Day - Individual Project : “LeBlog”

```
<?php
/*
Exercise: Mini Blog in PHP

**1. Set Up Your Environment**

Docker.

**2. Create Database**

Create a MySQL database named "mini_blog". Inside this database, create a table named "pos
ts" with the following fields:

* id: primary key, auto-increment
* title: varchar(255)
* content: text
* author: varchar(100)
* created_at: datetime

**3. Connect Database with PHP - db_connect.php**

In this script, write a function to establish a connection to your MySQL database. This sc
ript will be included in all other scripts where a
database connection is required.

**4. Create Posts - new_post.php**

Create a form with fields for `title`, `content`, and `author`. When this form is submitte
d, it should call a PHP function that inserts a new
record into the "posts" table in your database. After successful
creation, redirect the user to the "post.php" page to view all posts.

**5. Read Posts - post.php**

Create a script that fetches all the posts from the database and displays them on the pag
e. Each post should display the `title`, `content`,
`author`, and `created_at` fields. Each post should also have 'Edit' and 'Delete' buttons
 associated with them. When clicked, these buttons
should lead to "edit_post.php" and call the delete function,
respectively.

**6. Update Posts - edit_post.php**

This page should initially display a form with pre-filled data of the
```

```
post to be edited, which you can get using the post `id` passed from the "post.php" page.
 The form should accept `title`, `content`, and
`author` as input. When the form is submitted, it should call a PHP
function that updates the specified post in the database using the
post `id`. After successful update, redirect the user back to the
"post.php" page.

**7. Delete Posts**

In the "post.php" page, the 'Delete' button associated with each post
 should call a PHP function that deletes the respective post. This
 function should accept the `id` of the post to be deleted. After
 successful deletion, the post should immediately disappear from the
"post.php" page.

Bonus
**8. Frontend**

Use HTML/CSS (Bootstrap or plain CSS) to design the frontend. The
"new_post.php" and "edit_post.php" pages should have forms for creating
and editing posts, respectively. The "post.php" page should display a
list of all existing posts.

**9. Error Handling**

Add code in your scripts to handle possible errors. For instance, you
could display an error message if a post cannot be created, read,
updated, or deleted.

Remember to follow good coding practices, such as commenting your code
and using meaningful variable names. By the end of this exercise, you
should have a better understanding of how CRUD operations work in PHP.
*/
```