Web Application Programming

Week 8: Building Dynamic Web Applications(Interaction with databases: User authentication and sessions management)

By Elubu Joseph - MSc.IS

Lecturer

Department of Information Technology

Kumi University

Email: josebulinda@gmail.com or jose@kumiuniversity.ac.ug

Agenda

1. Summary of previous lecture
2. Building Dynamic Web Applications(Interaction with databases:
    i. Build User Login system,
    ii. User authentication and authorization
    iii. Sessions management)

Summary of previous lecture

1. Building Dynamic Web Applications(
    i. Introduction(Definition, x-tics and examples of dynamic content)
    ii. Created a Database(wap) and table(users)
    iii. Established a Database Connection using connect.php
    iv. Performed CRUD Operations(insert, read, update and delete)

Build User Login system

A login system is a fundamental component of most web applications and online services that requires users to authenticate themselves before accessing certain resources, features, or content. It serves as a security and identity verification mechanism to ensure that only authorized individuals can interact with or view specific parts of a website or application, OpenAI. (2021)..

Key components and characteristics of a typical login system:

1. **User Credentials:** Users provide their unique identifying information to access the system. This typically includes a username or email and a password. Some systems may use other methods for authentication, such as biometrics or two-factor authentication (2FA).
2. **Authentication:** The system checks the provided credentials against stored user data to verify the user's identity. This often involves comparing the input password to a hashed or encrypted version stored in the system's database.

Build User Login system

Key components and characteristics of a typical login system:+

3. **Authorization**: After successful authentication, the system determines what the user is allowed to do and what resources they can access. Authorization rules are often defined based on the user's role or permissions.
4. **Sessions**: To maintain a user's logged-in state, the system creates a session or token that identifies the user and grants them access during a session's duration. This session information can be stored as a server-side session or as a client-side token (e.g., cookies or JSON Web Tokens).
5. **Password Security**: Storing and managing user passwords securely is critical. It's best practice to use password hashing techniques (e.g., bcrypt) and to salt the passwords to protect against data breaches.

Build User Login system

Key components and characteristics of a typical login system:++

6. **Security Measures**: A login system should implement security measures like account lockouts after repeated failed login attempts to prevent brute force attacks, CAPTCHA verification, and mechanisms to handle forgotten passwords (e.g., password reset or recovery).
7. **Logging and Monitoring**: Systems often log login attempts, successful and failed, for security and auditing purposes. This can help identify unauthorized access attempts and provide a record of user activity.
8. **User Management**: An admin interface for user management is typically included in a login system. Admins can create, update, and delete user accounts and manage their roles and permissions.

Build User Login system

Key components and characteristics of a typical login system:++

9. **Multi-Factor Authentication (MFA):** For added security, some login systems support MFA, which requires users to provide two or more forms of verification before granting access (e.g., something the user knows, something they have, and something they are).

A well-designed login system is crucial for maintaining the security and privacy of user accounts and data. It plays a central role in protecting against unauthorized access, data breaches, and ensuring that only legitimate users have access to restricted areas of a website or application, OpenAI. (2021).

Build User Login system+ Requirements

When building a user login system using PHP, you'll need several components and considerations to ensure it's secure, efficient, and user-friendly. Here's a list of requirements and key considerations:

1. **Web Development Tools**:
   i. A text editor or integrated development environment (IDE) for writing PHP code.
   ii. A web server (e.g., Apache, Nginx) with PHP support.
   iii. A database management system (e.g., MySQL, PostgreSQL) to store user data.
2. **Database Design**:
   i. Create a database schema to store user data, typically in tables, including fields for user ID, username, email, password, roles, and any additional user-related data.
   ii. Hash and salt passwords before storing them in the database for security (e.g., using bcrypt).

Build User Login system+ Requirements

List of requirements and key considerations:

3. **User Registration**:
   i. User registration form for collecting user information (e.g., username, email, password).
   ii. Validation of user input data to ensure data integrity and security.
   iii. Verify unique usernames and email addresses to prevent duplicate accounts.
   iv. Send email confirmation links for user account activation (optional but recommended).

4. **User Authentication**:
   1. A login form for users to enter their credentials.
   2. Securely hash and compare passwords during the login process.
   3. Handle forgotten passwords with a password reset or recovery process.

