**WAPH-Web Application Programming and Hacking**

**Instructor: Dr. Phu Phung**

**Mini Facebook**

**Team – 01**

**Team members**

1. Tulasiram Nakkanaboina, [nakkantm@mail.uc.edu](mailto:nakkantm@mail.uc.edu)

2. Vihasith Rasala, [rasalavh@mail.uc.edu](mailto:rasalavh@mail.uc.edu)

3. Grahika Rampudi, [Rampudga@mail.uc.edu](mailto:Rampudga@mail.uc.edu)

4. Sai Sandeep Pasham, [pashamsp@mail.uc.edu](mailto:pashamsp@mail.uc.edu)

*Team members respective headshots*

A person with a mustache and a black shirt

Description automatically generatedA person wearing glasses and a suit jacket

Description automatically generated A person with long hair wearing glasses

Description automatically generated A person in a blue shirt

Description automatically generated

**Project Management Information**

Source code repository (private access):[**Click\_here**](https://github.com/waph-team01/waph-teamproject)

Project homepage (public): [**Click\_here**](https://github.com/waph-team01/waph-team01.github.io)

**Revision History**

|  |  |  |
| --- | --- | --- |
| Date | Version | Description |
| 26/03/2024 | 0.0 | Initial draft |
| 26/03/2024 | 0.1 | Drafting template overview |
| 26/03/2024 | 1.0 | Sprint 1 update |

**Overview**

This Project aims to develop a Mini Facebook web application using full stack web

development technologies, secure programming/hacking principles, and the practices of

agile development. This is a team project where all the team members collaborated and

worked together to create a database, creating simple login page and created an index.html

containing information about course overview, each member headshot and each member

personal portfolio link. We divided the tasks among us and pushed the code files into the

main branch.

**System Analysis**

During the initial phase that is for the sprint 0 of the project, we constructed the website utilizing the system configuration file, which was utilized and modified for the WAPH-team project file. The hosts file in the etc folder was updated with the public URL: [waph-team01.minifacebook.com](https://waph-team01.minifacebook.com/) alongside the IP address.

**Demo screenshots**

A screenshot of a computer

Description automatically generated

Figure 1 login admin page

A screenshot of a computer

Description automatically generated

Figure 2 change password form

A screenshot of a computer

Description automatically generated

Figure 3 password changed successfully

A screenshot of a computer

Description automatically generated

Figure 4 edit user profile

A screenshot of a computer

Description automatically generated

Figure 5 update profile

A screenshot of a computer

Description automatically generated

Figure 6 login page

A screenshot of a computer

Description automatically generated

Figure 7 new user registration form

A screenshot of a computer

Description automatically generated

Figure 8 succesful registation form

A screenshot of a computer

Description automatically generated

Figure 9 tables which stores the data

A screenshot of a computer

Description automatically generated

Figure 10 user tables

**Functional Requirements**

* **Features for posting like images and texts:** Enabling users to create and share posts with images/texts.
* **Database setup:** Configuring database system for storing application data securely.
* **Notification add on:** Implementing notification system for relevant user alerts.
* **Forms for authentication and user registration:** Implementing secure login and registration forms for user access.
* **Search functionality:** Enabling users to search for specific content within application.
* **Messaging features:** Integrating private messaging functionalities for user communication.

**Non - Functional Requirements**

* **Error handling:** Managing errors for smoother user experience and application stability.
* **Implementing security features:** Strengthening application security through implementation of various security measures.
* **Cookies:** Utilizing cookies for managing session information and enhancing user experience.
* **Session information:** Handling session data to maintain user context and interactions.

Front-End:

• In the front end we have implemented the new features that the user can be able to

change the password if they have any issues with the current.

• And also, we have implemented the edit profile option where the user will be able to

edit their full name, email and phone number respectively.

• We have implemented the posts visibility option. Once the user was login the user

will be able to see the posts that are posted over the site.

Back-End:

• We have created a database with two main tables which are required for our

application.

• The first table is users where it will store all the information of a user. Like full name,

username, password, Email, and phone number.

• The second table is posts where it will store all the posts which are posted by the

different users. The fields are Post ID, Title, content, postdate and owner.

• The database was connected with the front end where all the data as per the inputs

will be saved in the database.

