**Step 10: Implementing the Comment Model (*app/models/Comment.php*)**

We will start by adding code to the comment model. Let’s go ahead and do that.

**Comment.php Model**

<?php

namespace app\models;

// Include the bootstrap file for global setup

require\_once BASE\_DIR . '/bootstrap.php';

class Comment {

    private $db;

    // Initialize connection in constructor

    public function \_\_construct() {

        $this->db = new \Database; // Create a Database object for connection

    }

    public function createComment($postId, $content) {

        // Prepare a SQL statement

        $stmt = $this->db->conn->prepare("INSERT INTO comments (post\_id, content) VALUES (?, ?)");

        // Execute the statement with the post ID and the content

        $stmt->execute([$postId, $content]);

    }

    public function getComments($postId) {

        // Prepare a SQL statement

        $stmt = $this->db->conn->prepare("SELECT \* FROM comments WHERE post\_id = ?");

        // Execute the statement with the post ID

        $stmt->execute([$postId]);

        // Return the result as an object

        return $stmt->fetchAll(\PDO::FETCH\_OBJ);

    }

    public function getCommentById($id) {

        // Prepare a SQL statement

        $stmt = $this->db->conn->prepare("SELECT \* FROM comments WHERE id = ?");

        // Execute the statement with the ID

        $stmt->execute([$id]);

        // Return the result as an object

        return $stmt->fetch(\PDO::FETCH\_OBJ);

    }

    public function updateComment($id, $data) {

        // Prepare a SQL statement

        $stmt = $this->db->conn->prepare("UPDATE comments SET content = ? WHERE id = ?");

        // Execute the statement with the new content and the ID

        return $stmt->execute([$data['content'], $id]); // Return the result

    }

    public function deleteComment($id) {

        // Get the comment

        $comment = $this->getCommentById($id);

        // Check if the comment exists

        if ($comment) {

            // Prepare a SQL statement

            $stmt = $this->db->conn->prepare("DELETE FROM comments WHERE id = ?");

            // Execute the statement with the ID

            $stmt->execute([$id]);

        }

    }

}

?>

***app/controllers/CommentController.php***

Let’s update the ***app/controllers/CommentController.php***.

<?php

namespace app\controllers;

require\_once BASE\_DIR . '/app/models/Comment.php';

class CommentController {

    private $model;

    public function \_\_construct() {

        $this->model = new \app\models\Comment;

    }

    public function store($data) {

        $postId = $data['post\_id'];

        $content = $data['content'];

        // Store the comment in the database

        $this->model->createComment($postId, $content);

        header('Location: /posts/show/' . $postId);

    }

    public function edit($id) {

        // Get the comment from the database

        $comment = $this->model->getCommentById($id);

        // Check if the comment exists

        if (!$comment) {

            die('Comment not found');

        }

        // Create an instance of the Post model

        $postModel = new \app\models\Post;

        // Fetch the post related to the comment

        $post = $postModel->getPost($comment->post\_id);

        // Check if the post exists

        if (!$post) {

            die('Post not found');

        }

        // Include the edit form view, pass both comment and post

        require BASE\_DIR . '/app/views/admin/comments/edit.php';

    }

    public function update($data, $id) {

        $data = $\_POST;

        $result = $this->model->updateComment($id, $data); // Update the comment

        if (isset($data['post\_id']) && !empty($data['post\_id'])) {

            $postId = $data['post\_id'];

            $\_SESSION['message'] = 'Comment updated successfully!.';

            header('Location: /admin/posts/show/' . $postId);

        } else {

            // Handle the error or redirect to a default page

            header('Location: /admin/posts');

        }

    }

 public function delete($id) {

      // Delete the comment

     $this->model->deleteComment($id);

     // Redirect to the related post or a default page if successful

          header('Location: /admin/posts');

    }

    }

}

Now were still not able to create a comment for a post. This is because I decided that there is no need for an admin to do that. We will allow the comment to be created on the frontend.

So, what we will do here is that we will update the view file ***admin/posts/show.php***. It will loop over the comments created by a user on the frontend and the admin will be able to either edit or delete the comment. This is what I mean.

***admin/posts/show.php***

<div>

    <!-- View all Post Button -->

    <div style="text-align: right; margin-bottom: 20px;">

        <a href="/admin/posts/">View Posts</a>

    </div>

    <h2><?php echo htmlspecialchars($post->title, ENT\_QUOTES); ?></h2>

    <p><?php echo htmlspecialchars($post->content, ENT\_QUOTES); ?></p>

    <?php if (!empty($comments)): ?>

    <h3>Comments</h3>

    <?php foreach ($comments as $comment): ?>

        <div class="mb-2">

            <p><?php echo htmlspecialchars($comment->content, ENT\_QUOTES); ?></p>

            <a href="/admin/comments/edit/<?php echo $comment->id; ?>" class="btn btn-warning btn-sm">Edit</a>

            <form action="/admin/comments/delete/<?php echo $comment->id; ?>" method="POST" style="display: inline;">

                <input type="hidden" name="\_method" value="DELETE">

                <button type="submit">Delete</button>

                </form>

        </div>

    <?php endforeach; ?>

    <?php else: ?>

        <p>No comments found in this post.</p>

    <?php endif; ?>

</div>

**AdminController: Show Method**

Then Update the ***show method*** within the AdminController like this.

require\_once BASE\_DIR . '/app/models/Comment.php';

// Read: Display a single post

    public function show($id) {

        // Fetch the post with the given ID

        $post = $this->model->getPost($id);

        if ($post === null) {

            // Post not found, display 404 page

            require BASE\_DIR . '/app/views/404.php';

            return;

        }

         // Create an instance of the Comment model

         $commentModel = new \app\models\Comment;

         // Fetch the comments for the post

         $comments = $commentModel->getComments($id);

        // Load the view for displaying the single post

        require BASE\_DIR . '/app/views/admin/posts/show.php';

    }

So, what is happening here is that we are creating an object ***$commentModel.*** This object is used to access the method ***getComments*** where we pass an ***$id*** of the comment related to the post then we define a ***$comments*** variable to store the returned data.

