**INTRODUCTION TO INTERNET PRORAMMING**

**KEVIN MUTISO C024/402139/2023**

**ASSIGNMENT 1: Developing a Secure Login System**

Quizes;

a) Develop a simple login system using HTML, JavaScript, and PHP/ASP.NET.

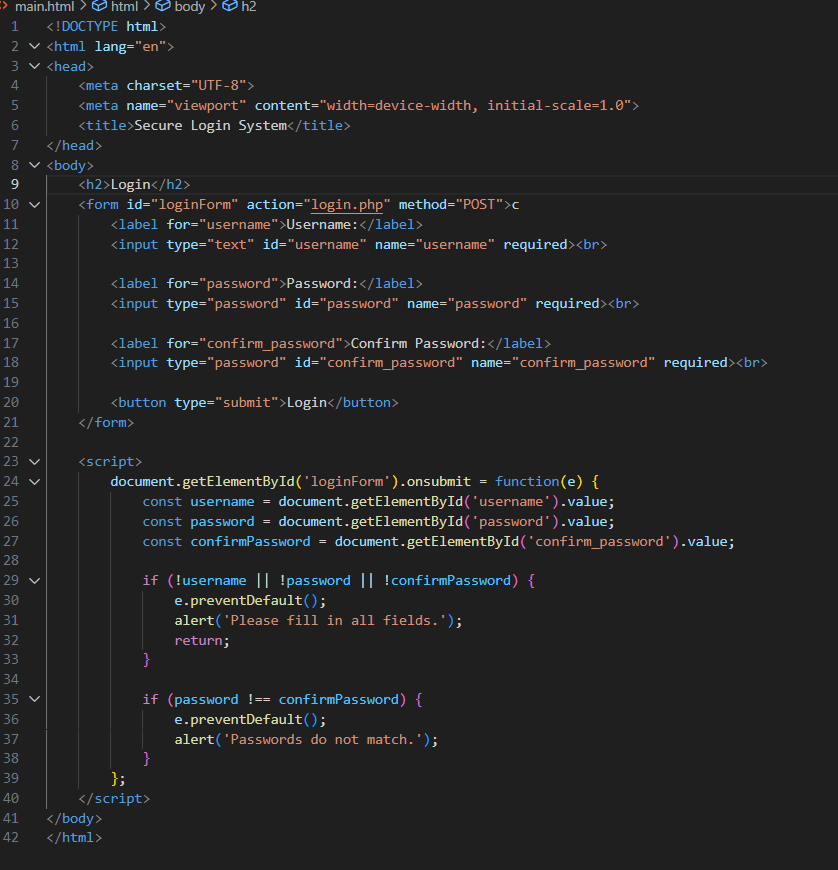
b) Implement session-based authentication.

c) Use password hashing and secure cookies for authentication.

d) Include a feature to prevent SQL Injection attacks.

e) Submit the code with explanationss

**Main.html**

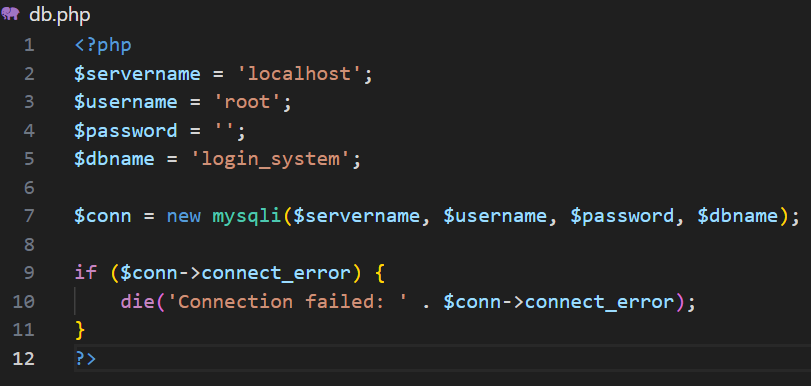
****

***Description***

This part of the code sets up the **HTML login form** and client-side validation using JavaScript.

* It has the html form setup for marking username, password and confirm password. It also has form action=”login.php” for sending data to php script.
* Method= “POST” ensure that the password is not visible to URL.
* Javascript validation- it captures form submission with onsubmit, checks that all fields are filled otherwise alerts the user. It ensures that password and confirm password match before sending data to server side.
* It uses e.preventDefault() to prevent the form from submitting if validation fails.

