README.md Template

WAPH-Web Application Programming and Hacking

Instructor: Dr. Phu Phung

Project Mini Facebook

Team members

1. Amit Gaddi, gaddiat@mail.uc.edu
2. Mohan Sai Jasti, jastimi@mail.uc.edu
3. Taraka Ram Annapaneni, annapatm@mail.uc.edu

Project Management Information

Source code repository (private access): https://github.com/waph-team3/waph-teamproject

Project homepage (public): https://github.com/waph-team3/waph-teamproject/blob/main/Welcome.pdf

Projet Video: https://github.com/waph-team3/waph-teamproject/blob/main/Team3_demonstration.mp4

Revision History

Date	Version	Description
25/04/2024	3.0	Final

Overview

The WAPH Team Project is a collaborative effort to create a "miniFacebook" online application employing full-stack web development technologies, secure programming techniques, and agile development processes. The project begins with Sprint 0, which is intended to build basic team communication and setup. During this phase, team members are required to become acquainted with one another and the project's objectives, clearly divide work, and rigorously document contributions. Effective cooperation requires the use of communication channels, notably Microsoft Teams, as well as frequent meetings for planning and progress monitoring.Sprint 1 focuses on laying the groundwork for the online application, which includes database design and implementation, user registration, and login capabilities. Logged-in individuals may change their passwords, alter their profiles (including name, extra email, and phone number), and access database postings.

The project entails creating a secure user management system with a variety of features. Users may register by entering their username, password, name, and email address, which is validated both on the client and server sides to verify data correctness. The login system employs secure authentication and session management. Once logged in, users may safely access, edit, and update their profiles. The program prioritizes security by delivering via HTTPS, hashing passwords before storage, and employing prepared statements to avoid SQL injection. To avoid XSS attacks, comprehensive input validation has been applied. The project entails database design, front-end programming in HTML, CSS (with optional frameworks), and JavaScript for a responsive interface. Secure session management and CSRF protection are also used to prevent session hijacking and CSRF attacks during database changes.

System Analysis

The primary objective of developing a "miniFacebook" online application is to thoroughly assess its system requirements, functions, and the technological foundation required to achieve the desired results. This initial phase is critical because it provides for a thorough study of the project's scope, simplifies the selection of relevant technology, and establishes an organized approach to development that incorporates secure programming techniques and agile approaches. This rigorous planning and evaluation lays the groundwork for a strong and efficient platform that fulfills the changing needs of its users while prioritizing security and adaptability.

High-level Requirements

The high-level objectives for the WAPH Team Project, which focuses on developing a "miniFacebook" web application, include a variety of functions required to create a user-friendly and secure social networking platform. At its heart, the system must allow users to safely register accounts, verify themselves, and efficiently maintain their profiles. It should enable user interactions by allowing them to connect with friends, exchange posts, photographs, and comments, and manage the privacy settings of their material. Furthermore, the system should be scalable to meet a rising user base, allowing it to manage increased traffic and data quantities effectively. Security is critical, needing strong safeguards such as encryption for sensitive data, secure authentication procedures, and frequent audits to detect and remediate possible flaws. Usability is another critical criterion, requiring an intuitive user interface, smooth navigation, and responsive design to improve the entire user experience. These high-level objectives form the foundation for the creation of a full "miniFacebook" application that satisfies user expectations while following to best practices in web development, security, and user interface design.

Functional Requirements

User Registration: Firstly, a database was created before creating any of the forms. For the user registration form, two PHP files were created.One of these is to accept user input, validate it, and save it in a database. After the user fills out the form, the data is forwarded to the next page, where it is validated and saved in the database. Log in: To implement a page for logging in, a login form has been developed that requests the username and password to authenticate the user. When the details match those in the database, the user is allowed to proceed to the next page where their name and email are displayed, along with options to change the profile details, update the password, and logout. When the user visits the `index.php` page, session management starts and the user's details are stored in the session data, which is then accessible across other pages. Profile Management: Here, the user is allowed to visit the "updateprofile" page only when logged in. On this page, the user can change basic details such as name and email. Password Update: Here too, the user is allowed to access the page only if authenticated. On this page, the user enters their username, current password, and new password.

System Design

Sprint 1

Use-Case Realization

- Implement user registration functionality
- Develop user login system
- Enable users to change their passwords
- Allow users to edit their profiles (name, additional email, phone)
- Enable logged-in users to view posts from the database

Database

- Refine database design based on Sprint 1 requirements
- Implement database schema for user management and posting features
- Populate users table with sample data for testing

User Interface

- Design and implement user registration form
- Develop login form for user authentication
- Create UI elements for password change and profile editing
- Design post viewing interface for logged-in users

Security analysis

- How did you apply the security programming principles in your project? To maintain a safe and trustworthy system, we followed a number of security programming standards throughout this project. We guaranteed the security and integrity of data in transit by integrating session management with PHP sessions and secure communication over HTTPS. Input validation is done implicitly, and SQL queries are prepared to prevent SQL injection threats. Furthermore, hashing passwords (albeit it is inferred rather than expressly stated that hashing is performed before saving passwords in the database) and using HTTPS (indicated by the secure cookie feature) are consistent with the ideas of least privilege and defense in depth.

- What database security principles have you used in your project? We used prepared statements for database interactions to successfully avoid SQL Injection attacks, which are an important part of database security. The notion of least privilege is demonstrated by the use of a specific database user ('team3') with presumably restricted rights customized to the application's needs. Password hashing is suggested, which improves data at rest security by guaranteeing that saved credentials are not in plain text.

- Is your code robust and defensive? How? The code exhibits defensive programming principles by checking session integrity (for example, matching the session's user agent to prevent session hijacking) and using prepared statements to avoid SQL injection. While more complete input validation and error handling would improve resilience, current features such as rigorous session management and secure cookie parameters (HTTPOnly and Secure flags) help to support a defense-in-depth approach.

- How did you defend your code against known attacks such as XSS, SQL Injection, CSRF, Session Hijacking XSS (Cross-Site Scripting): The usage of htmlentities() when repeating user input (for example, the username) reduces XSS risks by encoding potentially hazardous characters. SQL Injection: Prepared statements in SQL queries protect against injection by isolating SQL code from data. CSRF (Cross-Site Request Forgery): While not expressly stated, using secure session management procedures indirectly helps to reduce CSRF vulnerabilities. Using CSRF tokens in forms would be a straightforward measure. Session Hijacking: Checking the user agent string consistency throughout a session aids in the detection and defense against session hijacking attempts. The Secure flag on cookies guarantees that cookies are exclusively transferred via HTTPS, which reduces the danger of interception.

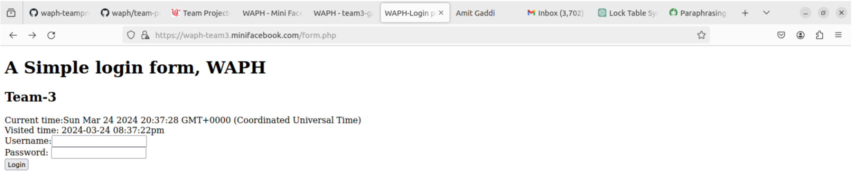
- How do you separate the roles of super users and regular users? The offered code samples do not explicitly explain role separation. However, establishing role-based access control (RBAC) often entails defining roles in the database and assigning users to their appropriate responsibilities.

Access control choices can then be made programmatically depending on the role(s) assigned to the presently authenticated user. This might include conditional checks inside the application logic to determine if a user can access particular features or activities based on their role.

Demo (screenshots)

Sprint 0

- A screenshot demo for the login page working on the HTTPS team's local domain

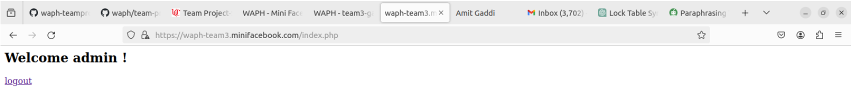


A Simple login form, WAPH

Team-3

Current time:Sun Mar 24 2024 20:37:28 GMT+0000 (Coordinated Universal Time)
Visited time: 2024-03-24 08:37:22pm
Username:
Password:
Login

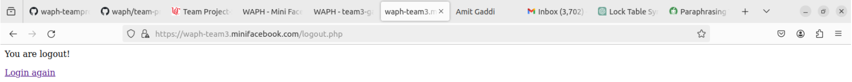
- A screenshot demo for the welcome page working on the HTTPS team's local domain



Welcome admin !

logout

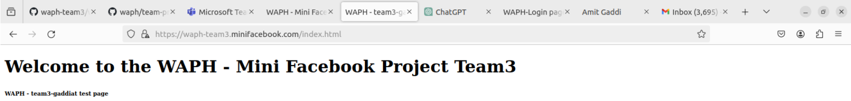
- A screenshot demo for the logout page working on the HTTPS team's local domain



You are logout!