Use-Case Realization

Database

User Interface

Implementation

**Form.php:**

<form action="index.php" method="POST" class="form login">

Username: <input type="text" class="text\_field" name="username" /><br>

Password: <input type="password" class="text\_field" name="password" /><br>

<button class="button" type="submit">

Login

</button>

</form>

<button class="button" onclick="window.location.href='registrationform.php';">

New User Registration

</button>

**Index.php:**

<?php

session\_set\_cookie\_params(15\*60, "/", "waph-team01.mini.facebook.com", TRUE, TRUE);

session\_start();

require "database.php";

if (isset($\_POST["username"]) && isset($\_POST["password"])) {

$username = $\_POST["username"];

$password = $\_POST["password"];

if (checklogin\_mysql($username, $password)) {

$\_SESSION["authenticated"] = TRUE;

$\_SESSION["username"] = $username;

$\_SESSION["browser"] = $\_SERVER["HTTP\_USER\_AGENT"];

} else {

session\_destroy();

echo "<script>alert('Invalid Username or password please recheck');window.location='form.php';</script>";

die();

}

}

if (!isset($\_SESSION["authenticated"]) || $\_SESSION["authenticated"] != TRUE) {

session\_destroy();

echo "<script>alert('You have not logged in. Please login first');</script>";

header("Refresh:0; url=form.php");

die();

}

if ($\_SESSION["browser"] != $\_SERVER["HTTP\_USER\_AGENT"]) {

session\_destroy();

echo "<script>alert('Session hijack detected')</script>";

header("Refresh:0; url=form.php");

die();

}

?>

<h2> Welcome <?php echo htmlentities($\_SESSION['username']); ?> !</h2>

<a href="changepasswordform.php?username=<?php echo urlencode($\_SESSION['username']); ?>">Change password</a> | <a href="edituser.php?username=<?php echo urlencode($\_SESSION['username']); ?>">Edit profile</a> | <a href ="logout.php">logout

</a>

**Changepasswordform.php:**

<h1>Change Password Form</h1>

<form action="changepassword.php" method="POST" class="form login">

Username: <input type="text" class="text\_field" name="username" value="<?php echo isset($\_GET['username']) ? htmlentities($\_GET['username']) : ''; ?>" readonly /> <br>

Current Password: <input type="password" class="text\_field" name="password" /> <br>

New Password: <input type="password" class="text\_field" name="newpassword" /> <br>

<button class="button" type="submit"> Change password </button>

Edituser.php:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Edit Profile - WAPH</title>

<script type="text/javascript">

function displayTime() {

document.getElementById('digit-clock').innerHTML = "Current time: " + new Date();

}

setInterval(displayTime, 500);

function validateForm() {

var email = document.forms["editProfileForm"]["email"].value;

var phone = document.forms["editProfileForm"]["phone"].value;

// Email validation

var emailPattern = /^[\w.-]+@[\w-]+(\.[\w-]+)\*$/;

if (!emailPattern.test(email)) {

alert("Please enter a valid email address");

return false;

}

// Phone number validation

var phonePattern = /^\d{10}$/;

if (!phonePattern.test(phone)) {

alert("Please enter a valid 10-digit phone number");

return false;

}

}

</script>

</head>

<body>

<h1>Edit User Profile - WAPH</h1>

<div id="digit-clock"></div>

<?php

echo "Visited time: " . date("Y-m-d h:i:sa");

?>

<form name="editProfileForm" action="updateuser.php" method="POST" class="form edit-profile" onsubmit="return validateForm();">

Username: <input type="text" class="text\_field" name="username" value="<?php echo isset($\_GET['username']) ? htmlentities($\_GET['username']) : ''; ?>" readonly /> <br>

Full Name: <input type="text" class="text\_field" name="fullName" required><br>

Email: <input type="email" class="text\_field" name="email" required placeholder="username@example.com"><br>

Password: <input type="password" class="text\_field" name="password" required><br>

Phone Number: <input type="tel" class="text\_field" name="phoneNumber" required pattern="[0-9]{10}" title="Please enter a 10-digit phone number"><br>

<button class="button" type="submit">Update Profile</button>

</form>

</body>

</html>

**registrationform.php:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>WAPH-Login page</title>

<script type="text/javascript">

function displayTime() {

document.getElementById('digit-clock').innerHTML = "Current time: " + new Date();

}

setInterval(displayTime, 500);

function validateForm() {

var email = document.forms["registrationForm"]["email"].value;

var phone = document.forms["registrationForm"]["phone"].value;

// Email validation

var emailPattern = /^[\w.-]+@[\w-]+(\.[\w-]+)\*$/;

if (!emailPattern.test(email)) {

alert("Please enter a valid email address");

return false;

}

// Phone number validation

var phonePattern = /^\d{10}$/;

if (!phonePattern.test(phoneNumber)) {

alert("Please enter a valid 10-digit phone number");

return false;

}

}

</script>

</head>

<body>

<h1>New user registration, WAPH</h1>

<div id="digit-clock"></div>

<?php

echo "Visited time: " . date("Y-m-d h:i:sa");

