Made by :

Muhammad Ashraf Malabali, Jefferson Ruben Sulaiman, Muhammad Dzaki Abrar

---

"Our project aims to create a social media website that took inspiration from Instagram, but instead of posting pictures of our life, it's pictures of food and recipes."

1. Database Table

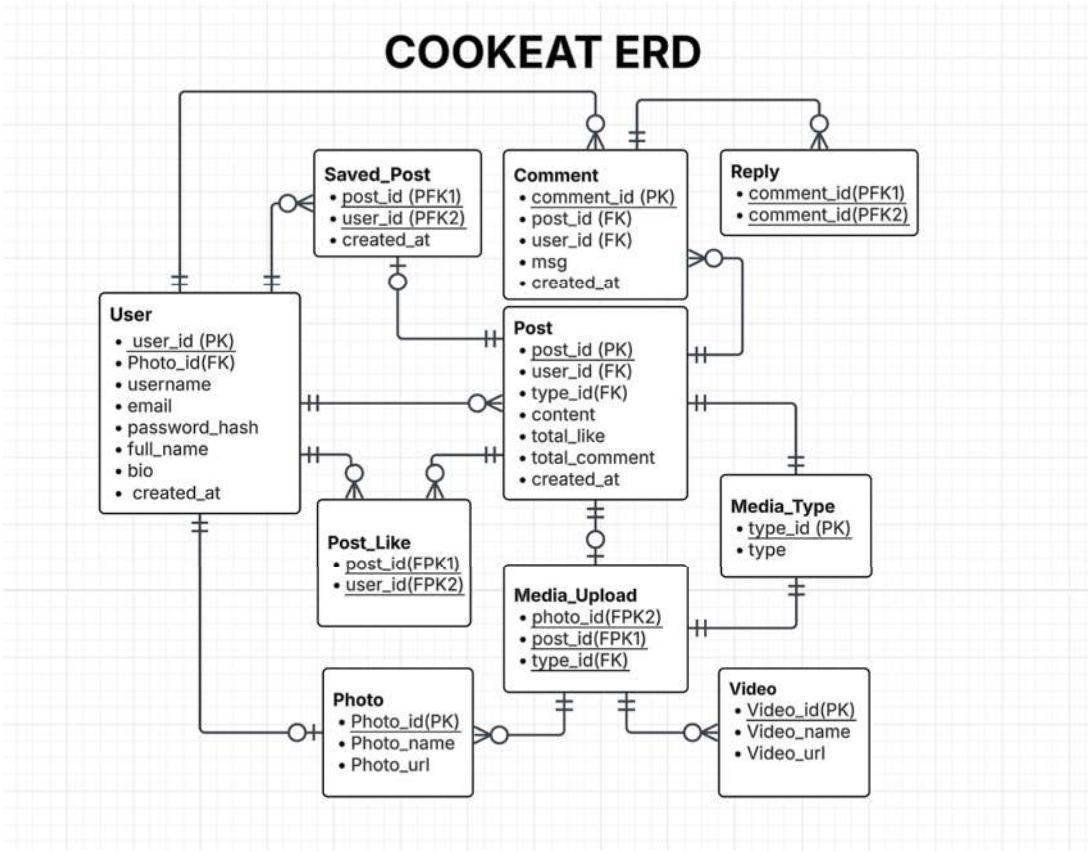


Table information:

User (user_id, username, email, full_name, password, profile_picture, bio)
- Stores information about each user.
- user_id is the primary key.
- username and email must be unique.
- profile_picture and bio are optional user profile details.

Post (post_id, user_id, description, media_url, media_type_id, created_at)
- Stores posts made by users.
- post_id is the primary key.
- user_id is a foreign key referencing User(user_id).
- media_url points to the uploaded image or video.
- media_type_id references Media_Type(type_id) to validate media type.
- created_at stores the timestamp of the post.