Login again

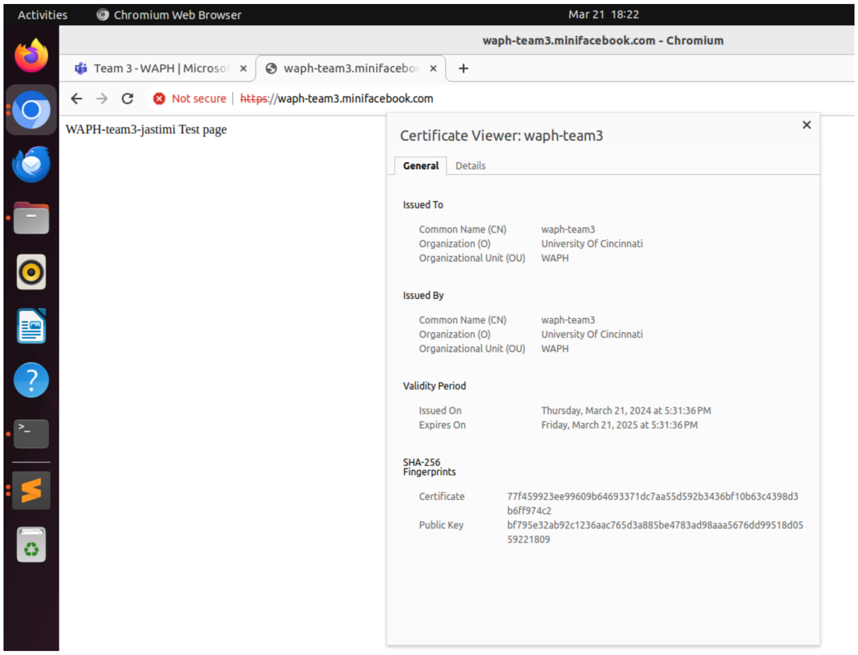
- One screenshot demo for each team member of the index.html page on the HTTPS team's local domain



https://waph-team3.minifacebook.com/index.html

Welcome to the WAPH - Mini Facebook Project Team3

WAPH - team3-gaddiat test page



~~https~~://waph-team3.minifacebook.com

WAPH-team3-jastimi Test page

Certificate Viewer: waph-team3

Issued To

Common Name (CN) waph-team3
Organization (O) University Of Cincinnati
Organizational Unit (OU) WAPH

Issued By

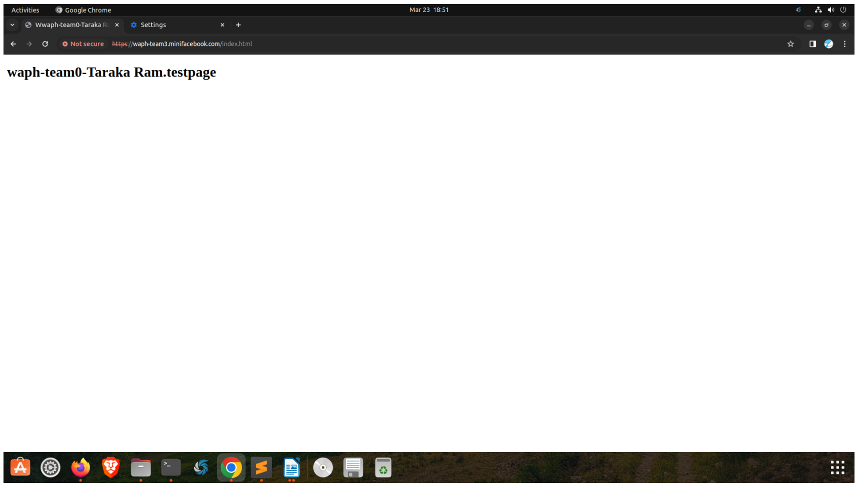
Common Name (CN) waph-team3
Organization (O) University Of Cincinnati
Organizational Unit (OU) WAPH

Validity Period

Issued On Thursday, March 21, 2024 at 5:31:36 PM
Expires On Friday, March 21, 2025 at 5:31:36 PM

SHA-256 Fingerprints

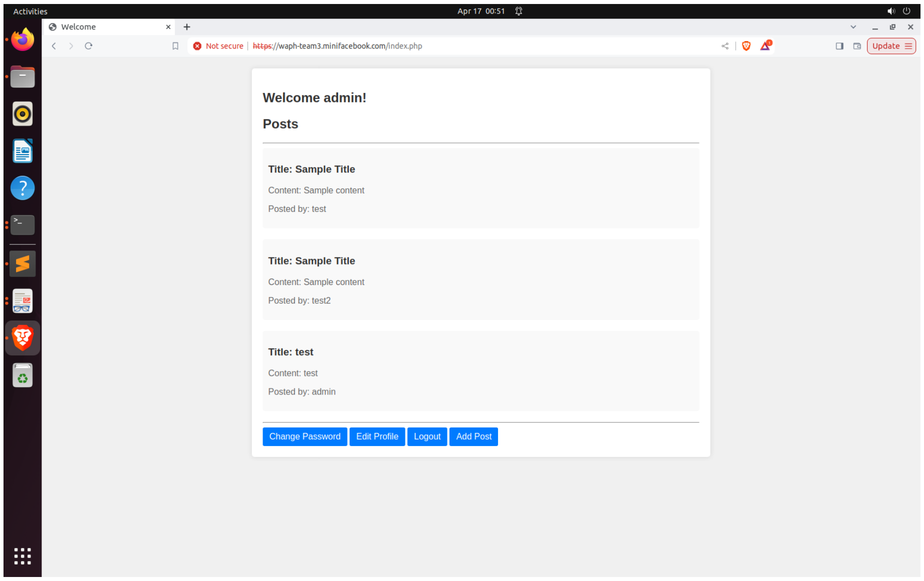
Certificate 77f459923ee99609b64693371dc7aa55d592b3436bf10b63c4398d3b6ff974c2
Public Key bf795e32ab92c1236aac765d3a885be4783ad98aaa5676dd99518d0559221809



waph-team0-Taraka Ram.testpage

Sprint 1

- A index page showing the posts from database.



Welcome admin!

Posts

Title: Sample Title

Content: Sample content

Posted by: test

Title: Sample Title

Content: Sample content

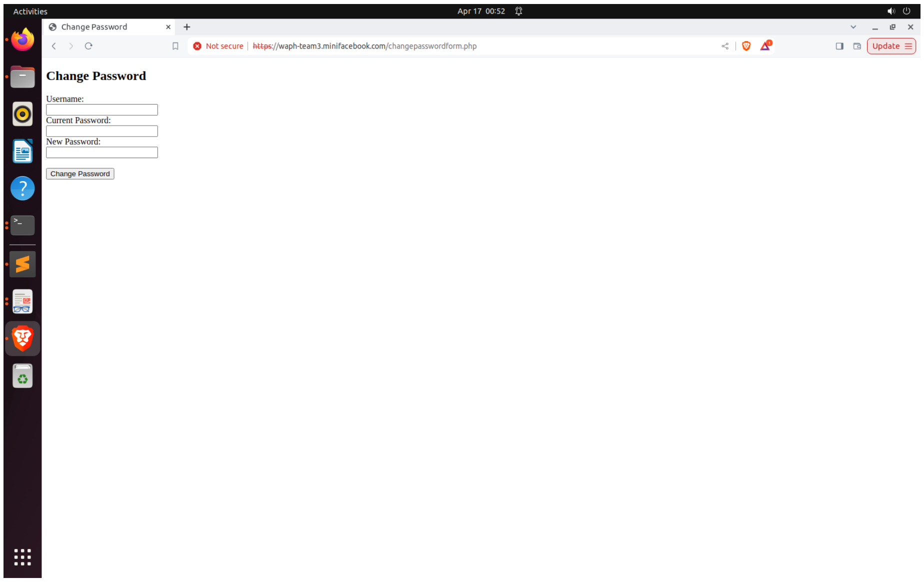
Posted by: test2

Title: test

Content: test

Posted by: admin

- A screenshot that shows the user can change password.



Change Password

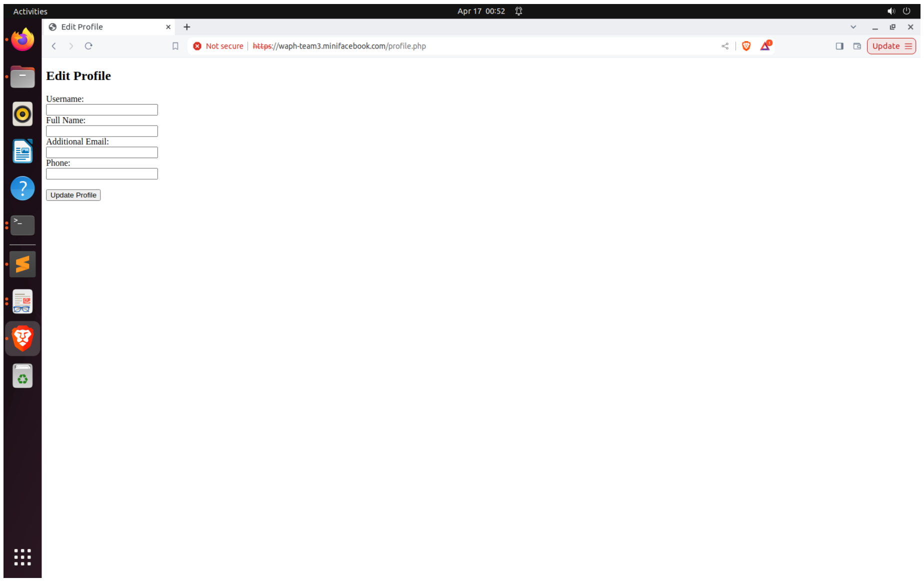
Username:

Current Password:

New Password:

Change Password

- Screenshots showcasing the edit profile functionalities.



