WAPH-jaindy Web Application Programming and Hacking

Instructor: Dr. Phu Phung

Student Name: Divyani Jain

Email: jaindy@mail.uc.edu

Short bio: I am a graduate student in Information Technology.

Repository Information

Private Repository's URL: https://github.com/jaindy/waph-jaindy.git

Individual Project 2 url: https://github.com/jaindy/waph-jaindy/tree/main/IndividualProjects/IndividualProject02

Video Link: https://github.com/jaindy/waph-jaindy/blob/main/IndividualProjects/IndividualProject02/jaindy-waph-project2.mp4

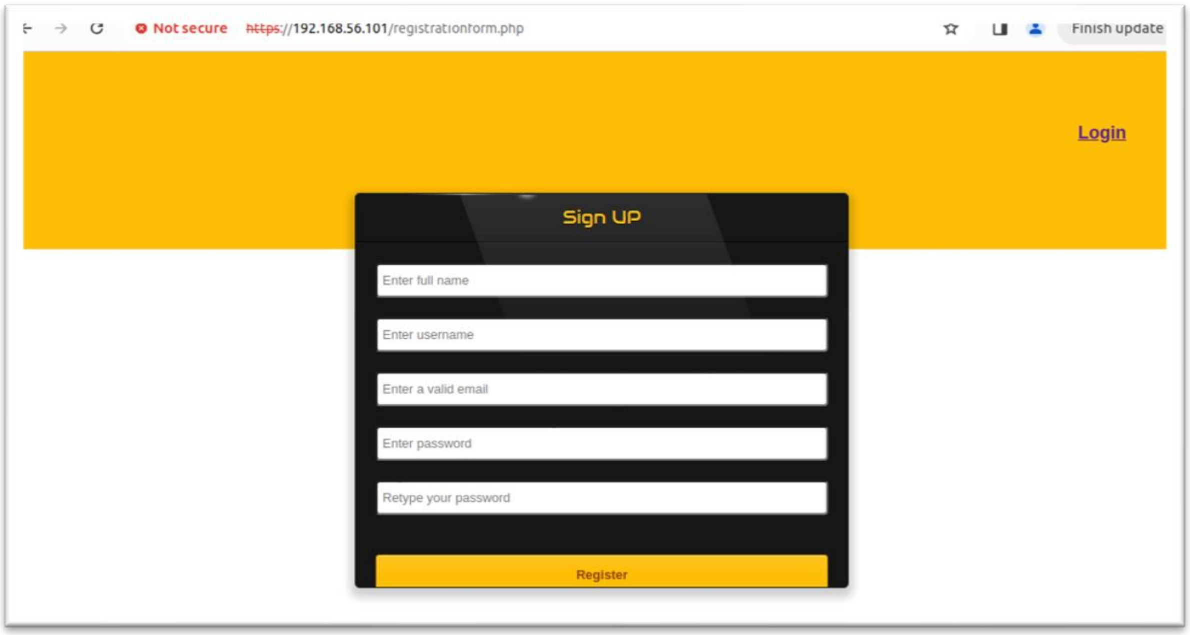
Individual Project 2 - Full-stack Web Application Development

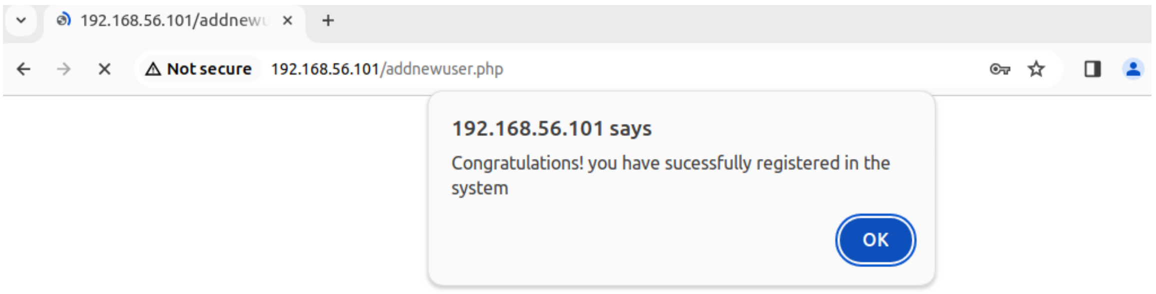
Project Overview:

The project aims to build a secure login system, encompassing user registration, login functionality, profile viewing and editing, and password management. Robust security measures have been implemented to ensure the integrity and confidentiality of user data. Input validation is enforced both at the client and server sides, and session management techniques have been employed to safeguard against session hijacking attacks. Furthermore, the application is fortified against SQL injection attacks through the utilization of user accounts for MySQL database interaction, hashing passwords, and utilizing prepared statements for all SQL operations. Deployment over HTTPS ensures secure communication, while HTML, CSS Bootstrap templates, and JavaScript enhance the user interface. Additionally, tokens have been implemented to mitigate Cross-Site Request Forgery (CSRF) attacks.

Functional Requirements

1. User Registration: Sign up option allows new users to register in users registered system by entering their name, username, email addresses, and password. To ensure data integrity, provide input validation on both the client and the server side. After filling all the required fields, the system allows user to click on register button which gives a success pop up alert that account registered successfully. After clicking on ok user redirect it to login screen.