Build User Login system+ Requirements

List of requirements and key considerations:

5. **Session Management**:
   i. Use sessions to maintain user login state across multiple pages.
   ii. Implement mechanisms to control session timeout and user inactivity.
   iii. Secure session data to prevent session fixation and session hijacking.
6. **Access Control**:
   i. Define roles and permissions to control user access to different parts of the application.
   ii. Authorize users based on their roles and restrict access to unauthorized resources.

Build User Login system+ Requirements

List of requirements and key considerations:

7. Security Measures:

i. Implement security features like rate limiting and account lockout to protect against brute force attacks.

ii. Apply input validation and output encoding to prevent SQL injection, Cross-Site Scripting (XSS), and other security vulnerabilities.

iii. Implement measures to prevent Cross-Site Request Forgery (CSRF) attacks.

iv. Ensure HTTPS is used for secure data transmission (especially on the login page).

8.Multi-Factor Authentication (MFA):

i. Consider implementing MFA for enhanced security, using methods like SMS codes, email codes, or time-based one-time passwords (TOTP).

Build User Login system+ Requirements

List of requirements and key considerations:

9. **Logging and Monitoring**:
    i. Log login attempts, both successful and failed, for auditing purposes.
    ii. Implement monitoring and alerting for suspicious or unauthorized access.
10. **User Management**:
    i. Create an admin interface to manage user accounts, roles, and permissions.
    ii. Allow users to update their profile information and change their password.
11. **User Experience**:
    i. Design user-friendly login and registration forms with clear error messages.
    ii. Implement "Remember Me" functionality for persistent sessions.
    iii. Provide feedback on successful login or registration.

Build User Login system+ Requirements

List of requirements and key considerations:

12. **Testing and Validation**:
    i. Test your login system thoroughly, including edge cases and security testing.
    ii. Ensure that the system meets legal and regulatory requirements, such as GDPR compliance for user data.
13. **Documentation**:
    i. Document your login system's design, architecture, and usage for future reference and for other developers working on the project.

Remember that building a secure login system is an ongoing process, and you should stay up-to-date with best practices and security trends to continually improve your system's security. Regularly apply security patches and updates to your server and PHP environment.

Build User Login system+ Requirements+php pages needed

We need to create six pages to allow see how web apps with login systems work.

1. **index.php** – Landing point where login form is displayed
2. **connect.php** – contains database connection code and it is included all pages that need connection to the database.
3. **loign.php** – contain both login form creation and form data submission code
4. **logout.php** . Contains user session clearence code
5. **core.inc.php** – Contains server associative array $_SERVER[“SCRIPT_NAME”]- that helps to maintain address of the login form action, stored in the variable called **$current_file**.
6. **success.php** – the landing point after use successful login.
7. **forms.css** – contains some simple css for the login form.

Note that we had our database called wap containing users table.