Media_Type (type_id, type)
- Stores the allowed media types for posts.
- type_id is the primary key (e.g., 1 for image, 2 for video).
- type is a descriptive string (e.g., 'image', 'video').

Comment (comment_id, post_id, user_id, msg, created_at)
- Stores comments made by users on posts.
- comment_id is the primary key.
- post_id references the post being commented on.
- user_id references the user who made the comment.
- created_at stores when the comment was made.

Reply (reply_id, comment_id, user_id, msg, created_at)
- Stores replies to comments (not to other replies).
- reply_id is the primary key.
- comment_id is a foreign key referencing the parent comment.
- user_id is a foreign key referencing the replier.
- msg is the reply content.
- created_at tracks when the reply was made.

Like (post_id, user_id)
- Represents users liking posts.
- Composite primary key of post_id and user_id.
- post_id references the liked post.
- user_id references the liking user.
- Each user can like a post only once.

Comment_Like (comment_id, user_id)
- Represents users liking comments.
- Composite primary key of comment_id and user_id.
- comment_id references the liked comment.
- user_id references the user who liked the comment.

Saved (post_id, user_id)
- Represents users saving posts.
- Composite primary key of post_id and user_id.
- post_id references the saved post.
- user_id references the saving user.
- Each user can save a post only once.

2. CRUD Implementation

A. Create

Here we have several Create features, such creating a new account, creating posts, and commenting.

- **Create new account (register.php)**


```php
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $username = $_POST['username'];
    $email = $_POST['email'];
    $password = password_hash(password: $_POST['password'], algo: PASSWORD_DEFAULT);
    $full_name = $_POST['full_name'];

    $checkEmailStmt = $pdo->prepare("SELECT * FROM user WHERE email = ? OR username = ?");
    $checkEmailStmt->execute([$email, $username]);
    $result = $checkEmailStmt->fetchAll();

    if ($result) {
        $message = "Email or Username is already exists";
    }else{
        $stmt = $pdo->prepare("INSERT INTO User (username, email, password_hash, full_name, bio, photo_id) VALUES (?, ?, ?, ?, ?, 1)");
        $stmt->execute([$username, $email, $password, $full_name, $bio]);
        $_SESSION['user_id'] = $user['user_id'];
        header(header: "Location: index.php");
        exit;
    }
}
?>
```

process of creating a new account is done by entering data such as username, full_name, email, password, and photo_id into the User table, and using hashed for the password.