Edit Profile

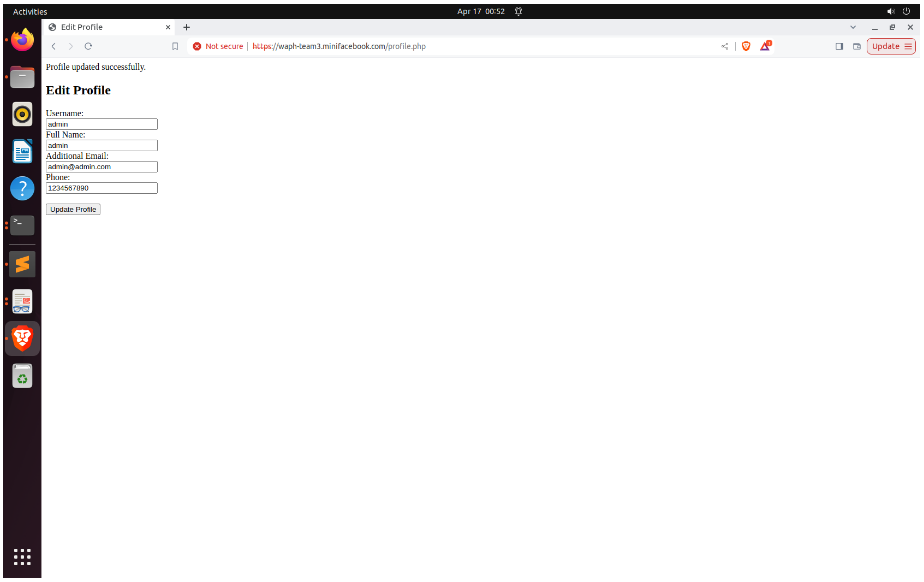
Username:

Full Name:

Additional Email:

Phone:

Update Profile



Profile updated successfully.

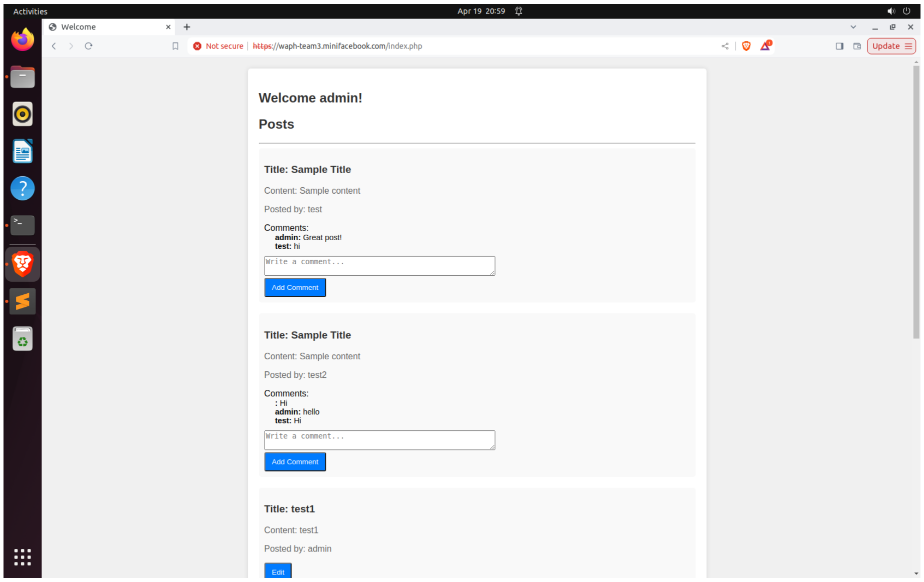
Edit Profile

Username: admin
Full Name: admin
Additional Email: admin@admin.com
Phone: 1234567890

Update Profile

Sprint 2

- A index page showing the posts and comments from database.



Welcome admin!

Posts

Title: Sample Title

Content: Sample content

Posted by: test

Comments:
admin: Great post!
test: hi

Write a comment...

Add Comment

Title: Sample Title

Content: Sample content

Posted by: test2

Comments:
: Hi
admin: hello
test: Hi

Write a comment...

Add Comment

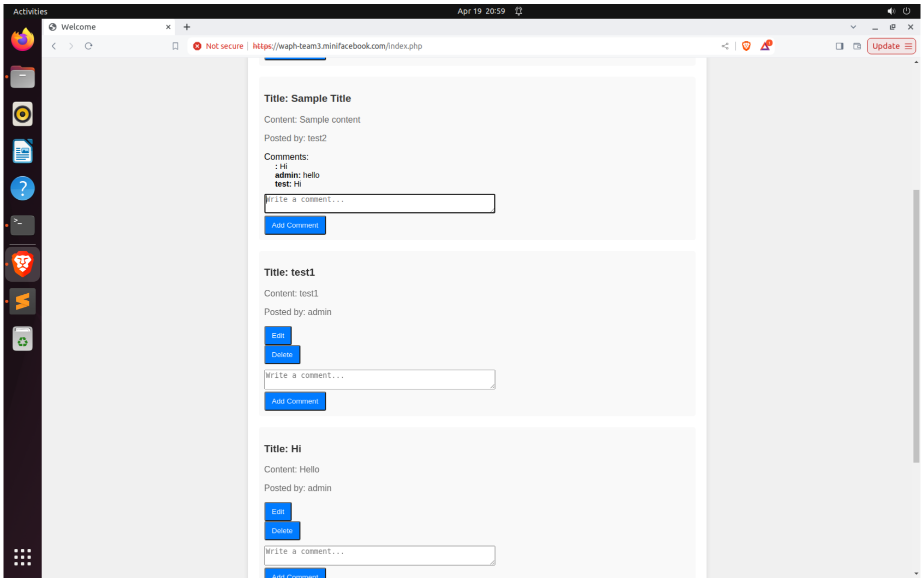
Title: test1

Content: test1

Posted by: admin

Edit

- A index page showing the posts and comments from database and edit and deletion option only to the owner of that post.



Title: Sample Title

Content: Sample content

Posted by: test2

Comments:
: Hi
admin: hello
test: Hi

Title: test1

Content: test1

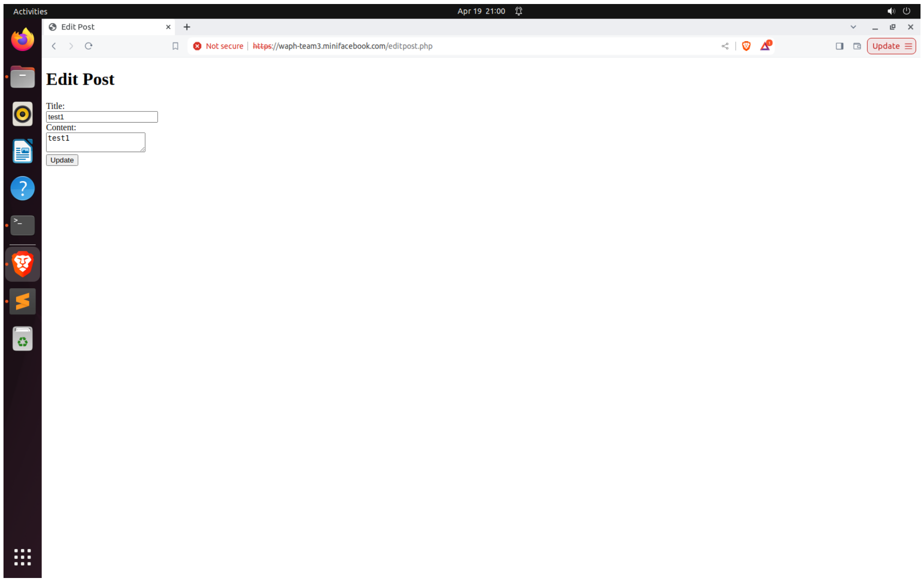
Posted by: admin

Title: Hi

Content: Hello

Posted by: admin

- Screenshots showcasing the edit post functionalities.



Edit Post

Title: test1

Content: test1

Software Process Management

We employed an agile software management approach, especially Scrum, to successfully lead our cooperation and communication for our project, which was centered on creating the “miniFacebook” web application with three individuals. As part of our Scrum strategy, we divided our project into sprints, each lasting one to two weeks. During sprint planning sessions, we describe the tasks that must be accomplished, rank them according to their significance and dependencies, and assign them to specific team members. This ensures that each team member understands their duties and the sprint's goals. Throughout the sprint, we conduct frequent synchronous meetings, to review progress, handle any issues or roadblocks, and alter our plans as necessary. These sessions also serve as chances for cooperation, allowing team members to exchange ideas, offer feedback, and help each other complete their jobs. We utilize collaboration platforms like Microsoft Teams to communicate, share information, and coordinate our work. In addition, we manage our codebase using version control platforms like as Git and GitHub, which allow for seamless collaboration, versioning, and code review procedures. By adopting the Scrum process and cultivating a culture of cooperation and communication, our team ensures that we stay flexible, adaptive, and focused on producing a high-quality product that fulfills project needs and stakeholder expectations.

Scrum process

Sprint 0

Duration: 19/03/2024-24/03/2024

Completed Tasks:

1. Organization Setup on GitHub
2. Member Contributions to Team Organization
3. Team SSL Key/Certificate Setup
4. Design and Setup of Team Database
5. Code Skeleton Copy and Revision

Contributions:

1. Amit, 8 commits, 4 hours, contributed in All the tasks.
2. Mohan, 8 commits, 4 hours, contributed in Tasks - 2 to 5.
3. Taraka , 8 commits, 4 hours, contributed in Tasks - 2 to 5.