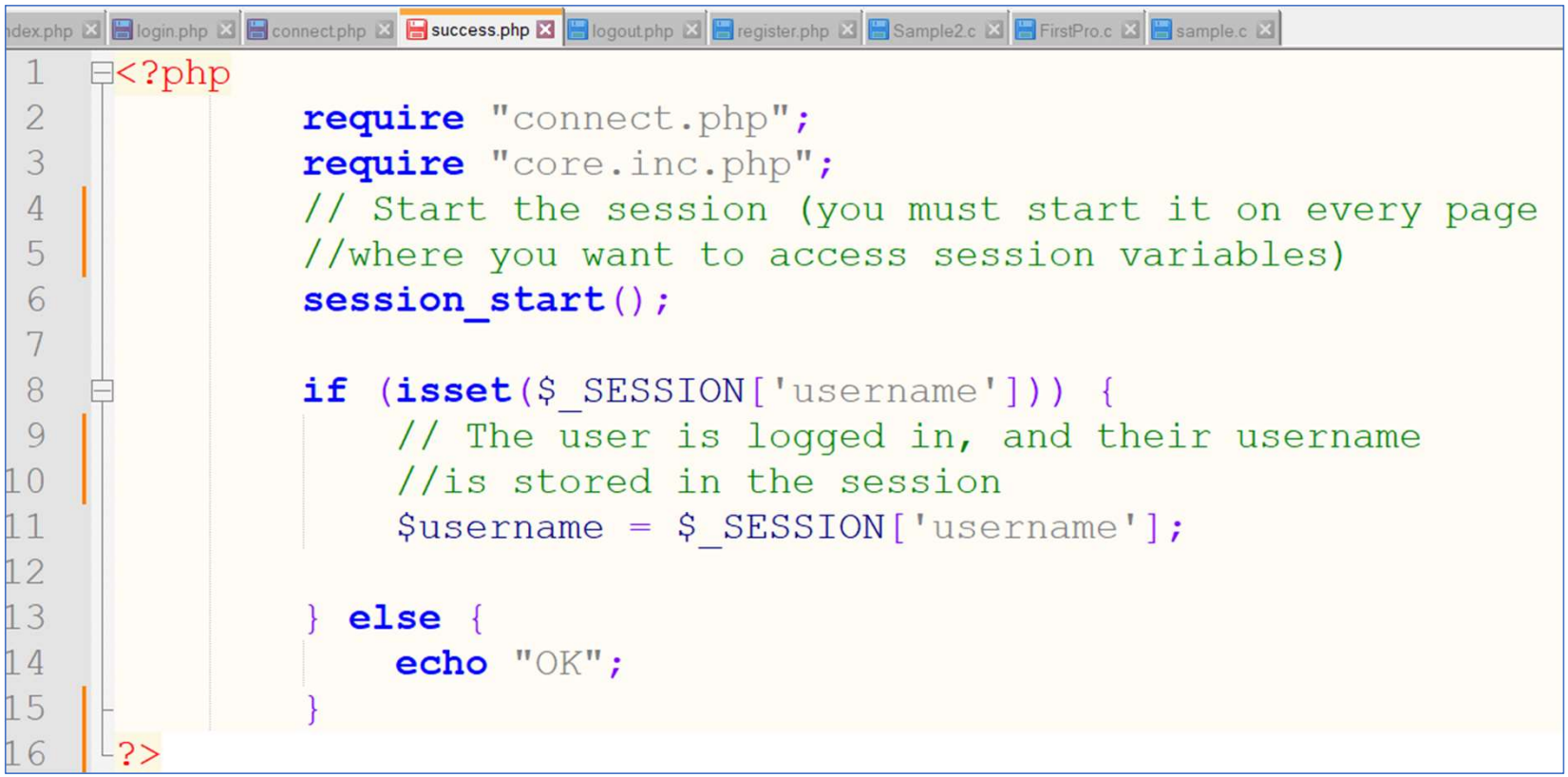
Connection to the database

We did went a head and created the file called **connect.php** where we entered connection script as see below

```php
<?php
	$server = "localhost";
	$username = "wap";
	$password = "wap";
	$database = "wap";
	$conn = new mysqli($server, $username, $password, $database);

	if ($conn->connect_error) {
		die("Connection failed: " . $conn->connect_error);
	}
	else{
	//echo 'connection successful.’;
	}
?>
```

Success.php page code



Logout.php code

dex.php | login.php | connect.php | success.php | logout.php | register.php | Sample2.c | FirstPro.c | sample.c

```php
<?php
    session_start(); // Start the session

    // Clear all session data and destroy the session
    session_unset();
    session_destroy();
    // Redirect to the login page after logout
    header("Location: index.php");
    exit();
?>
```

When the user click logout link, the **session_unset()** function un sets user session then **session_destroy();** destroys the session details.

