📌 Task 2: Login & Signup System using PHP & MySQL

🎯 Objective:
Develop a **secure login and signup system** using **PHP and MySQL** with a **user-friendly interface** for authentication.

🚀 Key Features

✅ **User Signup** → Register with a **username, email, and password**.
✅ **User Login** → Authenticate using **email/username and password**.
✅ **Password Hashing** → Store **hashed passwords** for security.
✅ **Error Handling** → Show **appropriate error messages**.
✅ **Session Management** → Maintain login sessions.
✅ **Database Integration** → Connect **PHP with MySQL**.

🛠️ Technologies to Use

Frontend

- **HTML** → Structure the **Signup & Login** forms
- **CSS** → Style the forms for a **clean & responsive UI**

Backend

- **PHP** → Handle form submissions, authentication, and session management

Database

- **MySQL** → Store **user credentials securely**

📌 Development Steps

1☐ Create a MySQL Database

✔ **Database Name:** `user_auth`
✔ **Table:** **`users`**

- `id` (INT, PRIMARY KEY, AUTO_INCREMENT)
- `username` (VARCHAR, UNIQUE)
- `email` (VARCHAR, UNIQUE)
- `password` (VARCHAR)

✓ **Ensure data validation** (unique email & username).

2☐ Create Signup & Login Forms (Frontend)

✓ Signup Form

- Fields: **Username, Email, Password, Confirm Password**
- Use **placeholders & labels** for better UX

✓ Login Form

- Fields: **Username/Email, Password**

✓ Styling (CSS)

- Simple & clean **form design**
- Use **responsive layout**

3☐ Handle Form Submissions with PHP

✓ Signup Process

1☐ Validate inputs (**empty fields, email format, password match**).
2☐ Check if **email/username already exists**.
3☐ **Hash passwords** using `password_hash()`.
4☐ Insert user data into the **MySQL database**.
5☐ Show **error/success messages**.

✓ Login Process

1☐ Validate input fields (**empty fields**).
2☐ Retrieve the **hashed password** from the database.
3☐ Verify the password using `password_verify()`.
4☐ Start a **session** and redirect users to the **dashboard**.
5☐ Show **error message if login fails**.

4☐ Ensure Data Security

✓ Password Security

- Hash passwords using `password_hash()`.
- Verify passwords using `password_verify()`.

✓ Input Validation

- Ensure **correct email format**.
- Prevent **SQL Injection** (Use prepared statements).

✓ Session Management

- Start a session on login (`session_start()`).
- Store **user details securely**.

5☐ Test the Forms

✓ Verify **successful user signup**.
✓ Test **login with correct and incorrect credentials**.
✓ Check if **users can access a protected page after login**.
✓ Ensure **error messages display properly**.

6☐ Basic Error Handling

✓ Signup Errors

- "Username or Email already exists"
- "Passwords do not match"
- "All fields are required"

✓ Login Errors

- "Incorrect username/email or password"
- "User does not exist"

📌 Deliverables

✓ **Fully functional Login & Signup system**.
✓ **Secure authentication with password hashing**.
✓ **Error messages for better UX**.
✓ **User session management** after login.
✓ **Clean UI with basic CSS styling**.

Would you like a **database schema diagram** or **PHP folder structure recommendations**?

Deadline Compliance

- **Restriction**: **Submit the project within 7 days** from the start date.
- **Reason**: Meeting deadlines is crucial in the real-world software development environment. This restriction helps students practice **time management** and **task prioritization**. In professional settings, tight deadlines are often the norm, and learning to meet them without compromising quality is an essential skill.

Learning Outcome: Students will learn to manage their time effectively, complete projects under pressure, and **deliver results on time**, which are all important skills in the workplace.