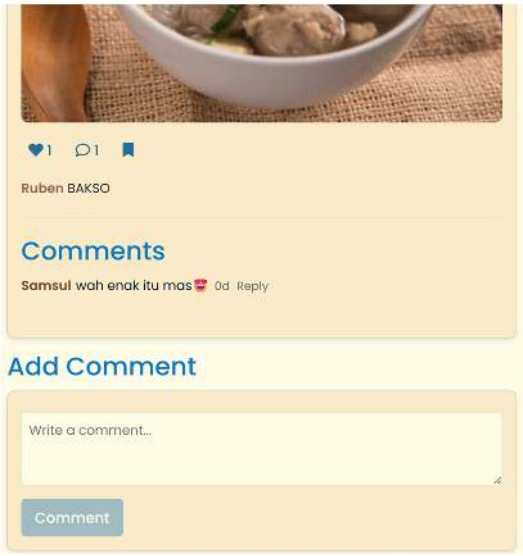
- **Create post (create_post.php)**

```php
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $user_id = $_SESSION['user_id'];

    // Validasi bahwa field content dan type_id ada
    if (!isset($_POST['content']) || !isset($_POST['type_id'])) {
        $error_message = "Please fill in all required fields.";
    } else {
        $content = $_POST['content'];
        $type_id = $_POST['type_id'];

        if (!empty($_FILES['media']['name'])) {
            $allowed_image_types = ['image/jpeg', 'image/png', 'image/gif'];
            $allowed_video_types = ['video/mp4', 'video/webm', 'video/ogg'];
            $file_type = $_FILES['media']['type'];

            // Validasi ukuran file (maksimal 5MB)
            $max_file_size = 50 * 1024 * 1024; // 5MB dalam bytes
            if ($_FILES['media']['size'] > $max_file_size) {
                $error_message = "File size exceeds 50MB limit.";
            } elseif ($_FILES['media']['error'] !== UPLOAD_ERR_OK) {
                $error_message = "Error uploading file. Please try again.";
            } elseif ($type_id == 1 && !in_array(needle: $file_type, haystack: $allowed_image_types)) {
                $error_message = "Only JPEG, PNG, or GIF images are allowed.";
            } elseif ($type_id == 2 && !in_array(needle: $file_type, haystack: $allowed_video_types)) {
                $error_message = "Only MP4, WebM, or OGG videos are allowed.";
            } else {
                $stmt = $pdo->prepare("INSERT INTO Post (user_id, type_id, content) VALUES (?, ?, ?)");
                $stmt->execute([$user_id, $type_id, $content]);
                $post_id = $pdo->lastInsertId();
```

```php
                if ($type_id == 1) { // Photo
                    $photo_name = $_FILES['media']['name'];
                    $photo_url = "assets/uploads/" . $photo_name;
                    move_uploaded_file(from: $_FILES['media']['tmp_name'], to: $photo_url);

                    $stmt = $pdo->prepare("INSERT INTO Photo (photo_name, photo_url) VALUES (?, ?)");
                    $stmt->execute([$photo_name, $photo_url]);
                    $photo_id = $pdo->lastInsertId();

                    $stmt = $pdo->prepare("INSERT INTO Media_Upload (photo_id, post_id, type_id) VALUES (?, ?, ?)");
                    $stmt->execute([$photo_id, $post_id, $type_id]);
                } elseif ($type_id == 2) { // Video
                    $video_name = $_FILES['media']['name'];
                    $video_url = "assets/uploads/" . $video_name;
                    move_uploaded_file(from: $_FILES['media']['tmp_name'], to: $video_url);

                    $stmt = $pdo->prepare("INSERT INTO Video (video_name, video_url) VALUES (?, ?)");
                    $stmt->execute([$video_name, $video_url]);
                    $video_id = $pdo->lastInsertId();

                    $stmt = $pdo->prepare("INSERT INTO Media_Upload (photo_id, post_id, type_id) VALUES (?, ?, ?)");
                    $stmt->execute([$video_id, $post_id, $type_id]);
                }

                header(header: "Location: index.php");
                exit();
            }
        } else {
            $error_message = "Please upload a file.";
        }
    }
```

Create a new post is involve saving data such as content, user_id, type_id for photo/video, and media to the Post and media_upload tables, with validation to ensure content is not empty and media files only .jpg/.mp4

- **Comment (post.php)**

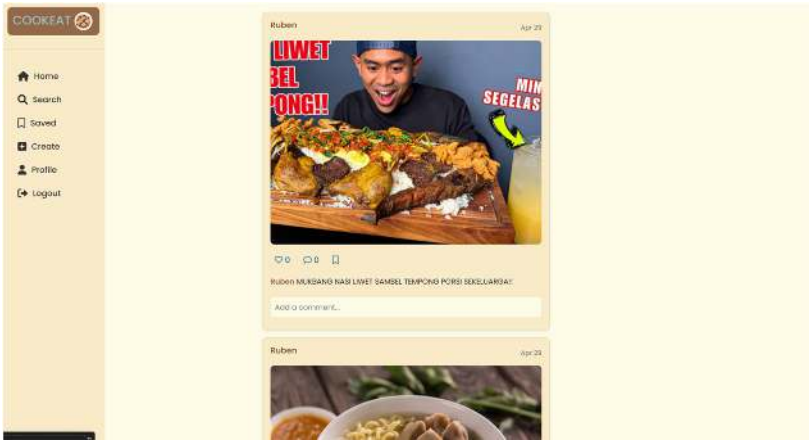



```php
// Add comment
if (isset($_POST['comment'])) {
    $msg = $_POST['msg'];
    $stmt = $pdo->prepare("INSERT INTO Comment (post_id, user_id, msg) VALUES (?, ?, ?)");
    $stmt->execute([$post_id, $user_id, $msg]);
    $stmt = $pdo->prepare("UPDATE Post SET total_comment = total_comment + 1 WHERE post_id = ?");
    $stmt->execute([$post_id]);
    header(header: "Location: post.php?post_id=$post_id");
}
```