Sprint Retrospection:

Good	Could have been better	How to improve?
Helpful team members	Prioritizing tasks	Conduct task estimation

- Prioritizing Tasks: To enhance task prioritizing, we may use a systematic methodology like a prioritization matrix or the MoSCoW method. This will allow us to identify and prioritize tasks that are vital to the project's success.

- Conduct assignment Estimation: We understood the significance of appropriately assessing the time and effort necessary for each assignment. To improve this process, we may use approaches like as story points or time-based estimates to ensure that jobs are properly scoped and planned for completion within the sprint period.

Sprint 1

Duration: 25/03/2024-31/03/2024

Completed Tasks:

1. Database Design and Implementation
2. User Registration Functionality
3. User Login Functionality
4. Password Change Feature
5. Profile Editing Feature (Name, Additional Email, Phone)
6. Viewing Posts from the Database

Contributions:

1. Amit, 6 commits, 5 hours, contributed in All the tasks.
2. Mohan, 6 commits, 5 hours, contributed in All the tasks.
3. Taraka , 6 commits, 5 hours, contributed in All the tasks.

Sprint Retrospection:

Good	Could have been better	How to improve?
Well-defined tasks	More detailed testing	Set clearer deadlines

Sprint 2

Duration: 08/04/2024-16/04/2024

Completed Tasks:

1. Database Design and Implementation
2. Permitting logged-in users to create new posts and comment on any existing posts
3. Allowing logged-in users to modify (edit or delete) their own posts
4. Ensuring that users cannot edit posts authored by other users

Contributions:

1. Amit, 6 commits, 4 hours, contributed in All the tasks.
2. Mohan, 6 commits, 4 hours, contributed in All the tasks.
3. Taraka , 6 commits, 4 hours, contributed in All the tasks.

Sprint Retrospection:

Strengths	Opportunities for Improvement	Suggestions for Enhancement
Effective team collaboration	Task prioritization	Implement task estimation techniques

Sprint Final

Duration: 17/04/2024-25/04/2024

Completed Tasks:

1. Super-user functionality
2. Live chat
3. CSRF protection
4. User disable functionlaity

Contributions:

1. Amit, 8 commits, 8 hours, contributed in All the tasks.
2. Mohan, 8 commits, 8 hours, contributed in All the tasks.
3. Taraka , 8 commits, 8 hours, contributed in All the tasks.

Sprint Retrospection:

Good	Could have been better	How to improve?
Well-defined tasks	More detailed testing	Set clearer deadlines

Security and Non-technical Requirements

- **Security**: The web application is HTTPS secured with a proper certificate provided for local IP and domain addresses. Passwords sent to the database are hashed, as seen in the `database.php` file. This file contains only functions related to the database, and non-root user details have been used to access the database. Every SQL query has a prepared statement to enhance page security as much as possible.

- **Input Validation**: Input validation has been implemented on both the server and client sides, as well as in the database. This includes the registration page, where data is sanitized using built-in functions like `trim()`, `htmlentities()`, and special functions to prevent XSS attacks.

- **Database Design**: Initially, a database account was created where a database and then a user for it were created, as seen in the file in the appendix. Furthermore, a table named users was created to store user details. To secure the database, prepared statements were used for every query execution, and a separate database.php file was maintained that contains only database methods.

- **Role-Based Access Control**: Role-based access control is implemented to differentiate between regular users and super users. Super users have access to administrative functionalities like enabling and disabling other users while regular users have limited access. Also a regular user cannot edit and delete the posts of other users.

- **Front-end Development**: HTML and CSS were used for visual purposes, along with JavaScript for client-side validations. Additionally, PHP was utilized for server-side functionality. A styles.css file was created and linked on every page to maintain a consistent theme across all pages.

- **Session Management**: session_start() was used on all pages except the login and registration ones for session management. Session authentication was implemented to ensure users must log in before accessing other pages. A session hijacking prevention mechanism was also added where the application checks the session browser and HTTP user-agent to verify the user's identity and grants access only if they match, thereby preventing session hijacking by attackers.

- **CSRF Protection**: CSRF protection was included on two pages where users can update their information: changepasswordform.php and updateprofile.php. First, a random value was assigned to a new session variable. Then, a hidden element was added in the form that sends the CSRF token generated from this session. When the form is submitted, the page checks whether the CSRF token from the session matches the one passed in the form. If they match, the form is submitted; otherwise, it is rejected.

Appendix

/database-account.sql

```sql
create database waph_team;
create user 'team3'@'localhost' IDENTIFIED BY '1234';
grant ALL on waph_team.* TO 'team3'@'localhost';
```

/database-data.sql

```sql
-- if the table exists, delete it
DROP TABLE IF EXISTS users;
```

```
-- create a new table
CREATE TABLE users(
    username VARCHAR(100) PRIMARY KEY,
    password VARCHAR(100) NOT NULL,
    fullname VARCHAR(100),
    otheremail VARCHAR(100),
    phone VARCHAR(15),
    disabled BOOLEAN DEFAULT 0
);

-- insert data to the table users
INSERT INTO users(username, password, fullname, otheremail, phone) VALUES ('admin', MD5('123
INSERT INTO users(username, password, fullname, otheremail, phone) VALUES ('test', MD5('test
INSERT INTO users(username, password, fullname, otheremail, phone,disabled) VALUES ('test1',
INSERT INTO users(username, password, fullname, otheremail, phone,disabled) VALUES ('test2',

-- if the table exists, delete it
DROP TABLE IF EXISTS posts;

-- create a new table
CREATE TABLE posts(
    postID VARCHAR(100) PRIMARY KEY,
    title VARCHAR(100) NOT NULL,
    content VARCHAR(100),
    posttitle VARCHAR(100),
    owner VARCHAR(100),
    FOREIGN KEY (owner) REFERENCES users (username) ON DELETE CASCADE
);

-- if the table exists, delete it
DROP TABLE IF EXISTS comments;

-- create a new table
CREATE TABLE comments(
    commentID VARCHAR(100) PRIMARY KEY,
    content VARCHAR(255) NOT NULL,
    postID VARCHAR(100),
    commenter VARCHAR(100),
    FOREIGN KEY (postID) REFERENCES posts (postID) ON DELETE CASCADE,
    FOREIGN KEY (commenter) REFERENCES users (username) ON DELETE CASCADE
);

-- if the table exists, delete it
DROP TABLE IF EXISTS super_users;
```

```
-- create a new table
CREATE TABLE super_users (
    username VARCHAR(100) PRIMARY KEY,
    password VARCHAR(100) NOT NULL,
    fullname VARCHAR(100),
    email VARCHAR(100)
);


INSERT INTO super_users(username, password, fullname, email) VALUES ('super', MD5('123'), 'S
```

/style.css

```
/* Reset default margin and padding */
body,
html {
  margin: 0;
  padding: 0;
}

/* Global styles */
body {
  font-family: Arial, sans-serif;
  background-color: #f5f5f5;
  color: #333;
}

.container {
  background-color: #fff;
  border-radius: 8px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
  margin: 50px auto;
  max-width: 800px;
  padding: 20px;
}

h1,
h2 {
  text-align: center;
  color: #007bff;
}

#digit-clock {
  text-align: center;
  font-weight: bold;
  margin-bottom: 20px;
```

```
}

.form {
  text-align: center;
}

.text_field,
textarea,
.button {
  display: block;
  width: 100%;
  margin-bottom: 15px;
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 5px;
  box-sizing: border-box; /* Ensure padding is included in width */
}

.button {
  background-color: #007bff;
  border: none;
  color: #fff;
  cursor: pointer;
}

.button:hover {
  background-color: #0056b3;
}

.error-message {
  color: #dc3545;
  font-size: 14px;
  text-align: center;
  margin-top: 5px;
  display: block;
}

.post {
  background-color: #f9f9f9;
  border-radius: 5px;
  margin-bottom: 20px;
  padding: 10px;
}

.comment {
  font-size: 0.9em;
  margin-left: 20px;
}

.btn {
  display: inline-block;
  padding: 8px 12px;
  margin-right: 10px;
  background-color: #007bff;
  color: #fff;
  border-radius: 3px;
  text-decoration: none;
}

.btn:hover {
  background-color: #0056b3;
}

/* Specific styles for form.php */
.form-container {
  margin: 50px auto;
  max-width: 400px;
  padding: 20px;
}

/* Additional styles specific to index.php */
.comment-form textarea {
  width: calc(100% - 20px); /* Adjust width to account for padding */
}

ul {
  list-style: none;
  padding: 0;
}

ul li {
  margin-bottom: 10px;
}

ul li a {
  color: #007bff;
  text-decoration: none;
}

ul li a:hover {
  text-decoration: underline;
}
```

```
/* Styles specific to newpost.php */
form {
  max-width: 600px;
  margin: 0 auto;
  padding: 20px;
  background-color: #fff;
  border-radius: 8px;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

form label {
  display: block;
  font-weight: bold;
  margin-bottom: 10px;
}

form input[type="text"],
form textarea {
  width: 100%;
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 5px;
  box-sizing: border-box;
  margin-bottom: 15px;
}

form input[type="submit"] {
  background-color: #007bff;
  border: none;
  color: #fff;
  padding: 10px 20px;
  border-radius: 5px;
  cursor: pointer;
}

form input[type="submit"]:hover {
  background-color: #0056b3;
}

/* Styles specific to chat.php */
#messages {
  height: 400px;
  overflow-y: scroll;
  border: 1px solid #ccc;
  padding: 10px;
  border-radius: 5px;
  background-color: #f9f9f9;
  margin-bottom: 20px;
}

#messageBox {
  display: block;
  width: 100%;
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 5px;
  margin-bottom: 10px;
}

#send {
  width: 100%;
  height: 40px;
  background-color: #007bff;
  color: #fff;
  border: none;
  border-radius: 5px;
  cursor: pointer;
}

#send:hover {
  background-color: #0056b3;
}

.visited-time {
    text-align: center; /* Center-align the content */
    margin-top: 20px; /* Optional: Add top margin for spacing */
}
```