**<! Db.php**

****

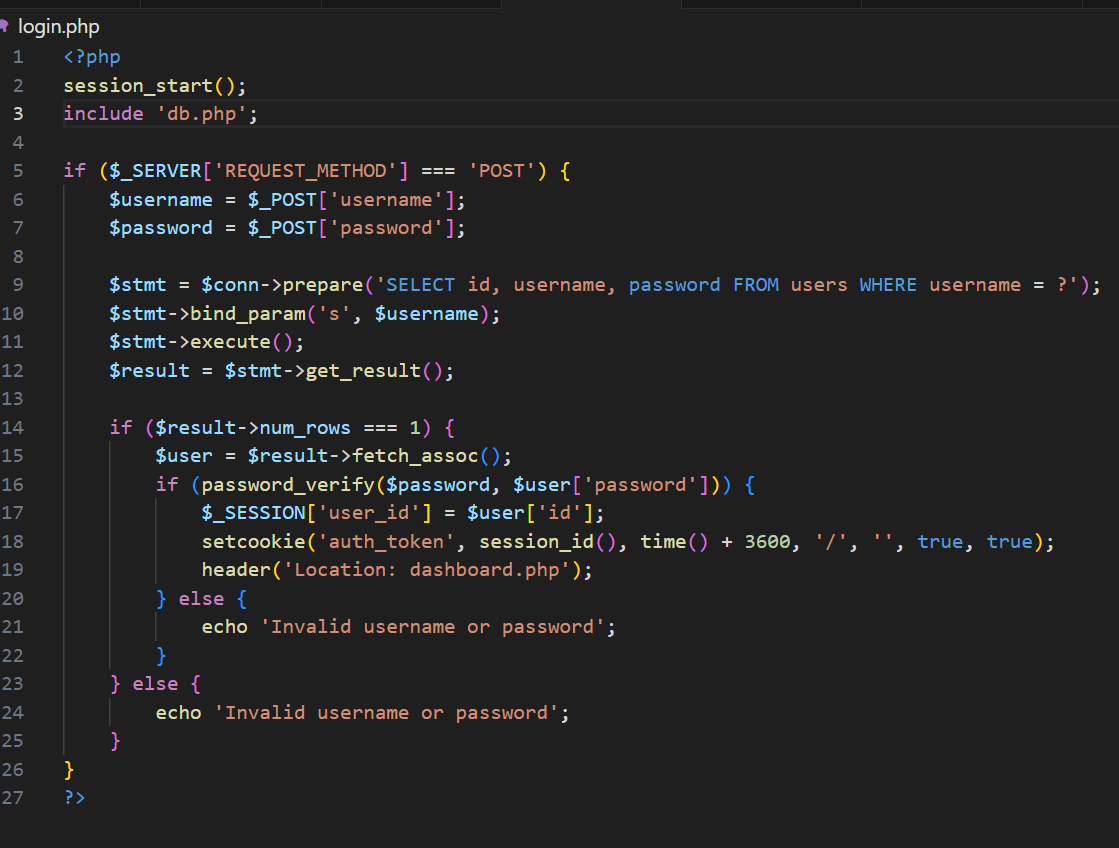
***Description***

This file establishes a connection to the MySQL database i.e

* $servername = ‘localhost’ – the database server runs locally
* $username = ‘root’ –the default MySQL root user
* $password = ‘’ –no password for the local root user.
* $dbname = ‘secure\_login’ – name of databse being linked to.

This also does databse connection and error handling. If the connection fails, this stops the script and shows an error message.

**<!-- login.php -->**



***Description***

* start the session- “session\_start ();” –this initializes a session so that we can store data in different pages.
* *Including database connection*- (include ‘db.php’) –this pulls in the bd.php script to access the $conn variable and connect to the database.
* if ($\_SERVER[‘REQUEST\_METHOD’] === ‘POST’){ -this ensures the script only runs if the form has been submitted via the
* POST method

*Capturing user input* - $username = $\_POST['username']; $password = $\_POST['password']; -this grabs the username and password the user submitted in the form.

* *preparing a SQL query* – prepared statements prevent SQL injection by treating user input as data, not code.

bind\_param('s', $username); -this binds the $username variable as a string.

$stmt->execute() –this runs the query.

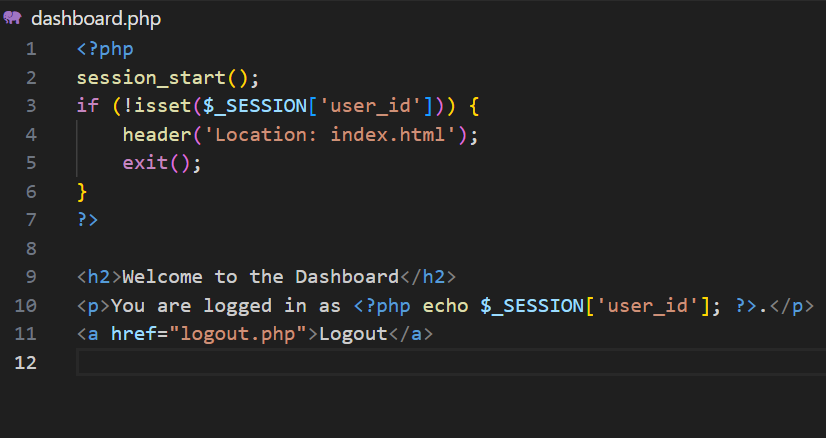
$result->get\_result() –this gets the results.