So far no comment will be returned. As we can inspect this by passing the ***<?php if (!empty($comments)): ?>*** code.

**Update the Routes File (*public/index.php*)**

Since we have the end points ***/admin/comments/edit/*** and ***'admin/comments/update/*** that enables us to edit and update a comment from the admin side. Let’s create the routes to the controller methods edit and update in the comment controller.

$routes = [

 'GET' => [

 // Admin Comment Routes

        'admin/comments/edit/([0-9]+)' => ['controller' => '\app\controllers\CommentController', 'method' => 'edit'],

],

'POST' => [

// Admin Comment Routes

         'admin/comments/update/([0-9]+)' => ['controller' => '\app\controllers\CommentController', 'method' => 'update'],

 'admin/comments/delete/([0-9]+)' => ['controller' => '\app\controllers\CommentController', 'method' => 'delete'],

 ]

];

Lastly let’s create the view file (***admin/comments/edit.php***)

<?php

if (!$comment) {

    echo "Comment not found";

    return;

}

?>

<form method="POST" action="/admin/comments/update/<?php echo $comment->id; ?>">

    <input type="hidden" name="id" value="<?php echo $comment->id; ?>">

    <input type="hidden" name="post\_id" value="<?php echo $comment->post\_id; ?>">

    <label for="content">Comment Content</label>

    <textarea id="content" name="content"><?php echo htmlspecialchars($comment->content, ENT\_QUOTES); ?></textarea>

    <button type="submit">Update Comment</button>

</form>

**Updating The PostController**

To ensure users can add a comment from the frontend side let’s go ahead and update the PostController we created earlier. **Remember its only the admin who will edit and update a comment.**

**app/controllers/PostController.php**

<?php

namespace app\controllers;

require\_once BASE\_DIR . '/app/models/Post.php';

require\_once BASE\_DIR . '/app/models/Comment.php';

class PostController {

    private $model;

    public function \_\_construct() {

        $this->model = new \app\models\Post;

    }

    // Display all posts

    public function index() {

        $posts = $this->model->getPosts();

        require BASE\_DIR . '/app/views/posts/index.php';

    }

    // Display a single post

    public function show($id) {

        // Retrieve the specific post by its ID using the Post model

        $post = $this->model->getPost($id);

        // Instantiate the Comment model to interact with the comments data

        $commentModel = new \app\models\Comment;

        // Retrieve all comments associated with the post

        // This assumes that the getComments method fetches comments based on the post's ID

        $comments = $commentModel->getComments($id);

        // Load the view file that displays the details of the post

        // This view will show the post and its associated comments

        require BASE\_DIR . '/app/views/posts/show.php';

    }

}

?>

**Create the Frontend view Files (Index & Show)**

This view files will be involved in displaying the posts in the frontend side. Let’s start by updating the index.php view file.

***app/views/posts/index.php***

<?php foreach ($posts as $post): ?>

        <div style="margin-bottom: 20px; margin-top: 20px;">

            <div>

                <h2><?php echo htmlspecialchars($post->title, ENT\_QUOTES); ?></h2>

                <p><?php echo htmlspecialchars($post->content, ENT\_QUOTES); ?></p>

                <div>

                  <a href="/posts/show/<?php echo $post->id; ?>">View</a>

                </div>

            </div>

        </div>

    <?php endforeach; ?>

</div>

Here we loop over our posts from the database then we print them out using object-property access ***$post->title*** and ***$post->content.***

***app/views/posts/show.php***

<!-- show.php -->

<div>

    <div>

        <div>

            <h2><?php echo htmlspecialchars($post->title, ENT\_QUOTES); ?></h2>

        </div>

        <div>

            <p><?php echo htmlspecialchars($post->content, ENT\_QUOTES); ?></p>

        </div>

        <div>

            <h3>Comments</h3>

            <?php foreach ($comments as $comment): ?>

                <div style="margin-bottom: 20px;">

                    <p><?php echo htmlspecialchars($comment->content, ENT\_QUOTES); ?></p>

                </div>

            <?php endforeach; ?>

            <form action="/comments/store" method="post" style="margin-top: 20px;">

                <div style="margin-bottom: 20px;">

                    <label for="content">Content:</label>

                    <textarea id="content" name="content"></textarea>

                </div>

                <input type="hidden" name="post\_id" value="<?php echo $post->id; ?>">

                <button type="submit">Submit</button>

            </form>

        </div>

    </div>

</div>

We loop over the comments in the database and show them in our view file ***show***.***php***. When adding a new comment. We route to the ***“/comments/store”.*** This endpoint will call the Commentcontroller method store.

 'comments/store' => ['controller' => '\app\controllers\CommentController', 'method' => 'store'],

Which will store our comments in the database table comments. This is the method that does that.

    public function store($data) {

        $postId = $data['post\_id'];

        $content = $data['content'];

        // Store the comment in the database

        $this->model->createComment($postId, $content);

        header('Location: /posts/show/' . $postId);

    }

It captures the post id of the post we are currently commenting and the content of the comment then routes us back to the show view file.

**Editing and Deleting Comments on the Admin Side.**

We can now edit and delete comments on the admin side. This is because a user can now comment from the frontend side including the admin. So, the admin can see the comments and do any modification to them.