/server.js

```
const WebSocket = require('ws');

const wss = new WebSocket.Server({ port: 6969 });

wss.on('connection', function connection(ws) {
    console.log('A new client connected!');

    ws.on('message', function incoming(data) {
        console.log('Received: %s', data);

        // Broadcast the received message to all clients as a string
        const message = data.toString(); // Convert data to string
        wss.clients.forEach(function each(client) {
            if (client !== ws && client.readyState === WebSocket.OPEN) {
                client.send(message); // Send the message to other clients
            }
        });
    });

    ws.send('Welcome to the WebSocket server!'); // Send a welcome message to the connected
});

console.log('WebSocket server is running on ws://localhost:6969');
```

/database.php

```
<?php
$mysqli = new mysqli('localhost', 'team3', '1234', 'waph_team');

if($mysqli->connect_errno) {
    printf("Database connection failed: %s\n", $mysqli->connect_error);
    return FALSE;
}

function addNewUser($username, $password, $fullname, $otheremail) {
    global $mysqli;

    // Basic validation checks
    if (empty($username) || empty($password) || empty($fullname) || empty($otheremail)) {
        return false; // Ensures that no field is empty
    }

    if (!filter_var($otheremail, FILTER_VALIDATE_EMAIL)) {
        return false; // Ensures the email is in a valid format
    }

    // Here you can add additional validation as needed, e.g., minimum lengths
    if (strlen($password) < 8) {
        return false; // Password must be at least 8 characters
    }

    // Hash the password using a more secure method
    $hashedPassword = md5($password);

    $preparedSql = "INSERT INTO users (username, password, fullname, otheremail) VALUES (?,
    $stmt = $mysqli->prepare($preparedSql);
    if (!$stmt) {
        return false; // Could not prepare the statement
    }

    $stmt->bind_param("ssss", $username, $hashedPassword, $fullname, $otheremail); // Bind

    if ($stmt->execute()) {
        return true;
    } else {
        return false;
    }
}


function checklogin_mysql($username, $password) {
    global $mysqli;

    // Hash the password using md5 for comparison
    $hashed_password = md5($password);

    // Check if the user is enabled in the users table
    $prepared_sql_users = "SELECT * FROM users WHERE username=? AND password=?";
    $stmt_users = $mysqli->prepare($prepared_sql_users);
    $stmt_users->bind_param("ss", $username, $hashed_password);
    $stmt_users->execute();
    $result_users = $stmt_users->get_result();

    // If the user is found in the users table
    if($result_users->num_rows == 1) {
        $user = $result_users->fetch_assoc();
        if ($user['disabled'] == 1) {
            // User is disabled
            return 'disabled';
        } else {
            // User is enabled, determine user type and return it
            return 'regularuser';
        }
    }

    // Check in the super_users table
    $prepared_sql_superusers = "SELECT * FROM super_users WHERE username=? AND password=?";
    $stmt_superusers = $mysqli->prepare($prepared_sql_superusers);
    $stmt_superusers->bind_param("ss", $username, $hashed_password);
    $stmt_superusers->execute();
    $result_superusers = $stmt_superusers->get_result();

    // If the user is found in the super_users table
    if($result_superusers->num_rows == 1) {
        return 'superuser'; // Return 'superuser' if the user is a superuser
    }

    return FALSE; // Return FALSE if the user is not found in either table
}

// Function to check if a user is disabled
function isUserDisabled($username) {
    global $mysqli;
    $stmt = $mysqli->prepare("SELECT disabled FROM users WHERE username = ?");
    $stmt->bind_param("s", $username);
    $stmt->execute();
    $stmt->bind_result($disabled);
    $stmt->fetch();
    $stmt->close();
    return $disabled; // Returns true if the user is disabled, false otherwise
}


function updatePassword($username, $newPassword) {
    $mysqli = new mysqli('localhost','team3','1234','waph_team');

    if ($mysqli->connect_errno) {
        printf("Connect failed: %s\n", $mysqli->connect_error);
        exit();
    }
    $hashed_password = md5($newPassword);

    $prepared_sql = "UPDATE users SET password = ? WHERE username = ?";
    $stmt = $mysqli->prepare($prepared_sql);
    $stmt->bind_param("ss", $hashed_password, $username);

    if ($stmt->execute()) {
        $stmt->close();
        $mysqli->close();
        return true;
    } else {
        $stmt->close();
        $mysqli->close();
        return false;
    }
}

// Function to update user profile
function updateUserProfile($username, $fullname, $otheremail, $phone) {
    $mysqli = new mysqli('localhost','team3','1234','waph_team');

    if ($mysqli->connect_errno) {
        printf("Connect failed: %s\n", $mysqli->connect_error);
        exit();
    }
    $prepared_sql = "UPDATE users SET fullname = ?, otheremail = ?, phone = ? WHERE username
    $stmt = $mysqli->prepare($prepared_sql);
    $stmt->bind_param("ssss", $fullname, $otheremail, $phone, $username);

    if ($stmt->execute()) {
        return true;
    } else {
        return false;
    }
}


// function fetchPosts() {
//     global $mysqli;
//     $posts = array();

//     $sql = "SELECT * FROM posts";
//     $result = $mysqli->query($sql);

//     if ($result->num_rows > 0) {
//         while ($row = $result->fetch_assoc()) {
//             $posts[] = $row;
//         }
//     }

//     return $posts;
// }

function fetchPosts() {
    global $mysqli;
    $posts = array();

    // Prepare the SQL query with a parameter placeholder
    $sql = "SELECT * FROM posts";
    $stmt = $mysqli->prepare($sql);

    if ($stmt) {
        // Execute the prepared statement
        $stmt->execute();
        // Get the result set
        $result = $stmt->get_result();

        if ($result->num_rows > 0) {
            // Fetch rows as associative array and add them to $posts array
            while ($row = $result->fetch_assoc()) {
                $posts[] = $row;
            }
        }

        // Close the statement
        $stmt->close();
    }

    return $posts;
}


// Function to fetch a post by its ID from the database
function fetchPostById($postID) {
    global $mysqli;

    // Prepare and execute query to fetch the post
    $stmt = $mysqli->prepare("SELECT title, content, owner FROM posts WHERE postID = ?");
    $stmt->bind_param("s", $postID);
    $stmt->execute();
    $stmt->bind_result($title, $content, $owner);

    // Fetch the result
    $stmt->fetch();

    // Create an associative array to hold the post details
    $post = array(
        'title' => $title,
        'content' => $content,
        'owner' => $owner
    );

    // Close the statement
    $stmt->close();

    // Return the post details
    return $post;
}
```

```
    function addNewPost($title, $content, $username) {
    // Perform input validation (you can add more validation as needed)
        global $mysqli;
    if (empty($title) || empty($content)) {
        return "Please fill out all fields.";
    } else {
        // Insert the new post into the database
        $postID = uniqid(); // Generate a unique postID
        $owner = $username; // Assuming the username is passed to the function
        $sql = "INSERT INTO posts (postID, title, content, owner) VALUES (?, ?, ?, ?)";

        if ($stmt = $mysqli->prepare($sql)) {
            $stmt->bind_param("ssss", $postID, $title, $content, $owner);
            if ($stmt->execute()) {
                $stmt->close();
                return "Post added successfully!";
            } else {
                $stmt->close();
                return "Error adding post: " . $mysqli->error;
            }
        } else {
            return "Error preparing statement: " . $mysqli->error;
        }
    }
}

// Function to update a post in the database
function updatePost($postID, $title, $content) {
    global $mysqli;

    // Prepare and execute query to update the post
    $stmt = $mysqli->prepare("UPDATE posts SET title = ?, content = ? WHERE postID = ?");
    $stmt->bind_param("sss", $title, $content, $postID);
    $stmt->execute();
    $stmt->close();
}

function deletePost($postID) {
    global $mysqli;

    // Prepare and execute query to delete the post
    $stmt = $mysqli->prepare("DELETE FROM posts WHERE postID = ?");
    $stmt->bind_param("s", $postID);
    $stmt->execute();

    // Check for errors
    if ($stmt->error) {
        echo "Error deleting post: " . $stmt->error;
    } else {
        echo "Post deleted successfully.";
    }

    $stmt->close();
}

function fetchComments($mysqli, $postID) {
    // Prepare SQL query to select comments for the specified postID
    $query = "SELECT content, commenter FROM comments WHERE postID = ?";

    // Prepare the SQL statement
    $stmt = $mysqli->prepare($query);

    // Bind the postID parameter to the prepared statement
    $stmt->bind_param("s", $postID);

    // Execute the prepared statement
    $stmt->execute();

    // Get result set from the executed statement
    $result = $stmt->get_result();

    // Initialize an empty array to store comments
    $comments = [];

    // Fetch each row from the result set
    while ($row = $result->fetch_assoc()) {
        // Add the fetched comment to the comments array
        $comments[] = $row;
    }

    // Close the prepared statement
    $stmt->close();

    // Return the array of comments
    return $comments;
}

?>
```