The comment creation process saves the content, user_id, and post_id to the comments table, with validation to ensure the comment is not empty and the post exists.

B. Read

- **Show post feed (index.php)**



Displays a feed of recent posts from all users, get from the ‘Post’ table with information (`User`) and media (`Photo` or `Video`) information via `Media_Upload`. Posts are sorted by creation time `created_at` in descending order.

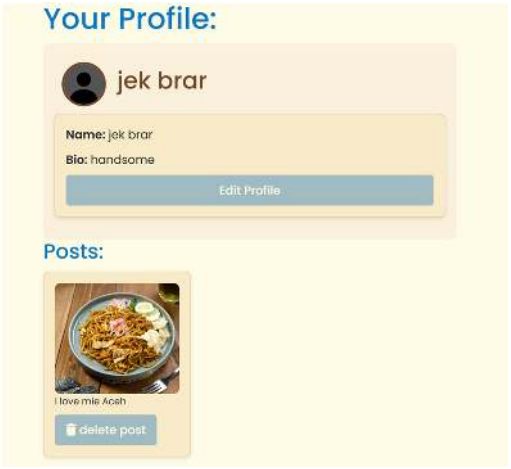
```php
$stmt = $pdo->query("SELECT p.*, u.username, ph.photo_url, v.video_url, mu.type_id
                     FROM Post p
                     JOIN User u ON p.user_id = u.user_id
                     LEFT JOIN Media_Upload mu ON p.post_id = mu.post_id
                     LEFT JOIN Photo ph ON mu.photo_id = ph.photo_id
                     LEFT JOIN Video v ON mu.photo_id = v.video_id
                     ORDER BY p.created_at DESC");
```

```php
<div class="feed bawah">
    <?php foreach ($posts as $post): ?>
        <?php
        // Check if user has liked the post
        $stmt = $pdo->prepare("SELECT * FROM Post_Like WHERE post_id = ? AND user_id = ?");
        $stmt->execute([$post['post_id'], $user_id]);
        $has_liked = $stmt->fetch();

        // Check if user has saved the post
        $stmt = $pdo->prepare("SELECT * FROM Saved_Post WHERE post_id = ? AND user_id = ?");
        $stmt->execute([$post['post_id'], $user_id]);
        $has_saved = $stmt->fetch();
        ?>

        <div class="post card">
            <div onclick="post(<?php echo $post['post_id']; ?>)">
                <div class="post-header">
                    <a href="profile.php?username=<?php echo $post['username']; ?>"><h3 style="color:#744928;"><?php echo $post['username']; ?></h3></a>
                    <span class="post-time"><?php echo date(format: 'M j', timestamp: strtotime(datetime: $post['created_at'])); ?></span>
                </div>
                <?php if ($post['type_id'] == 1): ?>
                    <img src="<?php echo $post['photo_url']; ?>" alt="Post Media">
                <?php elseif ($post['type_id'] == 2): ?>
                    <video controls>
                        <source src="<?php echo $post['video_url']; ?>" type="video/mp4">
                        Your browser does not support the video tag.
                    </video>
                <?php else: ?>
                    <p>No media available</p>
                <?php endif; ?>
```