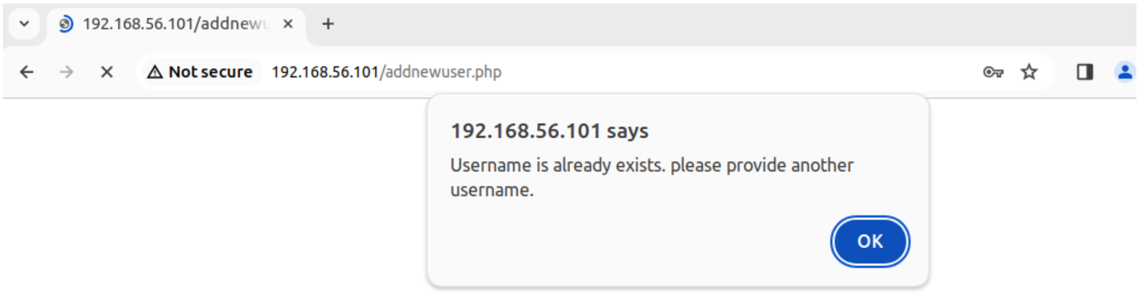




192.168.56.101 says

Congratulations! you have sucessfully registered in the system

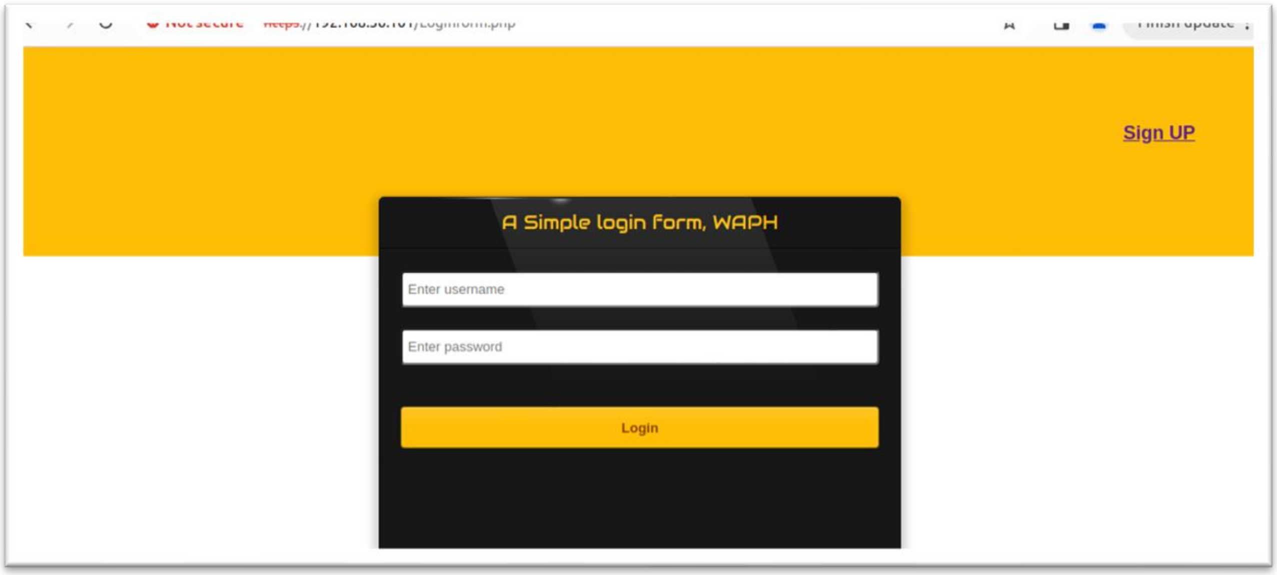
or if username is already registered then give an error message.

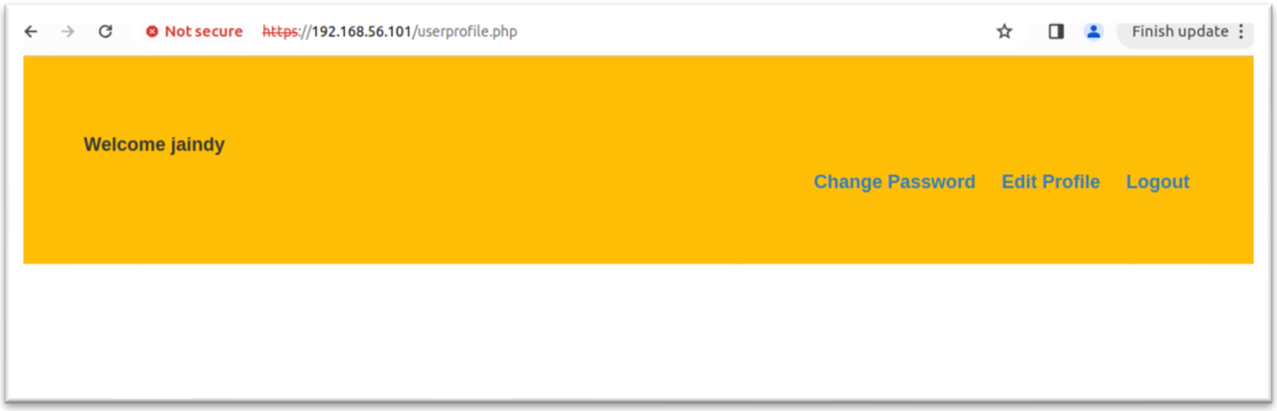


192.168.56.101 says

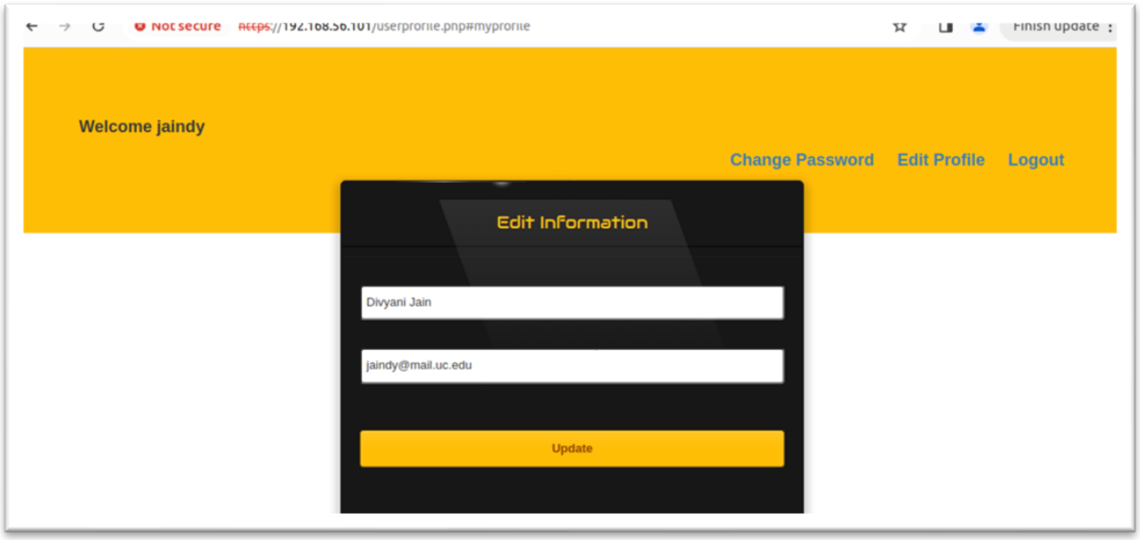
Username is already exists. please provide another username.