/addnewpost.php

```php
<?php
require "database.php";
session_start(); // Ensure session is started before using $_SESSION

$title = $_POST["title"] ?? ''; // Use null coalescing operator to avoid undefined index no
$content = $_POST["content"] ?? '';
$username = $_SESSION['username'] ?? ''; // Fallback to empty string if not set

// Check if both title and content are provided
if (!empty($title) && !empty($content)) {
    // Assuming addNewPost is a function defined in database.php or another included file
    if (addNewPost($title, $content, $username)) {
        echo "Post added";
    } else {
        echo "Failed";
    }
} else {
    echo "Please fill out all fields.";
}
?>
```

/deletepost.php

```php
<?php
// Start session
session_start();

// Include database configuration
require "database.php";

// Function to check if a post belongs to a specific user
function checkPostOwner($username, $postID) {
    global $mysqli;

    // Prepare and execute query to fetch the owner of the post
    $stmt = $mysqli->prepare("SELECT owner FROM posts WHERE postID = ?");
    $stmt->bind_param("s", $postID);
    $stmt->execute();

    // Check for errors
    if ($stmt->error) {
        echo "Error: " . $stmt->error;
        return false;
    }
    $stmt->bind_result($owner);
    $stmt->fetch();
    $stmt->close();

    // Check if the post exists and its owner matches the given username
    if ($owner === $username) {
        return true; // Post belongs to the user
    } else {
        return false; // Post does not belong to the user or does not exist
    }
}

// Check if user is logged in
if (!isset($_SESSION['authenticated']) || $_SESSION['authenticated'] !== true) {
    // If not logged in, destroy session and redirect to login form
    session_destroy();
    echo "<script>alert('You have not logged in, please login first!')</script>";
    header("Refresh: 0; url=form.php");
    exit();
}

// Check for session hijacking
if ($_SESSION['browser'] != $_SERVER["HTTP_USER_AGENT"]) {
    // If session hijacking is detected, destroy session and redirect to login form
    session_destroy();
    echo "<script>alert('Session hijacking is detected')</script>";
    header("Refresh: 0; url=form.php");
    exit();
}

// Handle post deletion
if (isset($_POST['delete']) && isset($_POST['postID'])) {
    $postID = $_POST['postID'];
    // Check if the post belongs to the current user
    if (checkPostOwner($_SESSION['username'], $postID)) {
        // Delete the post
        deletePost($postID);
        echo "<script>alert('Post deleted successfully.')</script>";
    } else {
        echo "<script>alert('You are not authorized to delete this post.')</script>";
    }
}

// Redirect to the main page
header("Location: index.php");
```

```
exit();
?>
```

/addnewuser.php

```
<?php
require "database.php";

// Initialize variables to null
$username = $password = $fullname = $primaryemail = "";

// Check if the form is submitted
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Basic validation
    $username = test_input($_POST["username"]);
    $password = test_input($_POST["password"]);
    $fullname = test_input($_POST["fullname"] ?? ''); // Using null coalescing operator
    $primaryemail = test_input($_POST["email"] ?? '');

    // Validate username and password
    if (empty($username) || empty($password)) {
        echo "No username/password provided";
    } elseif (strlen($password) < 8) {
        echo "Password must be at least 8 characters long";
    } elseif (!filter_var($primaryemail, FILTER_VALIDATE_EMAIL)) {
        echo "Invalid email format";
    } else {
        // Attempt to register the user
        $success = addNewUser($username, $password, $fullname, $primaryemail);
        echo $success ? "Registration Succeed" : "Registration Failed";
    }
} else {
    // Form not submitted
    echo "Please submit the form";
}

// Function to sanitize input data
function test_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}
?>

<!DOCTYPE html>
```

```
<html>
<head>
    <title>User Registration</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container">
        <br>
        <a href="form.php" class="login-link">Login here</a>
    </div>
</body>
</html>
```

/registrationform.php

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>WAPH-Registration page</title>
   <link rel="stylesheet" href="style.css">
  <script type="text/javascript">
      function displayTime() {
        document.getElementById('digit-clock').innerHTML = "Current time:" + new Date();
      }
      setInterval(displayTime, 500);

      function validateForm() {
      var fullname = document.getElementById('fullname').value.trim();
      var email = document.getElementById('email').value.trim();
      var username = document.getElementById('username').value.trim();
      var password = document.getElementById('password').value;
      var confirmPassword = document.getElementById('confirmPassword').value;
      var errorElement = document.getElementById('error-message');

      // Client-side validation for password match
      if (password !== confirmPassword) {
        errorElement.textContent = "Passwords do not match";
        return false;
      }

      // Client-side validation for preventing XSS (Cross-Site Scripting)
      var htmlPattern = /<\/?[a-z][\s\S]*>/i; // Regex to detect HTML tags
      if (htmlPattern.test(fullname) || htmlPattern.test(email) || htmlPattern.test(username
        errorElement.textContent = "Invalid characters detected";
        return false;
      }

      // Client-side validation for email format
      var emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
      if (!emailPattern.test(email)) {
        errorElement.textContent = "Invalid email format";
        return false;
      }-

      // Clear error message if all validations pass
      errorElement.textContent = "";
      return true;
    }
  </script>
</head>
<body>
  <div class="container">
    <h1>Registration form, WAPH</h1>
    <h2>Team-3</h2>
    <div id="digit-clock"></div>
    <?php
      // PHP code to display visited time
      echo "Visited time: " . date("Y-m-d h:i:sa");
    ?>
    <form action="addnewuser.php" method="POST" class="form login" onsubmit="return validate
      <input type="text" class="text_field" id="fullname" name="fullname" placeholder="Full
      <input type="text" class="text_field" id="email" name="email" placeholder="Email" requ
      <input type="text" class="text_field" id="username" name="username" placeholder="Usern
      <input type="password" class="text_field" id="password" name="password" placeholder="P
      <input type="password" class="text_field" id="confirmPassword" placeholder="Confirm Pa
      <span class="error-message" id="error-message"></span>
      <button class="button" type="submit">Register</button>
    </form>
  </div>
  <div class="container">
        <br>
        <a href="login.php" class="login-link">Already a user? Login here</a>
  </div>

</body>
</html>
```

/profile.php

```
<?php
```

```php
session_start();


if (!isset($_SESSION['authenticated']) or $_SESSION['authenticated'] != TRUE) {
        session_destroy();
        echo "<script>alert('Nice try!! You have to login first!')</script>";
        header("Refresh: 0; url=login.php");
        die();
    }

    if ($_SESSION['browser'] != $_SERVER["HTTP_USER_AGENT"]) {
    session_destroy();
    echo "<script>alert('Alert! Alert ! Session hijacking is detected')</script>";
    header("Refresh: 0; url=login.php");
    die();
}
require "database.php";

if (!isset($_SESSION['nocsrftoken'])) {
    $_SESSION['nocsrftoken'] = bin2hex(openssl_random_pseudo_bytes(32)); // Generate a rando
}

// Validate CSRF token on form submission
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $tokenFromForm = $_POST['nocsrftoken'] ?? '';
    if (!hash_equals($_SESSION['nocsrftoken'], $tokenFromForm)) {
        die("CSRF Token Validation Failed.");
    }
}


// Retrieve user's current profile data
$username = ""; // Initialize variables to store current user's data
$fullname = "";
$otheremail = "";
$phone = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (isset($_POST['username']) && isset($_POST['fullname']) && isset($_POST['otheremail'
        $username = $_POST['username'];
        $fullname = $_POST['fullname'];
        $otheremail = $_POST['otheremail'];
        $phone = $_POST['phone'];

        // Update user's profile
        if (updateUserProfile($username, $fullname, $otheremail, $phone)) {
            echo "Profile updated successfully.";
        } else {
            echo "Failed to update profile.";
        }
    } else {
        echo "All fields are required.";
    }
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Edit Profile</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <h2>Edit Profile</h2>
    <form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>">
        <label>Username:</label><br>
        <input type="text" name="username" value="<?php echo $username; ?>" ><br>
        <label>Full Name:</label><br>
        <input type="text" name="fullname" value="<?php echo $fullname; ?>"><br>
        <label>Additional Email:</label><br>
        <input type="text" name="otheremail" value="<?php echo $otheremail; ?>"><br>
        <label>Phone:</label><br>
        <input type="text" name="phone" value="<?php echo $phone; ?>"><br><br>
        <input type="hidden" name="nocsrftoken" value="<?php echo htmlspecialchars($_SESSION
        <input type="submit" value="Update Profile">
    </form>
</body>
</html>
```

/form.php

```
<?php
session_start();
if (!isset($_SESSION['nocsrftoken'])) {
    $_SESSION['nocsrftoken'] = bin2hex(openssl_random_pseudo_bytes(32)); // Generate a rand
}
?>

<!DOCTYPE html>
<html lang="en">
```

```
<head>
  <meta charset="utf-8">
  <title>WAPH-Login page</title>
  <link rel="stylesheet" href="style.css">