* *Checking existence of the user*- if ($result->num\_rows === 1) { -if one matching row is found , the username exists.
* $user = $result->fetch\_assoc(); -Fetching user data
* if (password\_verify($password, $user['password'])) { -this verifies the password whether it matches with the one in the database.
* *Setting session and secure cookie* –($\_SESSION['user\_id'])- this stores user’s ID for identifying the logged-in user.

-setcookie() –this creates secure HTTP.

* *Error handling*- echo 'Invalid username or password'; - this ensures if login fails, error message displays.

**<!-- dashboard.php -->**



***Description***

* session\_start(); - ensures the user has access to session variables set during login
* !isset($\_SESSION['user\_id']) – checks whether variable ‘user\_id is not set
* header('Location: index.html') –redirects any unauthorized user back to the login page.
* <h2>Welcome to the Dashboard</h2> - displays welcome message

<p>You are logged in as <?php echo $\_SESSION['user\_id']; ?>.</p> - shows the currently logged-in user’s ID from the session

* exit() -stops any further script execution.
* <a href="logout.php">Logout</a> -this takes the user to logout.php which ends the session
* **<!-- logout.php -->**



<?php

session\_start();

session\_destroy(); //this clears al the session data

setcookie('auth\_token', '', time() - 3600, '/');

header('Location: index.html');

?>

***Description***

* setcookie() -updates auth\_token cookie by setting its value to an empty string
* time() – 3600 -sets the cookie’s expiration time to one hour in the past making it expire.
* header('Location: index.html'); -this sends the user to login page after logging out.

The resultant login webpage resembles the following.

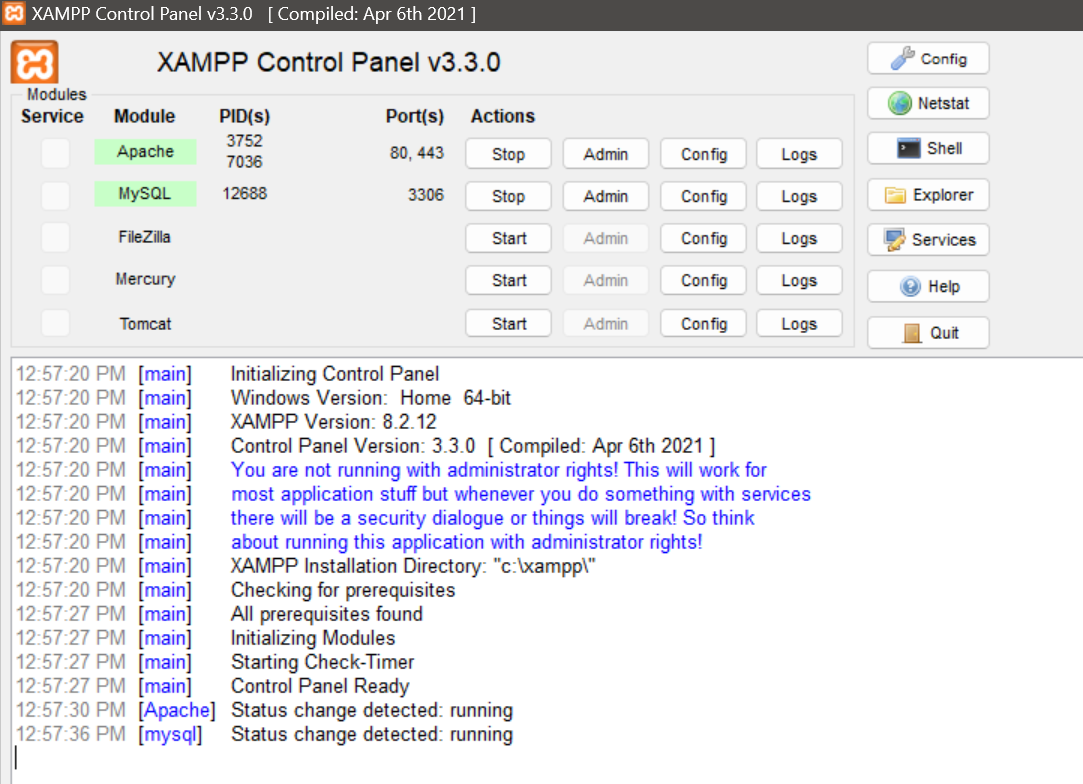
-the login webpage is able to take username and password

**How to run the code**

First you could run the code using a live server or alternatively follow the following steps.

**Step 1**

* Install xamp – its recommended since it entails apache, mysql and php required to set up the project. Once installed, install apache and mysql at the control panel.

****

**Step 2**

* Create the files using a code editor and place them in httdocs. i.e C:\xampp\htdocs\secure\_login\

**Step 3**

* Open in your browser <http://localhost/phpmyadmin>. Here create a database and call it secure\_login
* Run the following sql code to create a table where your elements will be saved

CREATE TABLE users

( id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(255) NOT NULL UNIQUE,

password VARCHAR(255) NOT NULL

);

set primary key.

**Step 4**

* Now run the application in your web browser http://localhost/secure\_login/main.html