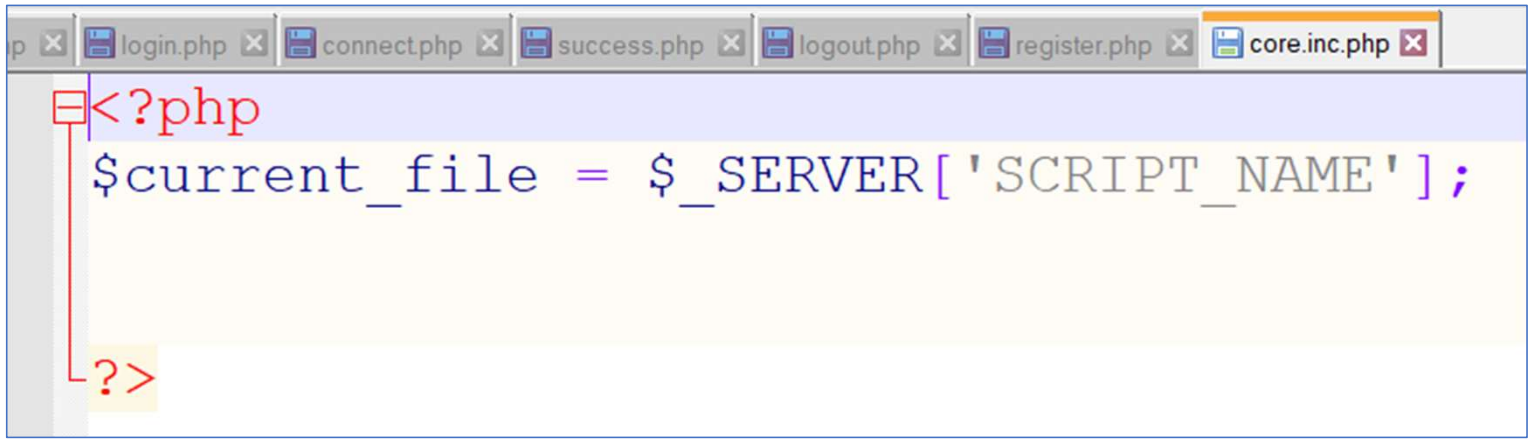
Then **header()** helps to direct which page to be displayed out logout.

index.php code

The php code in this file includes three calls to **connect.php**(allow index page to connect to the database), **core.inc.php** (maintains the login form action address dynamically) and finally **login.php** which allow the display of login form on index.php.

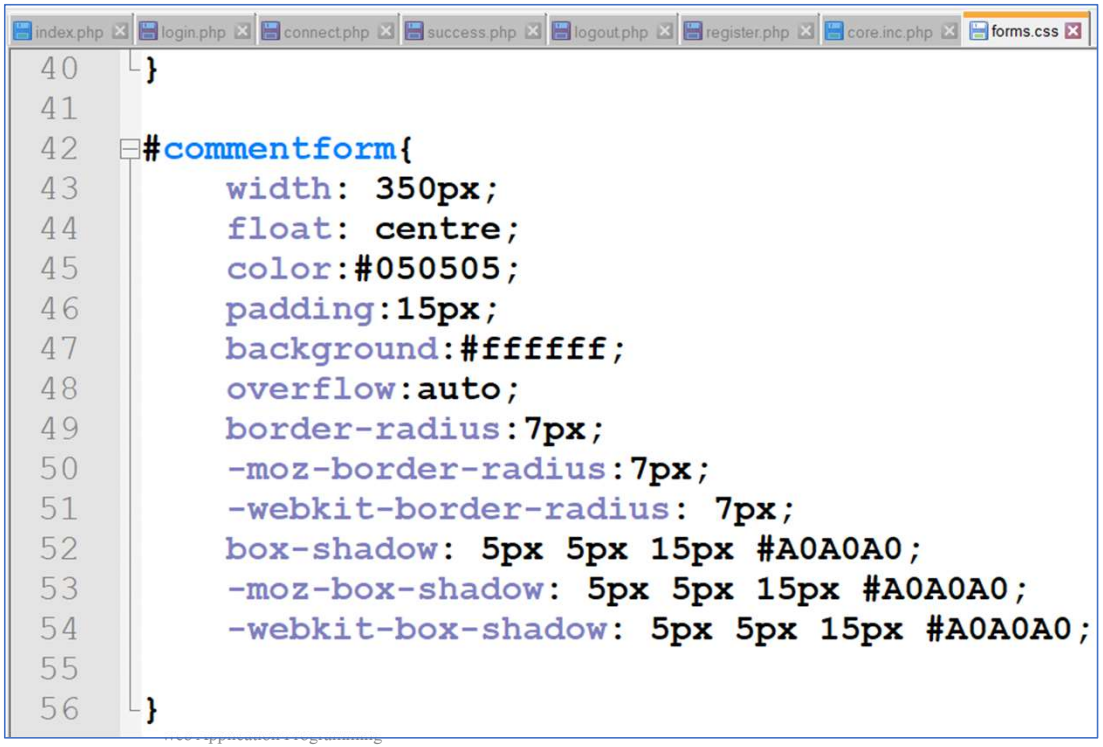
Core.inc.php

core.inc.php–Contains server associative array $_SERVER[“SCRIPT_NAME”]- that helps to maintain address of the login form action, stored in the variable called $current_file.



forms.css code

The css id call commentform has a number of properties whose values affect the login form in login.php page.



