➢ **Logout.php**

```php
<?php
session_start();
session_unset();
header("location:login.php");
```

Starts a new session or resumes an existing session using session_start().

Unsets all session variables, effectively logging out the user, using session_unset().

Redirects the user to the login page using header("location:login.php").

➢ **db_connect.php**

```php
<?php
// Database connection parameters
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "twitter_clone";

// Create connection
$conn = mysqli_connect($servername, $username,
$password, $dbname);

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
?>
```

Defines the database connection parameters: server name ($servername), username ($username), password ($password), and database name ($dbname).

Creates a connection to the MySQL database using these parameters with mysqli_connect().

Checks if the connection was successful. If not, it stops the execution and displays an error message using die() and mysqli_connect_error().

- **like_tweet.php.**

```php
<?php
session_start();
require_once('classes.php');

if (isset($_SESSION["user"])) {
    $user_data = $_SESSION["user"];
    $user = unserialize($user_data);
    if (is_object($user) &&
isset($_POST['tweet_id'])) {
        $tweet_id = $_POST['tweet_id'];
        $user->addLikeToTweet($tweet_id);
    }
}

header("Location: home.php");
exit();
?>
```

Starts a new session or resumes an existing session using session_start().

Includes the file classes.php using require_once(), which likely contains class definitions needed for the code.

Checks if there is a "user" stored in the session. If so, it retrieves the user data from the session and unserializes it to recreate the user object.

If the unserialized data is a valid object and a tweet_id is set in the POST request, it retrieves the tweet_id and calls the addLikeToTweet method on the user object to like the specified tweet.

Redirects the user to the home page using header("Location: home.php") and terminates the script with exit().

- **remove_like.php.**

```php
<?php
session_start();
require_once('classes.php');

if (isset($_POST['tweet_id']) &&
isset($_SESSION['user'])) {
    $tweet_id = $_POST['tweet_id'];
    $user = unserialize($_SESSION['user']);
    if ($user) {
        $user->removeLikeFromTweet($tweet_id);
        header("Location: home.php");
        exit;
    } else {
        echo "Error: User not logged in.";
        exit;
    }
}
?>
```

Starts a new session or resumes an existing session using session_start().

Includes the file classes.php using require_once(), which likely contains class definitions needed for the code.

Checks if a tweet_id is set in the POST request and if a "user" is stored in the session.

Retrieves the tweet_id from the POST request.

Unserializes the user object stored in the session.

If the user object is valid, calls the removeLikeFromTweet method on the user object with the tweet_id to unlike the specified tweet.

Redirects the user to the home page using header("Location: home.php") and terminates the script with exit().

If the user object is not valid, it outputs an error message "Error: User not logged in." and terminates the script.

- **delete_user.php**

```php
<?php
require_once('classes.php');
session_start();

if (isset($_SESSION["user"])) {
    $user_data = $_SESSION["user"];
    $user = unserialize($user_data);
    if (is_object($user)) {
        $username = $user->username;
    } else {
        echo "Error: Unable to unserialize user data.";
        exit;
    }
} else {
    echo "You are not logged in.";
    exit;
}

$user_id = $_REQUEST['user_id'];
$user->delete_user($user_id);
header('Location:admin.php?msg=done');
exit;
```

Includes the file classes.php using require_once(), which likely contains class definitions needed for the code.

Starts a new session or resumes an existing session using session_start().

Checks if a "user" is stored in the session.

If a user is found, it retrieves the user data from the session and unserializes it to recreate the user object.

If unserialization is successful and the data is a valid object, it retrieves the username from the user object.

If unserialization fails, it outputs an error message "Error: Unable to unserialize user data." and terminates the script.

If no user is found in the session, it outputs "You are not logged in." and terminates the script.

Retrieves a user ID from the request using $_REQUEST['user_id'].

Calls the delete_user method on the user object with the provided user ID to delete the specified user.

Redirects the user to admin.php with a query parameter msg=done using header('Location:admin.php?msg=done') and terminates the script with exit().

- **post_tweets.php**

```
<?php
$connection = mysqli_connect('localhost', 'root', '',
'twitter_clone');

if (!$connection) {
    die("Connection failed: " . mysqli_connect_error());
}

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (!empty($_POST["tweet"])) {
        $tweet_text = $_POST["tweet"];

        $query = "INSERT INTO tweets (user_id,
tweet_text, created_at) VALUES ('$user_id',
'$tweet_text', NOW())";

        if (mysqli_query($connection, $query)) {

            header("Location: home.php");
            exit();
        } else {
            echo "Error: " . $query . "<br>" .
mysqli_error($connection);
        }
    } else {
        echo "Tweet cannot be empty!";
    }
}
```

- $_POST["tweet"] retrieves the tweet content from the form submission.