  <script type="text/javascript">
      function displayTime() {
        document.getElementById('digit-clock').innerHTML = "Current time:" + new Date();
      }
      setInterval(displayTime, 500);


      function validateForm() {

      var username = document.getElementById('username').value.trim();
      var password = document.getElementById('password').value;
      var errorElement = document.getElementById('error-message');

      // Check if username is empty
        if (username === "") {
            errorElement.textContent = "Please enter your username.";
            return false; // Prevent form from submitting
        }

        // Check if password is empty
        if (password === "") {
            errorElement.textContent += (errorElement.textContent.length > 0 ? "\n" : "") +
            return false; // Prevent form from submitting
        }

          // Clear error message if all validations pass
          errorElement.textContent = "";
          return true;
        }
  </script>
</head>
<body>
  <div class="container">
    <h1>A Simple login form, WAPH</h1>
    <h2>Team-3</h2>
    <div id="digit-clock"></div>

    <div class="visited-time">
    <?php echo "Visited time: " . date("Y-m-d h:i:sa"); ?>
    </div>
    <form action="index.php" method="POST" class="form login" onsubmit="return validateForm(
      Username: <input type="text" class="text_field" name="username" /> <br>
      Password: <input type="password" class="text_field" name="password" /> <br>
      <span class="error-message" id="error-message"></span>
      <input type="hidden" name="nocsrftoken" value="<?php echo $_SESSION['nocsrftoken']; ?
      <button class="button" type="submit">Login</button>
    </form>
  </div>
</body>
</html>
```

/chat.php

```
<?php

session_start();
if (!isset($_SESSION['authenticated']) or $_SESSION['authenticated'] != TRUE) {
        session_destroy();
        echo "<script>alert('Nice try!! You have to login first!')</script>";
        header("Refresh: 0; url=login.php");
        die();
    }

    if ($_SESSION['browser'] != $_SERVER["HTTP_USER_AGENT"]) {
    session_destroy();
    echo "<script>alert('Alert! Alert ! Session hijacking is detected')</script>";
    header("Refresh: 0; url=login.php");
    die();
}

?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Live Chat</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
<h1>Group Live Chat</h1>
<pre id="messages" style="height: 400px; overflow: scroll"></pre>
<input type="text" id="messageBox" placeholder="Type your message here" style="display: bloc
<button id="send" title="Send Message!" style="width: 100%; height: 30px;">Send Message</but

<script>
```

```
(function() {
  const sendBtn = document.querySelector('#send');
  const messages = document.querySelector('#messages');
  const messageBox = document.querySelector('#messageBox');

  let ws;

  function showMessage(message) {
    messages.textContent += `\n\n${message}`;
    messages.scrollTop = messages.scrollHeight;
    messageBox.value = '';
  }

  function init() {
    if (ws) {
      ws.onerror = ws.onopen = ws.onclose = null;
      ws.close();
    }

    ws = new WebSocket('ws://localhost:6969');
    ws.onopen = () => {
      console.log('Connection opened!');
    }
    ws.onmessage = (event) => {
      const messageData = JSON.parse(event.data);
      showMessage(`${messageData.sender}: ${messageData.message}`);
    };
    ws.onclose = function() {
      ws = null;
    }
  }

  sendBtn.onclick = function() {
    if (!ws) {
      showMessage("No WebSocket connection :(");
      return;
    }

    const message = messageBox.value;
    if (message.trim() === '') return;

    const messageData = {
      sender: '<?php echo $_SESSION['username']; ?>',
      message: message
    };
```

```
   ws.send(JSON.stringify(messageData));
   showMessage(`${messageData.sender}: ${messageData.message}`);
  }

  init();
 })();
</script>
</body>
</html>
```

/changepasswordform.php

```
<?php

session_start();

if (!isset($_SESSION['authenticated']) or $_SESSION['authenticated'] != TRUE) {
    session_destroy();
    echo "<script>alert('Nice try!! You have to login first!')</script>";
    header("Refresh: 0; url=login.php");
    die();
  }

  if ($_SESSION['browser'] != $_SERVER["HTTP_USER_AGENT"]) {
  session_destroy();
  echo "<script>alert('Alert! Alert ! Session hijacking is detected')</script>";
  header("Refresh: 0; url=login.php");
  die();
}


// Generate and store CSRF token in the session if it doesn't exist
if (!isset($_SESSION['nocsrftoken'])) {
    $_SESSION['nocsrftoken'] = bin2hex(openssl_random_pseudo_bytes(32)); // Generate a rando
}

// Validate CSRF token on form submission
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $tokenFromForm = $_POST['nocsrftoken'] ?? '';
    if (!hash_equals($_SESSION['nocsrftoken'], $tokenFromForm)) {
        die("CSRF Token Validation Failed.");
    }
}


require "database.php";
```

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Check if all fields are filled
    if (isset($_POST['username']) && isset($_POST['current_password']) && isset($_POST['new_
        // Fetch the user's current password from the database
        $stmt = $mysqli->prepare("SELECT password FROM users WHERE username = ?");
        $stmt->bind_param("s", $_POST['username']);
        $stmt->execute();
        $result = $stmt->get_result();
        if ($result->num_rows == 1) {
            $row = $result->fetch_assoc();
            $current_password_from_database = $row['password'];

            // Check if the entered current password matches the one in the database
            if (md5($_POST['current_password']) == $current_password_from_database) {
                // Update password
                if (updatePassword($_POST['username'], $_POST['new_password'])) {
                    echo "Password updated successfully.";
                } else {
                    echo "Failed to update password.";
                }
            } else {
                echo "Current password is incorrect.";
            }
        } else {
            echo "User not found.";
        }
        $stmt->close();
    } else {
        echo "All fields are required.";
    }
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Change Password</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <h2>Change Password</h2>
    <form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>">
        <label>Username:</label><br>
        <input type="text" name="username"><br>
        <label>Current Password:</label><br>
        <input type="password" name="current_password"><br>
        <label>New Password:</label><br>
        <input type="password" name="new_password"><br><br>
        <input type="hidden" name="nocsrftoken" value="<?php echo htmlspecialchars($_SESSION
        <input type="submit" value="Change Password">
    </form>
</body>
</html>
```

/editpost.php

```
<?php
session_start();


if (!isset($_SESSION['authenticated']) or $_SESSION['authenticated'] != TRUE) {
        session_destroy();
        echo "<script>alert('Nice try!! You have to login first!')</script>";
        header("Refresh: 0; url=login.php");
        die();
    }

    if ($_SESSION['browser'] != $_SERVER["HTTP_USER_AGENT"]) {
    session_destroy();
    echo "<script>alert('Alert! Alert ! Session hijacking is detected')</script>";
    header("Refresh: 0; url=login.php");
    die();
}


if (!isset($_SESSION['nocsrftoken'])) {
    $_SESSION['nocsrftoken'] = bin2hex(openssl_random_pseudo_bytes(32)); // Generate a rand
}

// Validate CSRF token on form submission
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $tokenFromForm = $_POST['nocsrftoken'] ?? '';
    if (!hash_equals($_SESSION['nocsrftoken'], $tokenFromForm)) {
        die("CSRF Token Validation Failed.");
    }
}

// Include database configuration
require "database.php";
```

```
// Check if user is logged in
if (!isset($_SESSION['authenticated']) || $_SESSION['authenticated'] !== true) {
    header("Location: form.php");
    exit;
}

// Check if post ID is provided
if (!isset($_POST['postID'])) {
    header("Location: index.php");
    exit;
}

// Get post ID from the form submission
$postID = $_POST['postID'];

// Fetch post details from the database
$post = fetchPostById($postID);

// Check if the post exists
if (!$post) {
    echo "Post not found.";
    exit;
}

// Check if the current user is the owner of the post
if ($_SESSION['username'] !== $post['owner']) {
    echo "You are not authorized to edit this post.";
    exit;
}

// Handle update action if form is submitted
if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['update'])) {
    $title = $_POST['title'];
    $content = $_POST['content'];
    updatePost($postID, $title, $content);
    header("Location: index.php");
    exit;
}

?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Edit Post</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <h1>Edit Post</h1>
    <form method="post">
        <label for="title">Title:</label><br>
        <input type="text" id="title" name="title" value="<?php echo htmlspecialchars($post[
        <label for="content">Content:</label><br>
        <textarea id="content" name="content"><?php echo htmlspecialchars($post['content']);
        <input type="hidden" name="postID" value="<?php echo $postID; ?>">
          <input type="hidden" name="nocsrftoken" value="<?php echo htmlspecialchars($_SESSI
        <button type="submit" name="update">Update</button>
    </form>
</body>
</html>
```