- **Show comment**

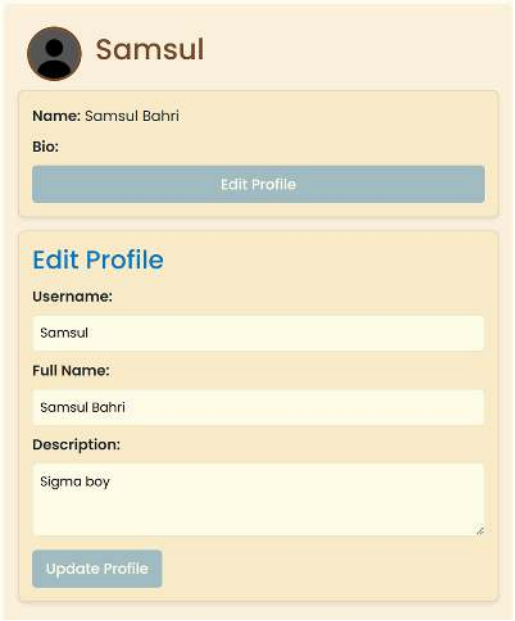
This allows all users to view existing comments. this will display a list of comments along with their usernames, sorted by the time they were created.

- **User profile (profile.php) etc.**



C. Update

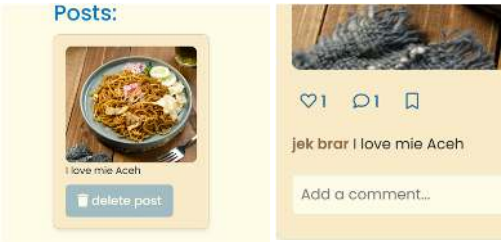
- **Update user profile (profile.php)**



Users are given the ability to edit their profile, including name, description and so on, with validation to ensure the username is not empty and unique, using prepared statement