2. Login: Login option allows users to enter valid credentials username and password in secure login system that authenticates users and allows them to access their profiles. Used session management to keep user data consistent across the application.





3. Profile Management: Allow users to view and change their profile information, such as name and email address. When user click on edit profile link the name and email is populated from database and allow user to edit and update. Update button allows users to update their name and email address. The logout button takes users back to login screen.

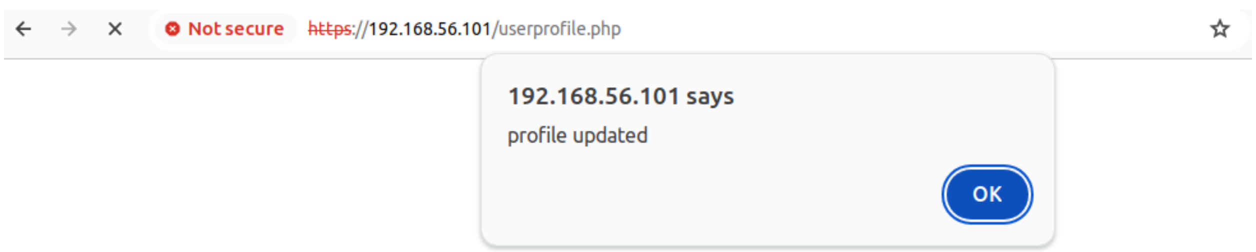

Not secure https://192.168.56.101/userprofile.php#myprofile

Welcome jaindy

Edit Information

Divyani Jain

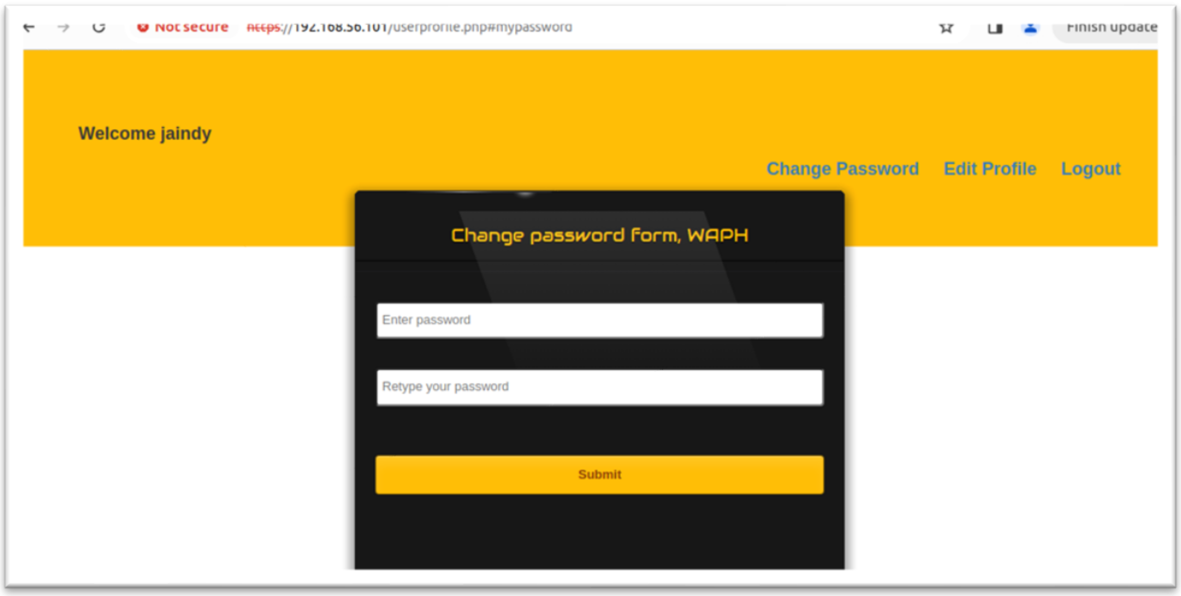
jaindy@mail.uc.edu


Not secure https://192.168.56.101/userprofile.php

192.168.56.101 says

profile updated

4. Password Update: Allow users to change their passwords securely. After changing the password, the system show the success message that password has been changed and redirect it to Login screen.