- The $query variable stores the SQL query to insert the new tweet into the tweets table. The NOW() function is used to automatically set the current timestamp for the created_at field.
- mysqli_query($connection, $query) executes the SQL query.
- If the query is successful, the user is redirected to home.php using the header() function, and exit() stops further script execution.
- If the query fails, an error message is displayed showing the query and the error returned by MySQL using mysqli_error($connection).

- If the tweet content is empty, an error message "Tweet cannot be empty!" is displayed.

```
$query = "INSERT INTO tweets (user_id, tweet_text,
created_at) VALUES ('$user_id', '$tweet_text', NOW())";

if (mysqli_query($connection, $query)) {
    header("Location:home.php");
    exit();
} else {
    echo "Error: " . $query . "<br>" .
mysqli_error($connection);
}


mysqli_close($connection);
?>
```

Connects to a MySQL database named twitter_clone.

Checks the connection and terminates with an error message if it fails.

If the request method is POST and the tweet is not empty:

Retrieves the tweet text from the POST request.

Executes an SQL query to insert the tweet into the tweets table with the current timestamp.

If the insertion is successful, redirects to home.php.

If there is an error, it displays the error message.

If the tweet is empty, it displays an error message.

Closes the database connection.

- **store_posts.php.**

```php
<?php
session_start();
require_once('classes.php');

if (isset($_SESSION["user"])) {
    $user_data = $_SESSION["user"];
    $user = unserialize($user_data);
    if (is_object($user)) {
        $username = $user->username;
    } else {
        echo "Error: Unable to unserialize user data.";
        exit;
    }
} else {
    echo "You are not logged in.";
    exit;
}

if (!empty($_REQUEST["title"]) &&
!empty($_REQUEST["content"])) {
    if (!empty($_FILES["image"]["name"])) {
        $uploadDir = "images/";
        $imageName = $uploadDir .
basename($_FILES["image"]["name"]);
```

```
        if
(move_uploaded_file($_FILES["image"]["tmp_name"],
$imageName)) {
        } else {
            $imageName = null;
        }
    } else {
        $imageName = null;
    }

    $createdAt = date('Y-m-d H:i:s');
    $user->store_posts($_REQUEST["title"],
$_REQUEST["content"], $imageName, $createdAt);
    header("Location: home.php?error_msg=done");
} else {
    header("Location:
home.php?error_msg=require_fields");
}
?>
```

Starts a session and includes the classes.php file.

Checks if a user is logged in by verifying the session data and unserializing the user object.

If the user is logged in and unserialized successfully, retrieves the username.

If the user is not logged in or the unserialization fails, it outputs an error message and terminates the script.

If the request contains both a title and content:

Checks if an image file is uploaded.

If an image is uploaded, moves it to the images/ directory and stores the file path; otherwise, sets the image path to null.

Gets the current date and time.

Calls the store_posts method on the user object to save the post with the provided title, content, image path, and timestamp.

Redirects to home.php with a success message.

If required fields are missing, redirects to home.php with an error message.

- **header.php.**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>home/ twitter</title>
    <link rel="stylesheet"href="css/style.css">
    <link
rel="stylesheet"href="https://cdnjs.cloudflare.com/ajax/
libs/font-awesome/5.15.1/css/all.min.css">
    <style>
        .navbar {
    margin-bottom: 30px;
    background-color: #ffc0cb;
    height: 70px;




}

.navbar-brand, .nav-link {
    color: #040404 !important;
}

.navbar-brand:hover, .nav-link:hover {
    color: #ffffff !important;
}

#logoutBtn {
    background-color: #dc3545;
    border: none;
}

#logoutBtn:hover {
    background-color: #c82333;
}
    </style>
</head>
<body>
     <!-- Navigation Bar -->
     <nav class="navbar navbar-expand-lg fixed-top">
        <a class="navbar-brand" href="#">
            <img src="images/twitter.png" width="30"
height="30" class="d-inline-block align-top " alt="">
            pink twitter
        </a>
        <button class="navbar-toggler" type="button"
data-toggle="collapse" data-target="#navbarNav" aria-
controls="navbarNav" aria-expanded="false" aria-
label="Toggle navigation">
            <span class="navbar-toggler-
icon"style="background-color: white ;"></span>
        </button>
```

```html
        <div class="collapse navbar-collapse"
id="navbarNav">
            <ul class="navbar-nav ml-auto">
                <li class="nav-item active">
                    <a class="nav-link"
href="home.php">Home <span class="sr-
only">(current)</span></a>
                </li>
                <li class="nav-item">
                    <a class="nav-link"
href="#">About</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link"
href="#">Contact</a>
                </li>
                <li class="nav-item">
                    <a class="btn btn-danger"
id="logoutBtn" href="logout.php">Logout</a>
                </li>
            </ul>
        </div>
    </nav>