Database users table fields

To enable us register and login users, we created a table called users with the following structure and attributes.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	*None*		AUTO_INCREMENT	Change Drop More
2	firstname	varchar(50)	utf8mb4_general_ci		No	*None*			Change Drop More
3	lastname	varchar(50)	utf8mb4_general_ci		No	*None*			Change Drop More
4	email	varchar(80)	utf8mb4_general_ci		No	*None*			Change Drop More
5	username	varchar(25)	utf8mb4_general_ci		No	*None*			Change Drop More
6	password	varchar(32)	utf8mb4_general_ci		No	*None*			Change Drop More

Data insertion in to the table

Before we allow users to register, we need to build a fully functional login system and therefore we need to insert some manual data to allow us a achieve this.

Note! The rest of the table fields contain plain text except password field which must stored chipper text encrypted using md5() function. NOT as in the example below.

id	firstname	lastname	email	username	password
1	Okullo	James	james@gmail.com	james	james
2	Kadimba	James	james@gmail.com	james	1613f36d06342303f69469f5a1b29c21

One way Data Encryption

One way to hide highly sensitive information from the attackers is by encrypting the data. We do this using md5() encryption function that helps to change plain text to chipper text. A 32 character text.

For example if the password is 123, we should be able to change it to chipper text which looks like; - 934b535800b1cba8f96a5d72f72f1611 which we store in the database instead of 123.

Using md5() encryption function

As said above, md5() function gives us ability to do what we call one way encryption. Which means it can not be reversed.

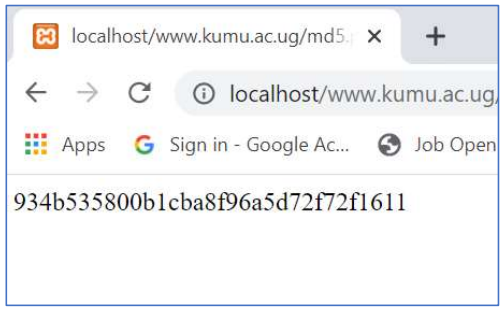
To encrypt anything using this function, you will need to place that inside the function parentheses e.g. md5(123). However, to do this better, we can store our plain text in a variable which we later pass

```php
<?php

$password = '2222';
$pass_has = md5($password);

echo $pass_has;

?>
```



Building a login system

In the login.php page, lets create login form

```
<html>
<body>
<form action=“ “ method="POST">
<table>
<tr>
<td width=100>UserName: </td>
<td><input type="text" name="username"></td>
</tr>
<tr>
<td>Password:</td>
<td><input type="password" name="password" ></td>
</tr><tr>
<td></td>
<td><input type="submit" name="submit" id="submit" class="button" value="Sign
In"/></td>
</tr>
</table>
</body></html>
```

Building a login system

Form action

Form action refer to the page where the form is located which in this case is **login.php**, however we would want this system to be dynamic to the extent that it always refer to a page where the login form is located although it is called onto another page e.g index.php.

Therefore we will need to use server specific associative array variable called **$_SERVER**[‘SCRIPT_NAME’] in another page that we will call **core.inc.php**

C: > xampp > htdocs > www.kumu.ac.ug > core.inc.php

```php
<?php
    $current_file= $_SERVER['SCRIPT_NAME'];

?>
```

```php
<?php
    $current_file= $_SERVER['SCRIPT_NAME'];
?>
```

Building a login system

Call core.inc.php into index.php

We now need to include core.inc.php into **index.php** in order for us to use the **$current_file** variable we created from **core.inc.php** page in login form action while in **index.php**. We also need to include **login.php** inside **index.php.**

So what this means is that, instead of the form action to be login.php while in index.php page, it dynamically changes the form action to index.php as seen below*: <form action="index.php" method="POST">* if one viewed the index.php page source code.