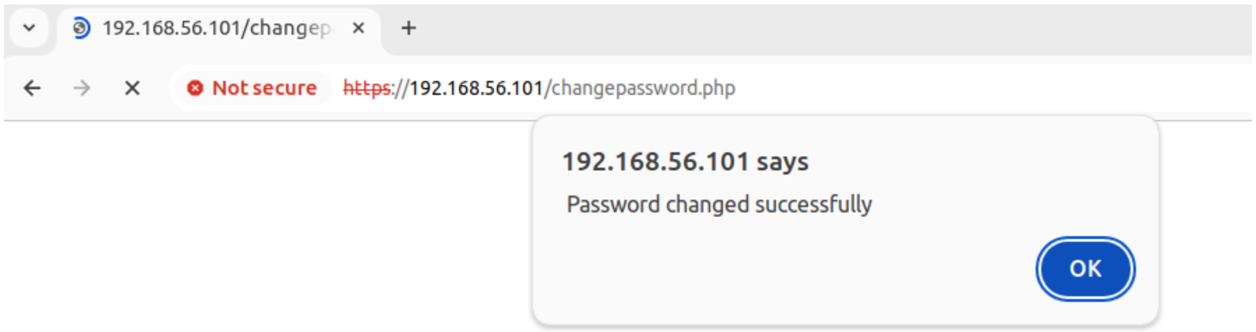

Not secure https://192.168.56.101/userprofile.php#mypassword

Welcome jaindy

Change password form, WAPH

Enter password

Retype your password



192.168.56.101 says
Password changed successfully

Security and Non-technical Requirements

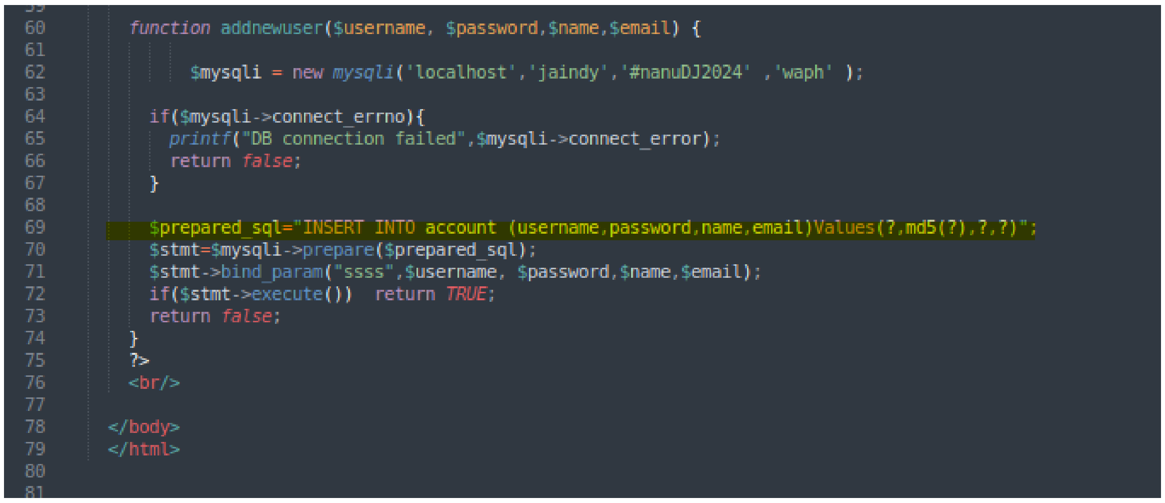
1. Security

a. Web application has deployed over HTTPs. (above screens in technical requirements deployed over https)

b. Used md5 hashed function in SQL statements to securely store passwords.

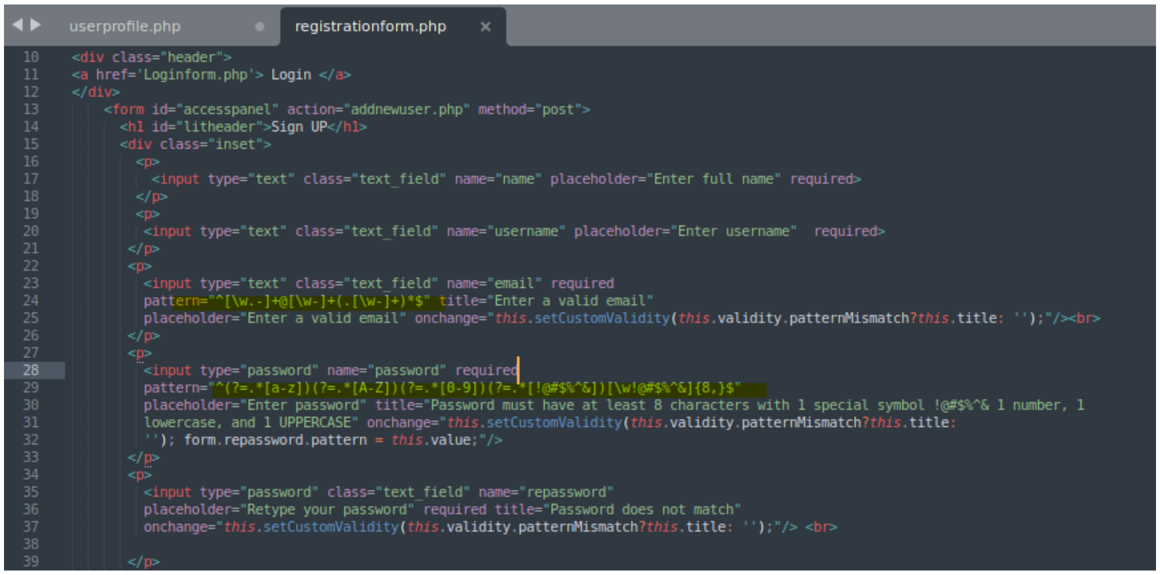
c. Used prepared statements to prevent SQL injection attacks.

d. Created user account in MySQL, didn't use root account in application.

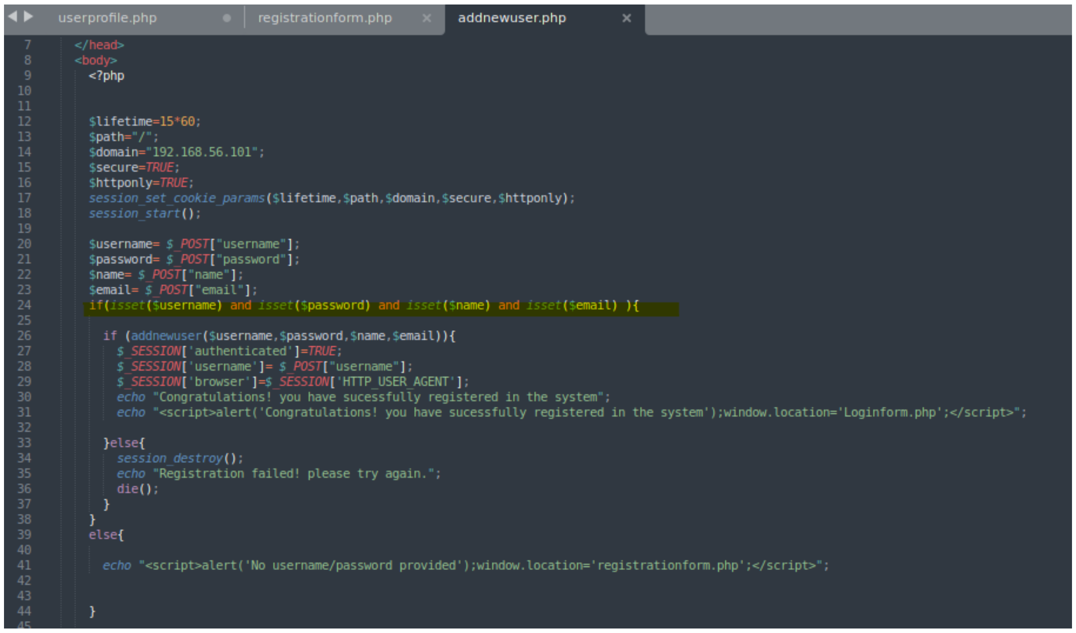


```
    function addnewuser($username, $password,$name,$email) {

            $mysqli = new mysqli('localhost','jaindy','#nanuDJ2024' ,'waph' );

        if($mysqli->connect_errno){
            printf("DB connection failed",$mysqli->connect_error);
            return false;
        }

        $prepared_sql="INSERT INTO account (username,password,name,email)Values(?,md5(?),?,?)";
        $stmt=$mysqli->prepare($prepared_sql);
        $stmt->bind_param("ssss",$username, $password,$name,$email);
        if($stmt->execute())  return TRUE;
        return false;
    }
    ?>
    <br/>

</body>
</html>
```