</body>
</html>
```

Defines an HTML document with the required metadata and styles.

Links external stylesheets for custom styles and Font Awesome icons.

Includes inline CSS to style the navigation bar with specific colors and hover effects.

Creates a navigation bar with:

A brand logo and title "pink twitter".

A collapsible menu with links to "Home", "About", "Contact", and a "Logout" button.

Styles the logout button to have a red background that changes on hover.

- **classes.php.**

```
<?php

class User {
    public $id;
    public $username;
    public $email;
    protected $hashed_password;
    public $created_at;
    public $updated_at;
    public $role;

    public function __construct($id, $username, $email,
$hashed_password, $created_at, $updated_at, $role) {
        $this->id = $id;
        $this->username = $username;
        $this->email = $email;
        $this->hashed_password = $hashed_password;
        $this->created_at = $created_at;
        $this->updated_at = $updated_at;
        $this->role = $role;
    }
    public static function register($username, $email,
$hashed_password) {
        $user = new Subscriber(null, $username, $email,
$hashed_password, date('Y-m-d H:i:s'), date('Y-m-d
H:i:s'), "subscriber");
        return $user;
    }

    public static function login($email,
$hashed_password) {
        $conn = mysqli_connect("localhost", "root", "",
"twitter_clone");
            if (!$conn) {
            die("Connection failed: " .
mysqli_connect_error());
        }

        $sql = "SELECT * FROM users WHERE email =
'$email' AND password = '$hashed_password'";
        $result = mysqli_query($conn, $sql);
        $user_data = mysqli_fetch_assoc($result);

        mysqli_close($conn);

        if ($user_data) {
            if ($user_data['role'] == "subscriber") {
                $user = new Subscriber($user_data['id'],
$user_data['username'], $user_data['email'],
$user_data['password'], $user_data['created_at'],
$user_data['updated_at'], $user_data['role']);
            } elseif ($user_data['role'] == "admin") {
                $user = new Admin($user_data['id'],
$user_data['username'], $user_data['email'],
$user_data['password'], $user_data['created_at'],
$user_data['updated_at'], $user_data['role']);
            }
            return $user;
        } else {
            return null;
        }
    }
}

class Subscriber extends User {
    public $role = "subscriber";

    public function __construct($id, $username, $email,
$hashed_password, $created_at, $updated_at, $role) {
        parent::__construct($id, $username, $email,
$hashed_password, $created_at, $updated_at, $role);
    }

    public function store_posts($title, $content,
$imageName, $created_at) {
        $sql = "INSERT INTO tweets (user_id, title,
content, image, created_at) VALUES ('$this-
>id','$title','$content','$imageName','$created_at')";
        $conn = mysqli_connect("localhost", "root", "",
"twitter_clone");
        $result = mysqli_query($conn, $sql);
        mysqli_close($conn);
        return $result;
    }
    public function getMyTweets() {
        $sql = "SELECT * FROM tweets WHERE user_id =
'$this->id' ORDER BY created_at DESC";
        $conn = mysqli_connect("localhost", "root", "",
"twitter_clone");
        $result = mysqli_query($conn, $sql);
        $tweets = array();
        while ($row = mysqli_fetch_assoc($result)) {
            $tweets[] = $row;
        }
        mysqli_close($conn);
        return $tweets;
    }


    public function store_comment($tweet_id, $user_id,
$comment_text) {
        $sql = "SELECT id FROM tweets WHERE id =
'$tweet_id'";
        $conn = mysqli_connect("localhost", "root", "",
"twitter_clone");
        $result = mysqli_query($conn, $sql);
            $sql = "INSERT INTO comments(tweet_id,
user_id, comment_text) VALUES
('$tweet_id','$user_id','$comment_text')";
            $result = mysqli_query($conn, $sql);
            mysqli_close($conn);
            return $result;
```

User Class

1. Properties
   - $id: The user's unique identifier.
   - $username: The user's name.
   - $email: The user's email address.
   - $hashed_password: The user's hashed password (protected for security).
   - $created_at: The timestamp when the user was created.
   - $updated_at: The timestamp when the user was last updated.
   - $role: The user's role (e.g., subscriber, admin).
2. Constructor
   - Initializes the user object with the given values.
3. Register Method
   - Creates a new Subscriber user with the provided username, email, and password.
   - Sets the current time for created_at and updated_at.
   - Returns the new Subscriber object.
4. Login Method
   - Connects to the database and looks for a user with the provided email and password.
   - If found, creates either a Subscriber or Admin object based on the user's role.
   - Returns the user object or null if not found.

Subscriber Class

1. Role Property
   - Sets the role of the Subscriber to "subscriber".
2. Constructor
   - Calls the parent (User) constructor to initialize the properties.
3. Store Posts Method
   - Inserts a new post (tweet) into the database with the provided title, content, image name, and creation time.
   - Connects to the database, executes the query, and returns the result.