D. Delete

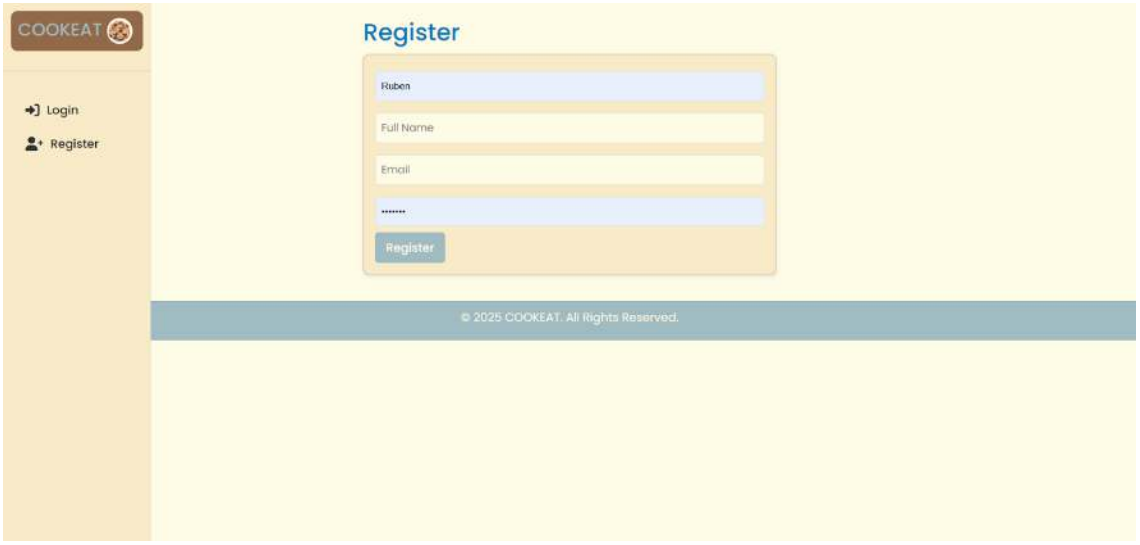
- **Delete post, unlike, unsave**



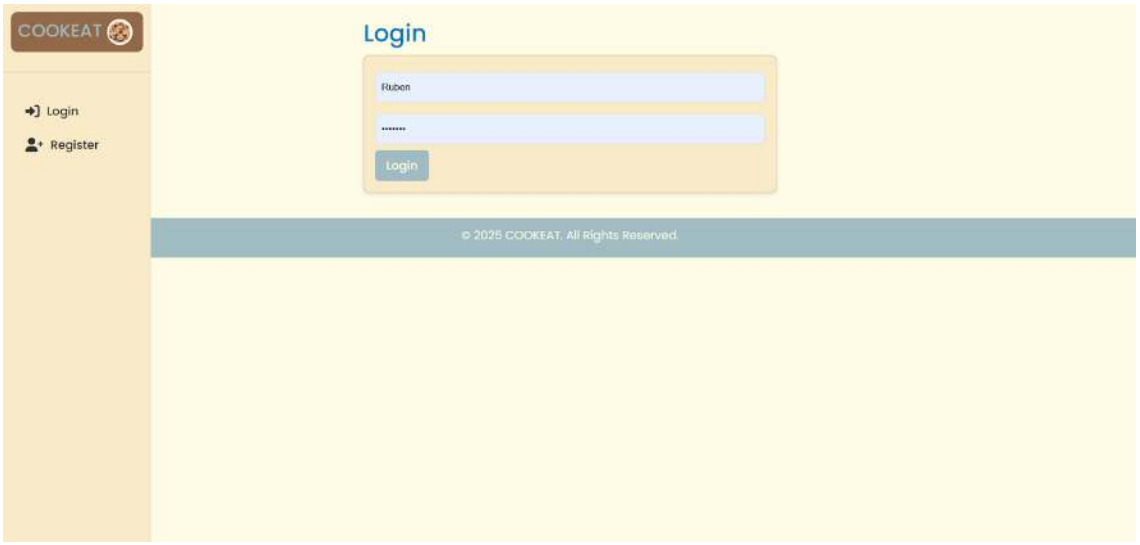
Delete post process involves verifying ownership based on user_id, then deleting entries from the comments, media_upload, saved_post, likes, and Post tables sequentially to maintain foreign key integrity. and also javascript confirmation.
unlike process is done by deleting entries from the likes table based on user_id and post_id, then reducing the like_count in the Post table using GREATEST to prevent negative values. as well as **unsave** by deleting entries from the saved_post table based on user_id and post_id.

3. Flow of Application

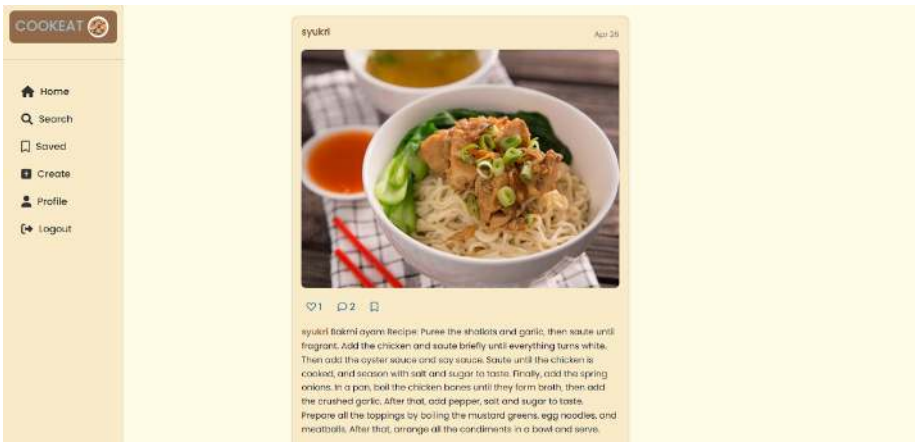
The website will first ask for us to login, but if we don't have an account, there is a "Register" button.