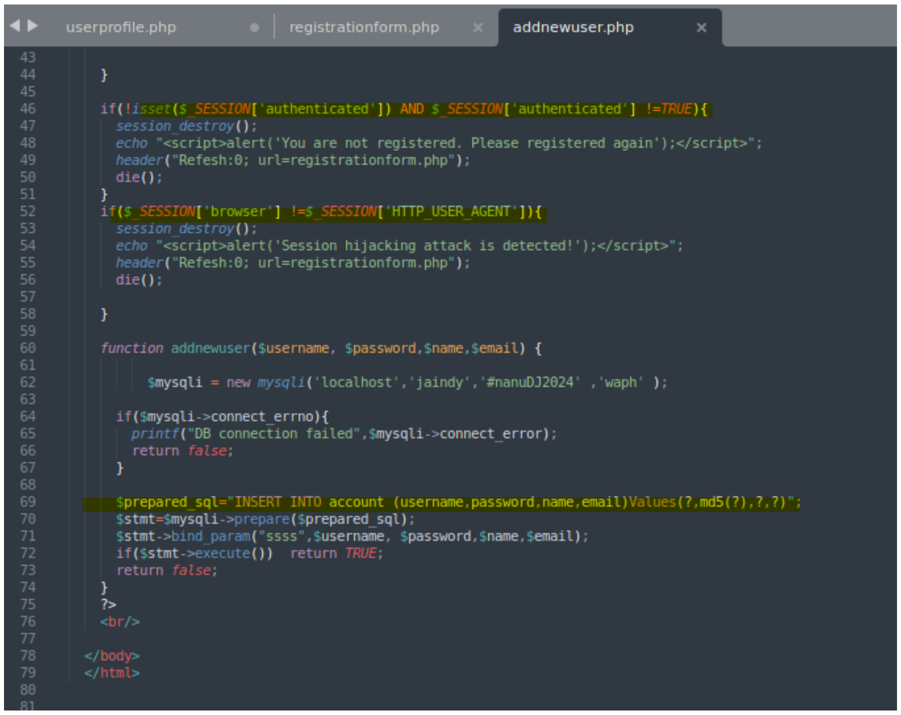
2. Input Validation: Implement comprehensive input validation on both the client and server sides to prevent common web vulnerabilities such as XSS attacks.



```
<div class="header">
<a href='Loginform.php'> Login </a>
</div>
    <form id="accesspanel" action="addnewuser.php" method="post">
      <h1 id="litheader">Sign UP</h1>
      <div class="inset">
        <p>
          <input type="text" class="text_field" name="name" placeholder="Enter full name" required>
        </p>
        <p>
         <input type="text" class="text_field" name="username" placeholder="Enter username"  required>
       </p>
       <p>
         <input type="text" class="text_field" name="email" required
         pattern="^[\w.-]+@[\w-]+(.[\w-]+)*$" title="Enter a valid email"
         placeholder="Enter a valid email" onchange="this.setCustomValidity(this.validity.patternMismatch?this.title: '');"/><br>
       </p>
       <p>
         <input type="password" name="password" required
         pattern="^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*[!@#$%^&])[\w!@#$%^&]{8,}$"
         placeholder="Enter password" title="Password must have at least 8 characters with 1 special symbol !@#$%^& 1 number, 1
         lowercase, and 1 UPPERCASE" onchange="this.setCustomValidity(this.validity.patternMismatch?this.title:
         ''); form.repassword.pattern = this.value;"/>
       </p>
       <p>
         <input type="password" class="text_field" name="repassword"
         placeholder="Retype your password" required title="Password does not match"
         onchange="this.setCustomValidity(this.validity.patternMismatch?this.title: '');"/> <br>

       </p>
```

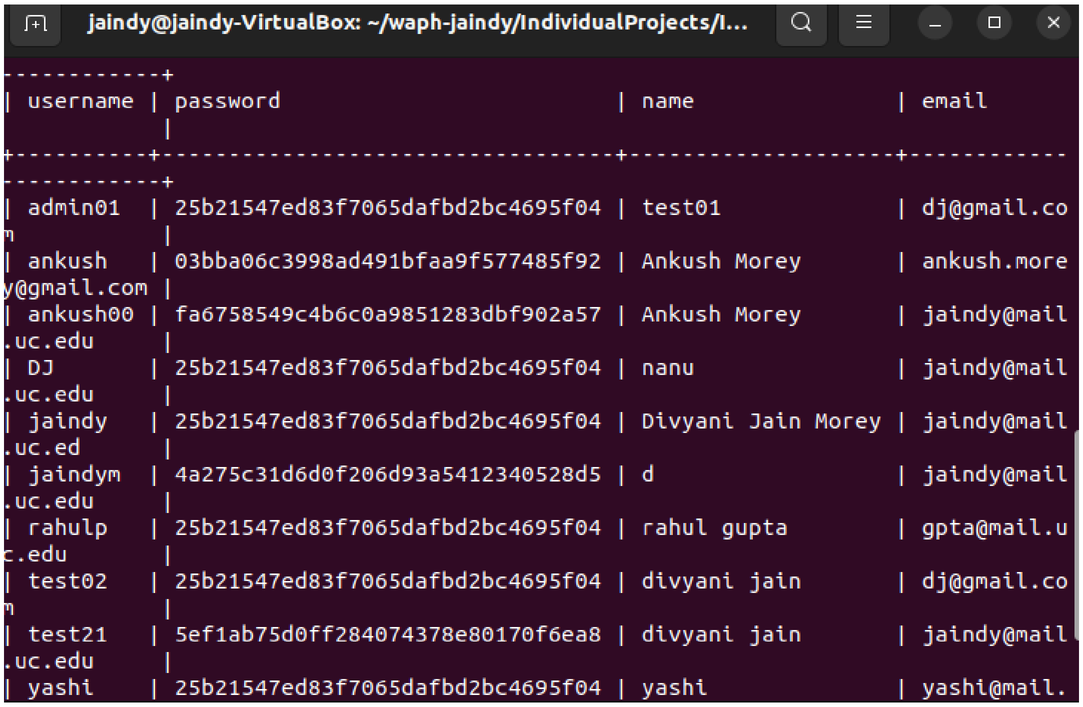


```
  </head>
  <body>
    <?php


    $lifetime=15*60;
    $path="/";
    $domain="192.168.56.101";
    $secure=TRUE;
    $httponly=TRUE;
    session_set_cookie_params($lifetime,$path,$domain,$secure,$httponly);
    session_start();

    $username= $_POST["username"];
    $password= $_POST["password"];
    $name= $_POST["name"];
    $email= $_POST["email"];
    if(isset($username) and isset($password) and isset($name) and isset($email) ){

      if (addnewuser($username,$password,$name,$email)){
        $_SESSION['authenticated']=TRUE;
        $_SESSION['username']= $_POST["username"];
        $_SESSION['browser']=$_SESSION['HTTP_USER_AGENT'];
        echo "Congratulations! you have sucessfully registered in the system";
        echo "<script>alert('Congratulations! you have sucessfully registered in the system');window.location='Loginform.php';</script>";

      }else{
        session_destroy();
        echo "Registration failed! please try again.";
        die();
      }
    }
    else{

      echo "<script>alert('No username/password provided');window.location='registrationform.php';</script>";


    }
```


