once the user has addressed the issues.
7. When the user successfully logs in (via the login or registration page) to the chatroom, the system should establish a new session for this user. The session will be terminated when the user closes the window or clicks the "Logout" button on the chatroom page. In addition, the session will be terminated automatically when the user does not have any messaging activities in 120 seconds.
8. When the user clicks the "Logout" button (on the chatroom page), the program sends a GET request to /login.php with the **query string** action=signout. By receiving this GET request, the

program closes the current session, clears all session variables, and redirects the browser to the login page.

9. User accounts
   You should create a table in the database (db3322) to store the **user account** information – email address and password. Here is a reference SQL statement for creating the user account table.

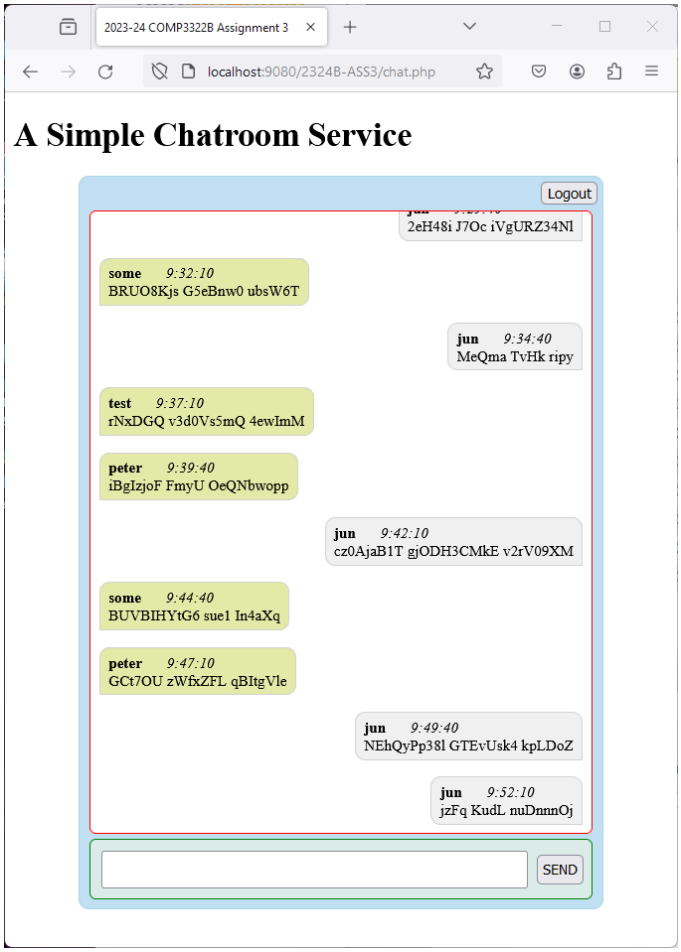
```
CREATE TABLE `account` (
  `id` smallint NOT NULL AUTO_INCREMENT,
  `useremail` varchar(60) NOT NULL,
  `password` varchar(50) NOT NULL,
  PRIMARY KEY (`id`)
);
```

   (**Note: you can design your schema for the account table**.)

Session Control and Chat Service

Four programs are involved in providing session control and chat service. They are the chat.php, chatmsg.php, chat.js, and chat.css files.

chat.php



10. After successfully logging into the system, the browser will be redirected to the chat.php program. This program provides the main **UI of the chatroom**. On the right side is an example implementation of the UI. It should contain **a title** and **a chat message window** for displaying chat messages. Below the chat message window, there should be **an input area** for the user to enter a chat message and **a "SEND" button** for sending the chat message to the chatroom. In addition, there should be a "Logout" button for the user to log out and close the session.

The interactions between the browser and the server are handled by the chat.js and chatmsg.php programs. Here are the basic requirements of the two programs.

chat.js, chatmsg.php, and chat.css

11. After the chat.php program has successfully loaded, the chat.js program will send an **AJAX GET request** to the chatmsg.php program to retrieve all chat messages that **were posted in the past hour**. All returned chat messages will be displayed on the chat message window with messages sent by the current user appearing on the right side and messages sent by other users appearing on the left side. The messages will be **ordered chronologically** based on the time they were received by the server. Each chat message will display the username, the time, and message content. In

addition, if there are more messages to be shown in the message window, the program automatically **scrolls to the bottom** of the window to show the latest message.