/index.php

```
<?php
// Start session
session_start();
if (!isset($_SESSION['nocsrftoken'])) {
    $_SESSION['nocsrftoken'] = bin2hex(openssl_random_pseudo_bytes(32)); // Generate a rand
}

// Set session cookie parameters
session_set_cookie_params([
    'lifetime' => 15 * 60,
    'path' => '/',
    'domain' => 'waph-team3.minifacebook.com',
    'secure' => true,
    'httponly' => true
]);


if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $tokenFromForm = $_POST['nocsrftoken'] ?? '';
    if (!hash_equals($_SESSION['nocsrftoken'], $tokenFromForm)) {
        die("CSRF Token Validation Failed.");
    }
}



require "database.php";
```

```
// Function to add a comment
if ($_SERVER['REQUEST_METHOD'] == 'POST' && isset($_POST['comment']) && isset($_POST['postI
    $comment = $_POST['comment'];
    $postID = $_POST['postID'];
    $commenter = $_SESSION['username']; // Username from session
    $stmt = $mysqli->prepare("INSERT INTO comments (commentID, content, postID, commenter) V
    $stmt->bind_param("sss", $comment, $postID, $commenter);
    $stmt->execute();
    $stmt->close();
    header("Location: index.php"); // Redirect to avoid resubmission
    exit;
}

// Check if login credentials are provided
if (isset($_POST["username"]) && isset($_POST["password"])) {
    $login_type = checklogin_mysql($_POST["username"], $_POST["password"]);
    if ($login_type === 'regularuser' || $login_type === 'superuser') {
        // If login is successful, set session variables
        $_SESSION['authenticated'] = true;
        $_SESSION['username'] = $_POST["username"];
        $_SESSION['browser'] = $_SERVER["HTTP_USER_AGENT"];
        $_SESSION['usertype'] = $login_type; // Set session variable for user type
    } elseif ($login_type === 'disabled') {
        // If the user is disabled, show a message and redirect to login form
        echo "<script>alert('Your account is disabled. Please contact the administrator.');<
        header("Refresh: 0; url=form.php");
        exit();
    } else {
        // If login fails (user not found or incorrect password), destroy session and show
        session_destroy();
        echo "<script>alert('Invalid username/password');</script>";
        header("Refresh: 0; url=form.php");
        exit();
    }
}

if (isset($_POST["username"]) and isset($_POST["password"])){
        $username = htmlspecialchars($_POST["username"]); // Sanitize input
        $password = htmlspecialchars($_POST["password"]); // Sanitize input

        if (checklogin_mysql($username,$password)) {
            $_SESSION['authenticated'] = TRUE;
            $_SESSION['username'] = $_POST["username"];
            $_SESSION['browser'] = $_SERVER["HTTP_USER_AGENT"];
        }else{
            session_destroy();
            echo "<script>alert('Invalid password/username');window.location='form.php';</sc
            die();
        }
    }
// Check if user is logged in
if (!isset($_SESSION['authenticated']) || $_SESSION['authenticated'] !== true) {
    // If not logged in, destroy session and redirect to login form
    session_destroy();
    echo "<script>alert('You have not logged in, please login first!')</script>";
    header("Refresh: 0; url=form.php");
    die();
}

// Check for session hijacking
if ($_SESSION['browser'] != $_SERVER["HTTP_USER_AGENT"]) {
    // If session hijacking is detected, destroy session and redirect to login form
    session_destroy();
    echo "<script>alert('Session hijacking is detected')</script>";
    header("Refresh: 0; url=form.php");
    die();
}

// Function to disable a user
function disableUser($username) {
    global $mysqli;
    $stmt = $mysqli->prepare("UPDATE users SET disabled = 1 WHERE username = ?");
    $stmt->bind_param("s", $username);
    $stmt->execute();
    $stmt->close();
}

// Function to enable a user
function enableUser($username) {
    global $mysqli;
    $stmt = $mysqli->prepare("UPDATE users SET disabled = 0 WHERE username = ?");
    $stmt->bind_param("s", $username);
    $stmt->execute();
    $stmt->close();
}

// Check if action is requested and the user is a superuser
if ($_SESSION['usertype'] === 'superuser' && isset($_GET['action']) && isset($_GET['username
    $action = $_GET['action'];
    $username = $_GET['username'];
    if ($action === 'disable') {
        disableUser($username);
    } elseif ($action === 'enable') {
        enableUser($username);
    }
}



function fetchUsers() {
    global $mysqli;
    $users = [];

    // Prepare the SQL query with a parameter placeholder
    $sql = "SELECT username FROM users";
    $stmt = $mysqli->prepare($sql);

    if ($stmt) {
        // Execute the prepared statement
        $stmt->execute();

        // Bind result variables
        $stmt->bind_result($username);

        // Fetch rows and store usernames in $users array
        while ($stmt->fetch()) {
            $users[] = $username;
        }

        // Close the statement
        $stmt->close();
    }

    return $users;
}



?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Welcome</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container">
        <h2>Welcome <?php echo htmlentities($_SESSION['username']); ?>!</h2>

        <h2>Posts</h2>
        <hr>
        <?php
        // Fetch and display posts
        $posts = fetchPosts($mysqli);
        if (!empty($posts)) {
            foreach ($posts as $post) {
                echo "<div class='post'>";
                echo "<h3>Title: " . $post['title'] . "</h3>";
                echo "<p>Content: " . $post['content'] . "</p>";
                echo "<p>Posted by: " . $post['owner'] . "</p>";
                // Show edit and delete buttons only for the owner of the post
                if ($_SESSION['username'] === $post['owner']) {
                    echo "<form method='post' action='editpost.php'>";
                    echo '<input type="hidden" name="nocsrftoken" value="' . htmlspecialchar
                    echo "<input type='hidden' name='postID' value='" . $post['postID'] . "'
                    echo "<button class='btn' type='submit' name='edit'>Edit</button>";
                    echo "</form>";

                    echo "<form method='post' action='deletepost.php' onsubmit='return confi
                    echo '<input type="hidden" name="nocsrftoken" value="' . htmlspecialchar
                    echo "<input type='hidden' name='postID' value='" . $post['postID'] . "'
                    echo "<button class='btn' type='submit' name='delete'>Delete</button>";
                    echo "</form>";
                }

                // Display comments
                $comments = fetchComments($mysqli, $post['postID']);
                if ($comments) {
                    echo "<div>Comments:</div>";
                    foreach ($comments as $comment) {
                        echo "<div class='comment'><strong>" . htmlentities($comment['commen
                    }
                }

                // Add comment form
                echo "<form class='comment-form' method='post' action=''>";
                echo '<input type="hidden" name="nocsrftoken" value="' . htmlspecialchars($_
                echo "<input type='hidden' name='postID' value='" . $post['postID'] . "'>";
                echo "<textarea name='comment' rows='2' cols='50' placeholder='Write a comme
                echo "<button class='btn' type='submit'>Add Comment</button>";
                echo "</form>";

                echo "</div>";
            }
        } else {
            echo "<p>No posts found.</p>";
        }
        ?>
        <hr>
        <a class="btn" href="changepasswordform.php">Change Password</a>
        <a class="btn" href="profile.php">Edit Profile</a>
        <a class="btn" href="logout.php">Logout</a>
        <a class="btn" href="newpost.php">Add Post</a>
        <a class="btn" href="chat.php">Chat Room</a>

        <!-- Additional functionality for superuser -->
        <?php if ($_SESSION['usertype'] === 'superuser'): ?>
            <h2>Manage Users</h2>
            <h3>User List</h3>
            <ul>
                <?php $users = fetchUsers();
                foreach ($users as $user) {
                    $disabled = isUserDisabled($user); // Check if the user is disabled
                    echo "<li>$user ";
                    if ($user !== $_SESSION['username']) { // Exclude the current user from
                        if ($disabled) {
                            // If the user is disabled, display the enable option
                            echo "<a href='?action=enable&username=$user'>Enable</a>";
                        } else {
                            // If the user is enabled, display the disable option
                            echo "<a href='?action=disable&username=$user'>Disable</a>";
                        }
                    }
                    echo "</li>";
                } ?>
            </ul>
        <?php endif; ?>

    </div>
</body>
</html>
```

/newpost.php

```
<?php


session_start();


if (!isset($_SESSION['authenticated']) or $_SESSION['authenticated'] != TRUE) {
        session_destroy();
        echo "<script>alert('Nice try!! You have to login first!')</script>";
        header("Refresh: 0; url=login.php");
        die();
    }

    if ($_SESSION['browser'] != $_SERVER["HTTP_USER_AGENT"]) {
    session_destroy();
    echo "<script>alert('Alert! Alert ! Session hijacking is detected')</script>";
    header("Refresh: 0; url=login.php");
    die();
}
require "database.php";

if (!isset($_SESSION['nocsrftoken'])) {
    $_SESSION['nocsrftoken'] = bin2hex(openssl_random_pseudo_bytes(32)); // Generate a rando
}

// Validate CSRF token on form submission
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $tokenFromForm = $_POST['nocsrftoken'] ?? '';
    if (!hash_equals($_SESSION['nocsrftoken'], $tokenFromForm)) {
        die("CSRF Token Validation Failed.");
    }
}

?>


<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>New Post</title>
    <link rel="stylesheet" href="style.css">
</head>
```

```
<body>
    <h1>New Post</h1>
    <form action="addnewpost.php" method="POST" onsubmit="return validateForm()">
        <label for="title">Title:</label><br>
        <input type="text" id="title" name="title" required><br>
        <label for="content">Content:</label><br>
        <textarea id="content" name="content" rows="4" required></textarea><br>
        <input type="hidden" name="nocsrftoken" value="<?php echo htmlspecialchars($_SESSION
        <input type="submit" value="Submit">
    </form>

    <script>
        function validateForm() {
            var title = document.getElementById("title").value;
            var content = document.getElementById("content").value;
            if (title.trim() === "" || content.trim() === "") {
                alert("Please fill out all fields.");
                return false;
            }
            return true;
        }
    </script>
</body>
</html>
```

/logout.php

```
<?php
session_start();
session_destroy();
?>
<p> You are logout! </p>

<a href="form.php">Login again</a>
```