userprofile.php registrationform.php addnewuser.php

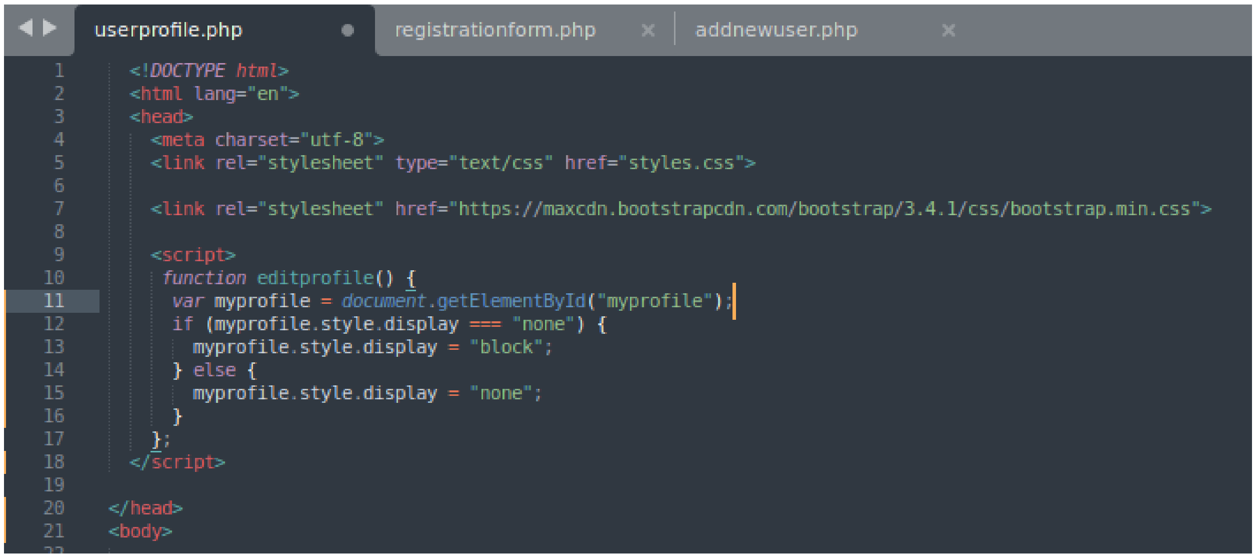
```
    }

    if(!isset($_SESSION['authenticated']) AND $_SESSION['authenticated'] !=TRUE){
      session_destroy();
      echo "<script>alert('You are not registered. Please registered again');</script>";
      header("Refesh:0; url=registrationform.php");
      die();
    }
    if($_SESSION['browser'] !=$_SESSION['HTTP_USER_AGENT']){
      session_destroy();
      echo "<script>alert('Session hijacking attack is detected!');</script>";
      header("Refesh:0; url=registrationform.php");
      die();

    }

    function addnewuser($username, $password,$name,$email) {

          $mysqli = new mysqli('localhost','jaindy','#nanuDJ2024' ,'waph' );

      if($mysqli->connect_errno){
        printf("DB connection failed",$mysqli->connect_error);
        return false;
      }

      $prepared_sql="INSERT INTO account (username,password,name,email)Values(?,md5(?),?,?)";
      $stmt=$mysqli->prepare($prepared_sql);
      $stmt->bind_param("ssss",$username, $password,$name,$email);
      if($stmt->execute())  return TRUE;
      return false;
    }
    ?>
    <br/>

  </body>
  </html>
```

3. Database Design: Design and implement a MySQL database to store user information securely. Used prepared statements in all SQL operations and password is hashed using md5 function.


jaindy@jaindy-VirtualBox: ~/waph-jaindy/IndividualProjects/I...