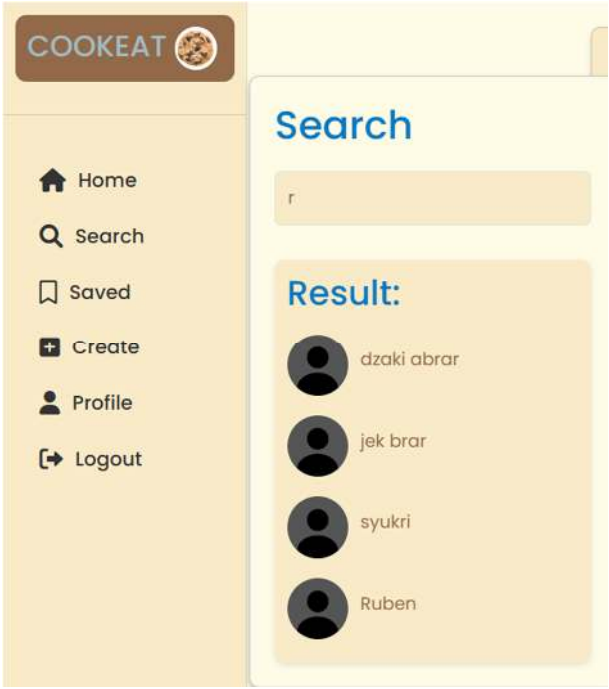
When the Register button is clicked, it goes to another page where you can fill in all of your credentials such as user name, email, full name, and lastly the password.



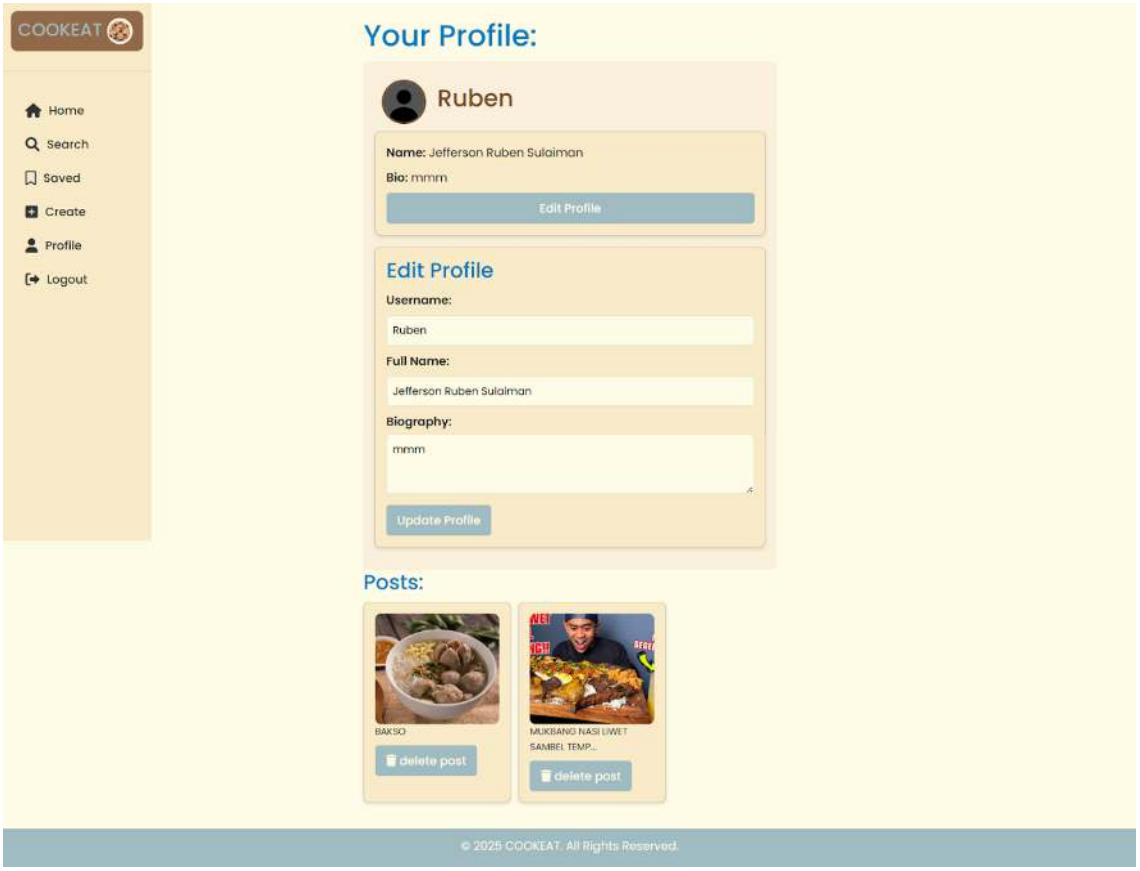
After signing up, the page then goes back to the login page and asks you to login using the email and password that you entered. Once entered, click the login button and it will direct you to the dashboard.