However, if one visited login.php directly, the form action in this case will dynamically change to login.php as below.; *<form action="login.php" method="POST">*

Building a login system

Finish the login form

Note the presence of echo **$current_file** variable inside the login.php page within the form action attribute.

Building a login system
Finish the login form+

```
<!DOCTYPE html>
<html>
<head><title></title>
<link rel="stylesheet" type="text/css" href="forms.css">
<link rel="stylesheet" type="text/css" href="major.css">
</head><body>
<h1>Sign In</h1><hr>
<form id="commentform" action="<?php echo $current_file; ?>" method="POST"><table>
<tr>
<td width=100>UserName: </td>
<td><input type="text" name="username"></td>
</tr><tr>
<td>Password:</td>
<td><input type="password" name="password" ></td>
</tr><tr>
<td></td>
<td><input type="submit" name="submit" id="submit" class="button" value="Sign In"/></td>
</tr>
</table></form></body></html>
```

Login page code Output

Building form data submission
Code in php

Now that we have finished to design the form, lets add in php code to submit data to the table.

To do this, we need to know: -

1. The name of the table
2. Login data field names in the table,

In our case users, username and password respectively

Building form data submission
Code in php+

```php
require 'connect.php';
session_start(); // Start the session
if(isset($_POST['username'])&&isset($_POST['password'])){

//Some variable will be defined here to hold the form data.

//Another if() function will be placed inside here.

}
```

Isset($_POST[]) function in php takes in associative array called $_POST[] that helps to check if the field data inside the array is posted by the form. Note! In this case we are checking if both the ‘username and ‘password’ are posted by the form.

session_start(); function allow user session management within the page it’s included.

Building form data submission
Code in php++

```php
if(isset($_POST['username'])&&isset($_POST['password'])){
    //Defining variables to be used
    $username = $_POST['username'];
    $password = $_POST['password'];
    $password_hash = md5($password);


//Another if() function will be placed inside here.

}
```

Now here we need to have some three variables to help us hold the values at the form text boxes

Building form data submission Code in php+++

Now we need to test if the form fields; username and password are not empty. To do this, we need to use if statement

```php
if(isset($_POST['username'])&&isset($_POST['password'])){
    //Defining variables to be used
    $username = $_POST['username'];
    $password = $_POST['password'];
    $password_hash = md5($password);

    if(!empty($username)&&!empty($password)){

        //Do something

    }else {
    echo 'Some field(s) is empty <br>';
    }

}
```

Building form data submission
Code in php+++

If the **$username** and **$password_hash** on the form is not empty then lets compare them with the username and password exiting in the database table by constructing the SQL query

```
if(!empty($username)&&!empty($password)){

      // Prevent SQL injection by using prepared statements
      $query = "SELECT * FROM users WHERE username = ? AND password = ? ";

      //if the above match then do something
}else {
   echo 'Some field(s) is empty <br>';
   }
```

Building form data submission Code in php++

```php
if(!empty($username)&&!empty($password)){

        // Prevent SQL injection by using prepared statements
        $query = "SELECT * FROM users WHERE username = ? AND password = ? ";

        if ($stmt = $conn->prepare($query)) {
           $stmt->bind_param("ss", $username, $password_hash);
           $stmt->execute();
           $stmt->store_result();
           $query_num_rows = $stmt->num_rows;

           if ($query_num_rows == 0) {
               echo 'Sorry!! Your Username or Password is incorrect, Please try again<br>';
           }
           else if ($query_num_rows == 1) {
                 // Fetch logged in user data here

        } else {
              echo 'You are not logged in';
           }
        $stmt->close();
       } else {
           echo "An error occurred. Please try again later.";
       }
}else {
    echo 'Some field(s) is empty <br>';
 }
```