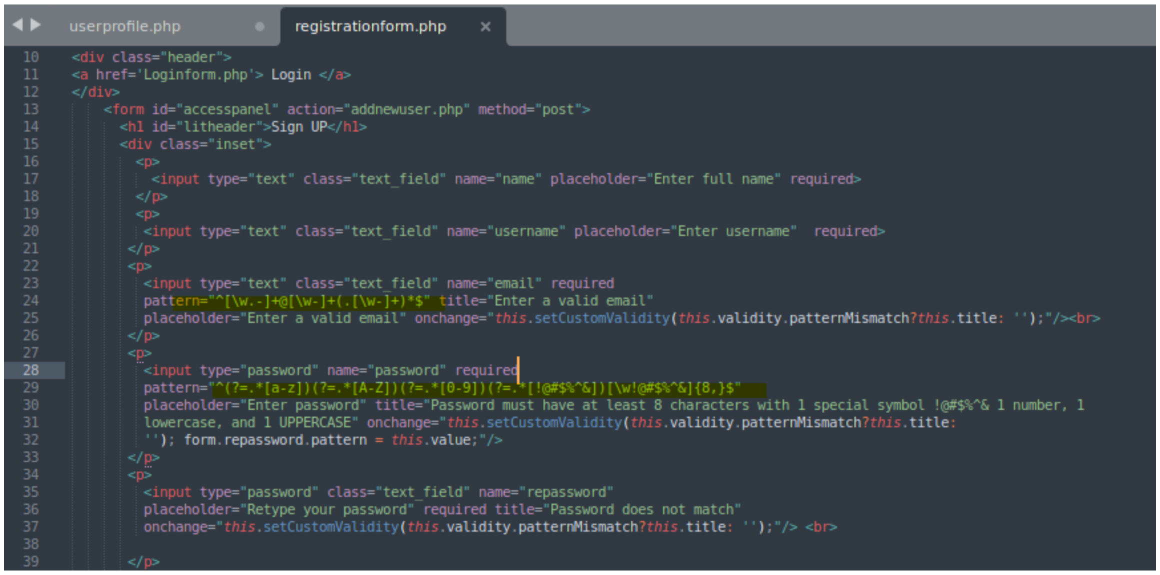
```
-----------+
| username | password                         | name               | email
           |
+----------+----------------------------------+--------------------+-----------
-----------+
| admin01  | 25b21547ed83f7065dafbd2bc4695f04 | test01             | dj@gmail.co
m          |
| ankush   | 03bba06c3998ad491bfaa9f577485f92 | Ankush Morey       | ankush.more
y@gmail.com |
| ankush00 | fa6758549c4b6c0a9851283dbf902a57 | Ankush Morey       | jaindy@mail
.uc.edu    |
| DJ       | 25b21547ed83f7065dafbd2bc4695f04 | nanu               | jaindy@mail
.uc.edu    |
| jaindy   | 25b21547ed83f7065dafbd2bc4695f04 | Divyani Jain Morey | jaindy@mail
.uc.ed     |
| jaindym  | 4a275c31d6d0f206d93a5412340528d5 | d                  | jaindy@mail
.uc.edu    |
| rahulp   | 25b21547ed83f7065dafbd2bc4695f04 | rahul gupta        | gpta@mail.u
c.edu      |
| test02   | 25b21547ed83f7065dafbd2bc4695f04 | divyani jain       | dj@gmail.co
m          |
| test21   | 5ef1ab75d0ff284074378e80170f6ea8 | divyani jain       | jaindy@mail
.uc.edu    |
| yashi    | 25b21547ed83f7065dafbd2bc4695f04 | yashi              | yashi@mail.
```

4. Front-end Development: Use HTML, bootstrap CSS template, and JavaScript to create an intuitive and responsive user interface. Include necessary client-side validations using HTML5 and JavaScript.



userprofile.php registrationform.php addnewuser.php

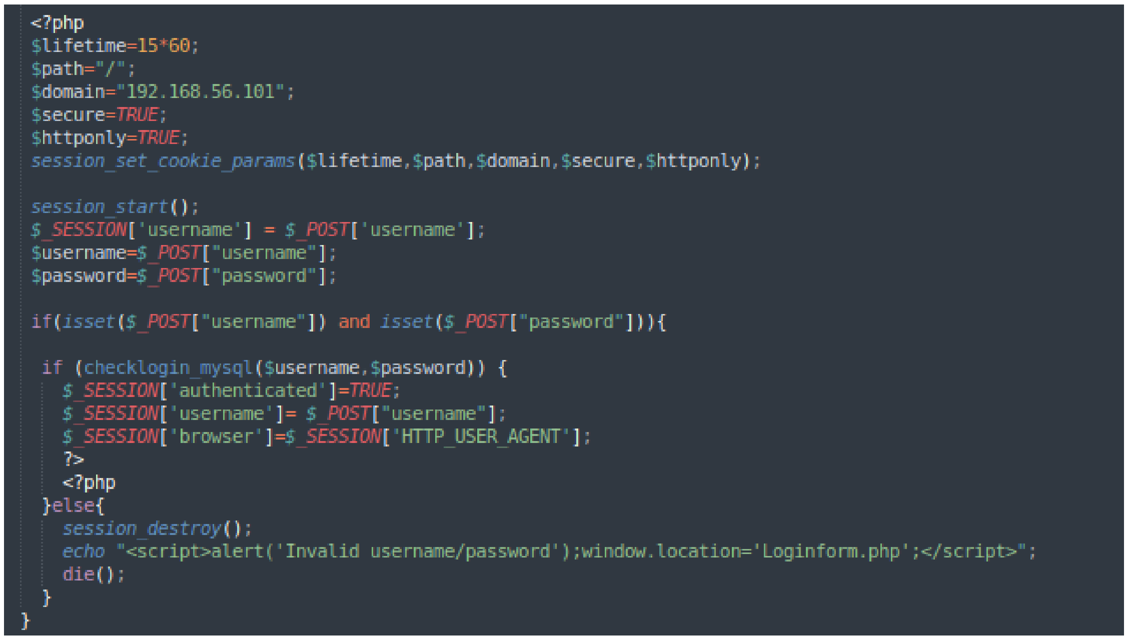
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <link rel="stylesheet" type="text/css" href="styles.css">

  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

  <script>
   function editprofile() {
    var myprofile = document.getElementById("myprofile");
    if (myprofile.style.display === "none") {
      myprofile.style.display = "block";
    } else {
      myprofile.style.display = "none";
    }
  };
 </script>

</head>
<body>
```



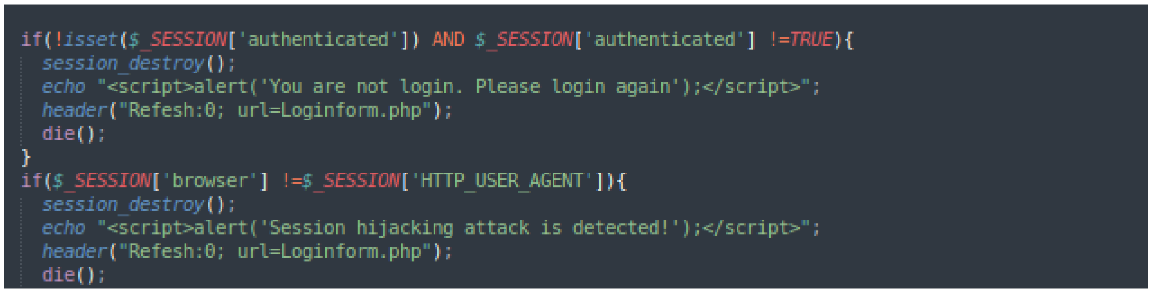
userprofile.php registrationform.php