12. After the user clicks the "SEND" button, if the input field is not empty, the chat.js program will send an **AJAX POST request** to the chatmsg.php program with the message (in which all leading and trailing spaces and newlines **are trimmed**). If the input field is empty, the program should ignore this click event. In addition, the program should display the message on the chat message window. (**Note: When to display the message on the message window is an implementation decision**.)
13. The chat.js program needs to **constantly poll** the chatroom (chatmsg.php) for updated chat messages. To simulate a real-time chatroom, the program needs to periodically (**every 5 seconds**) send an **AJAX GET request** to the chatmsg.php program to request the latest set of chat messages (**since the last polling**). Once received, display them at the end/bottom of the chat message window. The chat message window should support scrolling for the user to scroll up and down to view those chat messages.
14. **Session control**
    a. Users cannot access chat.php and chatmsg.php without an active session. The two programs should reject such access if they find that the user hasn't authenticated and logged on. If encounters such cases, the chat.php program should **redirect** the request to the login page and the chatmsg.php program should **reject** the request with the **401 status code**.
    b. If a user, who has already authenticated, tries to access the login.php page again (with another browser tab), the program should **direct** the browser to the chat.php page (rather than showing the login.php page).
    c. When the user clicks on the "Logout" button, the chat.js program should send a GET request to the login.php program with the query string action="signout".
    d. The chatmsg.php should keep track of the last messaging time of the user and once the user hasn't sent any messages (i.e., **idling**) in 120 seconds, the program should **close the current session** and clear all session variables for that user. In that case, when the chat.js program tries to poll for any chat messages, it should receive the **401 status** response and redirect the browser to the login.php page.
15. Chat messages
    You should create a table in the database (db3322) to **store the chat messages**. Each message should have a unique ID, the sender's username (extracted from the email address of the sender), the time of receiving the message, and the message content. Here is a reference SQL statement for creating this table.

```
CREATE TABLE `message` (
  `msgid` int NOT NULL AUTO_INCREMENT,
  `time` bigint NOT NULL,
  `message` varchar(250) NOT NULL,
  `person` varchar(20) NOT NULL,
  PRIMARY KEY (`msgid`)
);
```

    (**Note: you can design your schema for storing the chat messages. The program will not retrieve messages that are received over one hour. It is not a requirement to remove those old messages; however, it would be nice to do so**.)

Testing platform

We shall run the server program in the **LAMP container set** and use Firefox and Chrome to test the programs.

Submission

Please finish this assignment before Friday April 10 23:59. Submit a compressed file (.zip) that contains the following files:

1. login.php
2. login.js
3. login.css
4. check.php
5. chat.php
6. chatmsg.php
7. chat.js
8. chat.css
9. Export a copy of the account table and the message table by using the Export function of the phpMyAdmin program. Submit the .sql files to Moodle. (Just in case you are using different database schemas for the tables, we can use these files to build the tables for testing your program.)

Grading Policy

Points	Criteria
7.0	**Login and Registration** - login.php, login.css, login.js, and check.php ▪ Correctly display the login and registration forms, and handle all user login and registration activities ▪ Check the user input (at client-side and server-side) ▪ Detect and handle those error situations (i.e., display appropriate messages or redirection) ▪ Handle the logout request ▪ Use the database to store account information ▪ Apply basic styling
3.0	**Session control** - login.php, chat.php, and chatmsg.php ▪ Correctly set up session control and allow authenticated users to access the chatroom ▪ Remove the session when logging out ▪ Correctly detect session timeout and redirect the user to the login.php page ▪ Correctly reject all requests when no existing active session
8.0	**Chat service** - chat.php, chat.js, chat.css, and chatmsg.php ▪ Correctly display the main UI of the chatroom and handle all user activities ▪ Allow users to send messages and logout ▪ Correctly load most recent messages in the past hour during initial startup, and load most recent message automatically since last polling ▪ Correctly display messages and support scrolling ▪ Detect and handle session timeout ▪ Use the database to store chat messages ▪ Apply the required styling
-4.0	Using any external libraries other than jQuery
-4.0	Did not provide the SQL files in the submission

Plagiarism

Plagiarism is a very serious academic offence. Students should understand what constitutes plagiarism, the consequences of committing an offence of plagiarism, and how to avoid it. ***Please note that we may request you to explain to us how your program is functioning as well as we may also make use of software tools to detect software plagiarism.***