In the dashboard, users can navigate the website using the navigation bar on the left side of the screen. This includes buttons for Home, Search, Saved, Create, Profile, and Logout. The first page users will see is the Home page. The Home page displays the posts, which includes the post content itself, a comment section, a like button, and a save button. Posts can contain either a photo or a video.

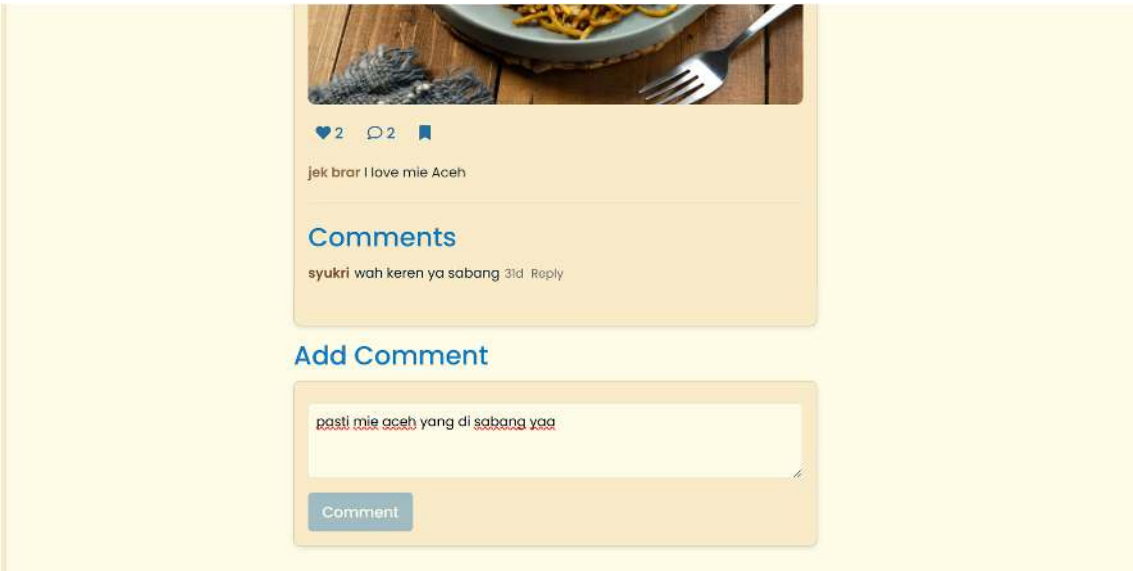


The search button which is located on the nav bar can be pressed to search for a specific user that is in the database. When pressed, the user is redirected to the clicked user's profile page. the profile page displays the user's profile picture, username, full name, biography, and their posts.



In the profile tab. It will show your name and bio, and also your posts which have a button that can delete them. There is also an edit profile button which is used to edit the username, full name, and the biography.

The saved tab is where all your bookmarked posts are located. If you ever need to check them for later. To unsave the posts, a dedicated button is available to remove the posts from the saved tab.



And we can comment on and even reply to other comments in all posts on the web.

The Posting tab lets users upload a photo or video and add a description. Once done, the post can be shared with others on the platform.

At the very bottom is the log out button which destroys the session and logs you out.