Absolutely! Here's a clear breakdown of your **PHP login script and HTML/CSS login form** for the **IPHS Campus** system:

**✅ PHP Part (Top Section)**

**1. Start session and connect to the database**

session\_start();

include("db\_connect.php");

* session\_start() initializes the session so you can store user info after login.
* db\_connect.php includes your database connection credentials.

**2. Handle POST Request (Form Submission)**

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

* This block only runs when the form is submitted using POST.

**3. Get and sanitize user input**

$input = trim($\_POST['username\_email']);

$password = $\_POST['password'];

$remember = isset($\_POST['remember']);

* username\_email can be a username or an email.
* trim() removes extra spaces.
* Checks if "Remember Me" is checked.

**4. Securely prepare and run SQL query**

$query = $conn->prepare("SELECT \* FROM users WHERE fullname = ? OR email = ?");

$query->bind\_param("ss", $input, $input);

$query->execute();

$result = $query->get\_result();

$user = $result->fetch\_assoc();

* Prepared statements prevent SQL injection.
* Checks both fullname and email fields.

**5. Verify password and create session**

if ($user && password\_verify($password, $user['password'])) {

session\_regenerate\_id(true);

$\_SESSION['user\_id'] = $user['id'];

$\_SESSION['role'] = htmlspecialchars($user['role'], ENT\_QUOTES, 'UTF-8');

* Checks if user exists and password is correct.
* session\_regenerate\_id(true) avoids session hijacking.
* Role is saved securely using htmlspecialchars() to prevent XSS.

**6. Set "Remember Me" cookie (optional)**

if ($remember) {

setcookie("remember\_me", $user['id'], time() + (86400 \* 30), "/");

}

* Stores user ID in a cookie for 30 days if "Remember Me" is selected.

**7. Role-based redirection**

$dashboard = [

"admin" => "admin/dashboardadmin.php",

"teacher" => "teacher/dashboardteacher.php",

"student" => "student/dashboardstudent.php"

];

header("Location: " . ($dashboard[$user['role']] ?? 'dashboard.php'));

exit();

* Redirects to the correct dashboard depending on the user’s role.
* Defaults to dashboard.php if role is not matched.

**✅ HTML + CSS (Login Form UI)**

**Login Form Design Highlights**

* Clean glassmorphism effect with backdrop-filter: blur().
* Responsive design with media query for smaller screens.
* Bootstrap and Font Awesome used for layout and icons.

**HTML Form Inputs**

<input type="text" name="username\_email" placeholder="Username or Email">

<input type="password" name="password" id="password" placeholder="Password">

* Accepts both email and username for login.

**Show/Hide Password Toggle**

<span toggle="#password" class="fas fa-eye toggle-password"></span>

* Eye icon toggles password visibility.

document.querySelector('.toggle-password').addEventListener('click', function () {

const password = document.getElementById('password');

const type = password.getAttribute('type') === 'password' ? 'text' : 'password';

password.setAttribute('type', type);

this.classList.toggle('fa-eye');

this.classList.toggle('fa-eye-slash');

});

* This JavaScript switches the password input between text and password.

**"Remember Me" Checkbox**

<input type="checkbox" name="remember">

* If checked, a cookie will be stored.

**Styling Notes**

* Responsive .login-container with a max width.
* Rounded inputs and buttons for modern look.
* Works well on both desktop and mobile.

**🔒 Security Features**

| **Feature** | **Purpose** |
| --- | --- |
| session\_regenerate\_id(true) | Prevent session hijacking |
| password\_verify() | Secure password comparison from DB hash |
| htmlspecialchars() | Avoid XSS when saving/displaying roles |
| Prepared SQL statements | Prevent SQL injection |

If you want, I can also help you:

* Auto-login via cookie (full remember-me logic).
* Add login attempt limits (to prevent brute force).
* Store more session info (like name, profile pic).
* Add CSRF protection.

Let me know what you’d like next!