Building form data submission Code in php++

```
  if ($query_num_rows == 0) {
         echo 'Sorry!! Your Username or Password is incorrect, Please try again<br>';
     }
else if ($query_num_rows == 1) {
       // Fetch logged in user data here
        $stmt->bind_result($user_id);
        $stmt->fetch();
        $_SESSION['id'] = $user_id;
       // Store user information in the session
       $_SESSION['username'] = $username;

       // Redirect to another page on successful login
        header("Location: success.php");
        exit(); // Terminate the script to ensure the redirection takes place

} else {
         echo 'You are not logged in';
     }
$stmt->close();
```

Building form data submission Code in php+++ complete code1

index.php | login.php | connect.php | success.php | logout.php | register.php

```php
<?php
    require 'connect.php';
    session_start(); // Start the session

    if (isset($_POST['username']) && isset($_POST['password'])) {
    // Defining variables to be used
    $username = $_POST['username'];
    $password = $_POST['password'];
    $password_hash = md5($password);

    if (!empty($username) && !empty($password)) {
        // Prevent SQL injection by using prepared statements
        $query = "SELECT id FROM users WHERE username = ? AND password = ?";
        if ($stmt = $conn->prepare($query)) {
            $stmt->bind_param("ss", $username, $password_hash);
            $stmt->execute();
            $stmt->store_result();
            $query_num_rows = $stmt->num_rows;

            if ($query_num_rows == 0) {
                echo 'Sorry!! Your Username or Password is incorrect, Please try again<br>';
```

Building form data submission Code in php+++ complete code2

```php
            } else if ($query_num_rows == 1) {
                $stmt->bind_result($user_id);
                $stmt->fetch();
                $_SESSION['id'] = $user_id;
                // Store user information in the session
                $_SESSION['username'] = $username;

                // Redirect to another page on successful login
                header("Location: success.php");
                exit(); // Terminate the script to ensure the redirection takes place
            } else {
                echo 'You are not logged in';
            }
            $stmt->close();
        } else {
            echo "An error occurred. Please try again later.";
        }
        // Close the database connection
        $conn->close();
    }
    else{
        echo "Some field(s) is empty";
    }
```

Building form data submission Code in php+++ complete code3

```
44          }
45      }else{
46          echo "Data not posted";
47      }
48
49  ?>
```

Building form +++ complete code4

```
51  <!DOCTYPE html>
52  <html>
53  <head><title></title>
54  <link rel="stylesheet" type="text/css" href="forms.css">
55  <link rel="stylesheet" type="text/css" href="major.css">
56  </head><body>
57  <h1>Sign In</h1><hr>
58  <form id="commentform" action="<?php echo $current_file; ?>" method="POST">
59  <table><tr>
60  <td width=100>UserName: </td>
61  <td><input type="text" name="username"></td>
62  </tr><td>Password:</td>
63  <td><input type="password" name="password" ></td>
64  </tr>
65  <tr><td></td>
66  <td><input type="submit" name="submit" id="submit" class="button" value="Sign In"/></td>
67  </tr></table>
68  </body></html>
```

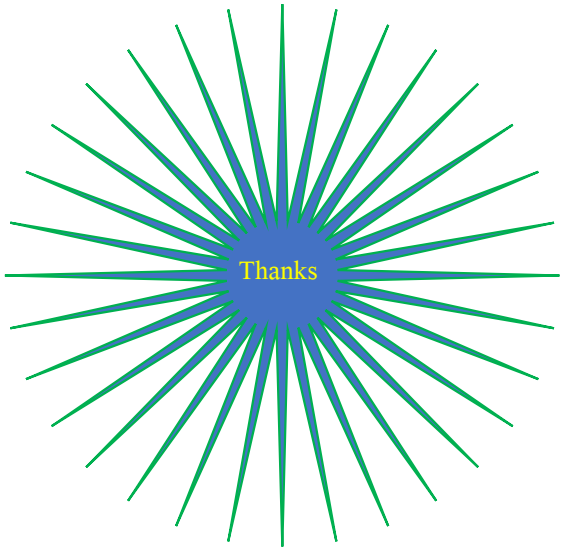
The code above are all in login.php file.

Summary

Agenda

1. Building Dynamic Web Applications(Interaction with databases:
   i. Build User Login system,
   ii. User authentication and authorization
   iii. Sessions management)

Thank you for Listening



References

JavaScript Data Types. OpenAI Knowledge Base. OpenAI. (2021, September 22)..