```
<div class="header">
<a href='Loginform.php'> Login </a>
</div>
    <form id="accesspanel" action="addnewuser.php" method="post">
      <hl id="litheader">Sign UP</hl>
      <div class="inset">
        <p>
          <input type="text" class="text_field" name="name" placeholder="Enter full name" required>
        </p>
        <p>
         <input type="text" class="text_field" name="username" placeholder="Enter username"  required>
       </p>
       <p>
         <input type="text" class="text_field" name="email" required
         pattern="^[\w.-]+@[\w-]+(.[\w-]+)*$" title="Enter a valid email"
         placeholder="Enter a valid email" onchange="this.setCustomValidity(this.validity.patternMismatch?this.title: '');"/><br>
       </p>
       <p>
         <input type="password" name="password" required
         pattern="^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*[!@#$%^&])[\w!@#$%^&]{8,}$"
         placeholder="Enter password" title="Password must have at least 8 characters with 1 special symbol !@#$%^& 1 number, 1
         lowercase, and 1 UPPERCASE" onchange="this.setCustomValidity(this.validity.patternMismatch?this.title:
         ''); form.repassword.pattern = this.value;"/>
       </p>
       <p>
         <input type="password" class="text_field" name="repassword"
         placeholder="Retype your password" required title="Password does not match"
         onchange="this.setCustomValidity(this.validity.patternMismatch?this.title: '');"/> <br>

       </p>
```

5. Session Management: Enable secure session management for user authentication. Prevent session hijacking and fixation attacks.

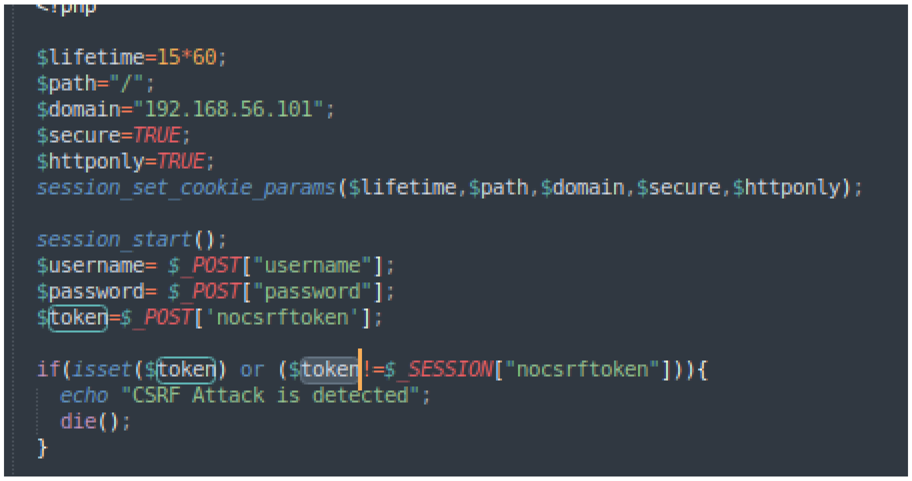


```php
<?php
$lifetime=15*60;
$path="/";
$domain="192.168.56.101";
$secure=TRUE;
$httponly=TRUE;
session_set_cookie_params($lifetime,$path,$domain,$secure,$httponly);

session_start();
$_SESSION['username'] = $_POST['username'];
$username=$_POST["username"];
$password=$_POST["password"];

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

 if (checklogin_mysql($username,$password)) {
   $_SESSION['authenticated']=TRUE;
   $_SESSION['username']= $_POST["username"];
   $_SESSION['browser']=$_SESSION['HTTP_USER_AGENT'];
   ?>
   <?php
 }else{
   session_destroy();
   echo "<script>alert('Invalid username/password');window.location='Loginform.php';</script>";
   die();
 }
}
```

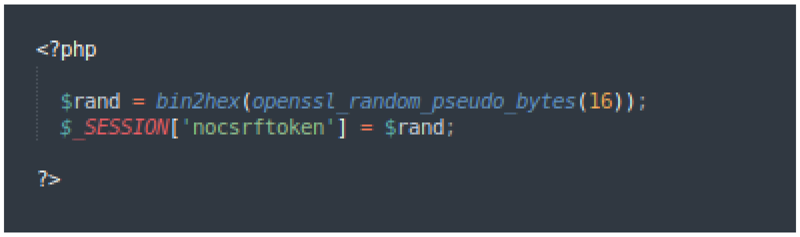


```php
if(!isset($_SESSION['authenticated']) AND $_SESSION['authenticated'] !=TRUE){
  session_destroy();
  echo "<script>alert('You are not login. Please login again');</script>";
  header("Refesh:0; url=Loginform.php");
  die();
}
if($_SESSION['browser'] !=$_SESSION['HTTP_USER_AGENT']){
  session_destroy();
  echo "<script>alert('Session hijacking attack is detected!');</script>";
  header("Refesh:0; url=Loginform.php");
  die();
```

6. CSRF Protection: In database modification use cases, use measures such as anti-CSRF tokens in password changes to defend against Cross-Site Request Forgery (CSRF) attacks.



```php
<?php

$lifetime=15*60;
$path="/";
$domain="192.168.56.101";
$secure=TRUE;
$httponly=TRUE;
session_set_cookie_params($lifetime,$path,$domain,$secure,$httponly);

session_start();
$username= $_POST["username"];
$password= $_POST["password"];
$token=$_POST['nocsrftoken'];

if(isset($token) or ($token!=$_SESSION["nocsrftoken"])){
  echo "CSRF Attack is detected";
  die();
}
```



```php
<?php

  $rand = bin2hex(openssl_random_pseudo_bytes(16));
  $_SESSION['nocsrftoken'] = $rand;

?>
```