SparkINN Novice Solution Pvt. Ltd

Week 2 PHP Weekend Task: Building a Simple User Registration System

---

Objective:
In this weekend task, you will apply the concepts learned in the PHP course to build a **Simple User Registration System**. This task will help you consolidate your understanding of PHP syntax, forms, handling user input, and basic database operations. You will create a system where users can register, log in, and view their profile. This task is designed to cover essential PHP topics such as form handling, sessions, and basic MySQL operations.

Task Overview:

You will be required to:

1. Set Up the Environment:

- Install PHP and set up a local web server (e.g., XAMPP, WAMP, or MAMP).
- Ensure that PHP, MySQL, and phpMyAdmin are correctly installed and configured.

2. Create the Project Structure:

- Create a new directory named `user_registration_system` in your web server's root directory.
- Inside the `user_registration_system` directory, create the following folders and files:
  - **includes/**: For reusable PHP files (e.g., database connection, utility functions).
  - **css/**: For stylesheets.
  - **register.php**: For user registration.
  - **login.php**: For user login.
  - **profile.php**: For viewing user profile.
  - **logout.php**: For logging out.

3. Database Setup:

- **Problem**: Create a MySQL database named `user_system` with a table `users` for storing user information.
  - **Table Schema**:
    - `id` (INT, auto-increment, primary key)
    - `username` (VARCHAR, unique)
    - `password` (VARCHAR)
    - `email` (VARCHAR, unique)

4. User Registration Form:

- **Problem**: Create a PHP registration form in `register.php` that allows users to create an account.
    - **Form Fields**:
        - Username
        - Password (with confirmation)
        - Email
- **Hint**: Use PHP to handle form submission and validate user input. Hash passwords before storing them in the database.

5. User Login Form:

- **Problem**: Create a PHP login form in `login.php` that allows users to log in to their accounts.
    - **Form Fields**:
        - Username
        - Password
- **Hint**: Use PHP to validate user credentials and start a session upon successful login.

6. User Profile Page:

- **Problem**: Create a `profile.php` page that displays the user's profile information.
    - Ensure that users are only able to view their own profile after logging in.
- **Hint**: Use PHP sessions to manage user authentication and display user data.

7. Implement Logout Functionality:

- **Problem**: Create a `logout.php` file to handle user logout and session destruction.

8. Handling Form Validation and Error Messages:

- **Problem**: Improve form validation by adding checks for valid email format, username length, and password strength.
    - Display appropriate error messages for invalid inputs.
- **Hint**: Use PHP's `filter_var()` function for email validation and regular expressions for password strength.

Submission:

- Zip your `user_registration_system` folder and submit it.
- Ensure that all functionalities (registration, login, profile view, logout) are working as expected.
- Include a **README** file with instructions on how to run the application.