?>

<form name="registrationForm" action="addnewuser.php" method="POST" class="form login" onsubmit="return validateForm();">

Username: <input type="text" class="text\_field" name="username" required

pattern="^[\w.-]+@[\w-]+(.[\w-]+)\*$"

title="Email address is required as username"

placeholder="username in email"

onchange="this.setCustomerValidity(this.validity.patternMismatch?this.title: '');" /> <br>

Full Name: <input type="text" class="text\_field" name="fullName" required><br>

Email: <input type="email" class="text\_field" name="email" required placeholder="username@example.com"><br>

Password: <input type="password" class="text\_field" name="password" required><br>

Phone Number: <input type="tel" class="text\_field" name="phoneNumber" required pattern="[0-9]{10}" title="Please enter a 10-digit phone number"><br>

<button class="button" type="submit">Register</button>

</form>

</body>

</html>

**Security Analysis**

1) How did you apply the security programming principles in your project?

Input validation : Once the username and password are provides the code will check the

details against the database to make sure if the credentials are valid or not. And also we

have implemented the parametrised statements which will help us to reduce the risk of

injection attacks.

Hashing Passwords: All the passwords in the database are stored in hashed way. So, that

we make sure that the passwords secure. For this we have used the md5 algorithm for

hashing the passwords.

2) What database security principles have you used in your project?

The parametrised statements are used in the queries which are used for validation against

the database records. This will help us to reduce the risk of injection attacks and also

“waphteamSpring” user has been provided with the least privileges than root user.

3) Is your code robust and defensive? How?

Session Management: we uses session\_start() and $\_SESSION to manage user sessions,

which is crucial for maintaining user authentication and preventing unauthorized access.

Error Handling: We includes error handling for database connection ($mysqli-

>connect\_errno) and login failures, which helps maintain our system integrity and prevents

information leakage.

Protection Against Session Hijacking: By comparing the user agent

($\_SERVER["HTTP\_USER\_AGENT"]) of the current request with the one stored in the session,

the code attempts to detect and prevent session hijacking attacks. Password Hashing: We

use md5() to hash passwords before comparing them in the SQL query, which is a basic but

essential step in protecting against password-related attacks.

Prepared Statements: To protect against SQL injection attacks, we use prepared

statements ($stmt->bind\_param()) in the database query. This helps to separate the SQL

code from user input, making the system more secure.

4) How did you defend your code against known attacks such as XSS, SQL Injection, CSRF,

Session Hijacking.

XSS (Cross-Site Scripting): WE use a technique called HTML entities to make sure that any

special characters entered by users are treated as plain text and not as code. This helps to

prevent attacks where hackers try to inject harmful code into a website.

SQL Injection: Our code uses a method called prepared statements with parameterized

queries to separate the user's input from the actual SQL code. This helps to prevent attacks

where hackers try to manipulate the database by inserting malicious SQL commands.

Session Hijacking: Our code checks if the user's browser matches the one stored in the

session. If they do not match, it assumes that someone is trying to hijack the session and

terminates it. This is an effective way to defend against session hijacking.

5) How do you separate the roles of super users and regular users?

Role-Based Access Control: To distinguish users with different roles, we added a column

in the database that specifies the role for each user. By doing this, we can utilize this role

information to decide which actions each user is permitted to perform.

Software Process Management

In general, every project in an organization requires a bigger team to work on the project.

There will be n number of tasks and subtasks that every team should work on. Then all the

subtasks will be integrated as a single task. For this to be complete without any

communication gap, there is a tool called GitHub to manage work collaborations. We

followed the pipeline process where we divided the tasks among ourselves and then

developed code is pushed to the main branch. Like this all of us are aware of the new files

created/deleted.

Scrum process

Sprint 1

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

Completed Tasks:

1. Task 1: Created the database as per the requirements mentioned in the rubrics.

2. Task 2: Implemented the login form and the registration form.

3. Task 3: Implemented the basic functions for the logged-in users. That is to change

passwords, edit their profile, including name, additional email, phone, and View posts

from the database.

Contributions:

1. Pushpa Kafley, 2 commits, 5 hours, contributed in changing database, form.php and

modifying readme contents

2. Lakshmi Narayana, 3 commits, 5 hours, contributed in modifying index.php,

changeprofile.php and modifying readme contents

3. Ganesh Atmakuri, 4 commits, 5 hours, contributed in implementing

registrationform.php, addnewuser.php, readme and modifying readme contents

4. Sai Keerthi Vadnala, 2 commits, 5 hours, contributed in modifying recent posts,

updating database and modifying readme contents