```
    public function getCommentsForTweet($tweet_id) {
        $sql = "SELECT * FROM comments WHERE tweet_id =
'$tweet_id'";
        $conn = mysqli_connect("localhost", "root", "",
"twitter_clone");
        $result = mysqli_query($conn, $sql);
        $comments = array();
        while ($row = mysqli_fetch_assoc($result)) {
            $user_id = $row['user_id'];
            $comments[] = array( 'comment_text' =>
$row['comment_text']);
        }
        mysqli_close($conn);
        return $comments;
    }
    public function getLikesForTweet($tweet_id) {
        $sql = "SELECT COUNT(*) as like_count FROM likes
WHERE tweet_id = '$tweet_id'";
        $conn = mysqli_connect("localhost", "root", "",
"twitter_clone");
        $result = mysqli_query($conn, $sql);
        $like_count =
mysqli_fetch_assoc($result)['like_count'];
        mysqli_close($conn);
        return $like_count;
    }

    public function addLikeToTweet($tweet_id) {
        $sql = "INSERT INTO likes (tweet_id, user_id)
VALUES ('$tweet_id', '$this->id')";
        $conn = mysqli_connect("localhost", "root", "",
"twitter_clone");
        $result = mysqli_query($conn, $sql);
        mysqli_close($conn);
        return $result;
    }
    public function removeLikeFromTweet($tweet_id) {
        $sql = "DELETE FROM likes WHERE tweet_id =
'$tweet_id' AND user_id = '$this->id'";
        $conn = mysqli_connect("localhost", "root", "",
"twitter_clone");
```

```
        $result = mysqli_query($conn, $sql);
        mysqli_close($conn);
        return $result;
    }

}


class Admin extends User {
    public $role = "admin";

    public function __construct($id, $username, $email,
$hashed_password, $created_at, $updated_at, $role) {
        parent::__construct($id, $username, $email,
$hashed_password, $created_at, $updated_at, $role);
    }
    public function get_all_users(){
        $sql = "SELECT * FROM users";
        $conn = mysqli_connect("localhost", "root", "",
"twitter_clone");
        if (!$conn) {
            die("Connection failed: ".
mysqli_connect_error());
        }
        $result = mysqli_query($conn, $sql);
        if (!$result) {
            die("Query failed: ". mysqli_error($conn));
        }
        $user_data = array();
        while ($row = mysqli_fetch_assoc($result)) {
            $user_data[] = $row;
        }
        mysqli_close($conn);
        return $user_data;
    }
    public function get_all_tweets(){
        $sql = "SELECT * FROM tweets";
        $conn = mysqli_connect("localhost", "root", "",
"twitter_clone");
        if (!$conn) {
            die("Connection failed: ".
mysqli_connect_error());
        }
        $result = mysqli_query($conn, $sql);
        if (!$result) {
            die("Query failed: ". mysqli_error($conn));
        }
        $user_data = array();
        while ($row = mysqli_fetch_assoc($result)) {
            $user_data[] = $row;
        }
        mysqli_close($conn);
        return $user_data;
    }
    public function delete_user($user_id) {
        $sql = "DELETE FROM likes WHERE user_id =
'$user_id'";
        $conn = mysqli_connect("localhost", "root", "",
"twitter_clone");
        if (!$conn) {
            die("Connection failed: ".
mysqli_connect_error());
        }
        $result = mysqli_query($conn, $sql);
        if (!$result) {
            die("Query failed: ". mysqli_error($conn));
        }

        $sql = "DELETE FROM comments WHERE user_id =
'$user_id'";
        $result = mysqli_query($conn, $sql);
        if (!$result) {
            die("Query failed: ". mysqli_error($conn));
        }

        $sql = "DELETE FROM tweets WHERE user_id =
'$user_id'";
        $result = mysqli_query($conn, $sql);
        if (!$result) {
            die("Query failed: ". mysqli_error($conn));
        }

        $sql = "DELETE FROM users WHERE id =
'$user_id'";
        $result = mysqli_query($conn, $sql);
        if (!$result) {
            die("Query failed: ". mysqli_error($conn));
        }
        mysqli_close($conn);
    }

}

?>
```

Constructor: Initializes the class by calling the parent constructor with provided parameters.

get_all_users():

Connects to the "twitter_clone" database.

Executes a SQL query to retrieve all records from the "users" table.

Collects the result set into an array and returns it.

get_all_tweets():

Connects to the "twitter_clone" database.

Executes a SQL query to retrieve all records from the "tweets" table.

Collects the result set into an array and returns it.

delete_user($user_id):

Connects to the "twitter_clone" database.

Executes a series of SQL queries to delete all likes, comments, tweets, and the user record associated with the given user ID.

Closes the database connection after executing the queries.

Each method includes error handling to terminate the script and display an error message if a database connection or query fails.