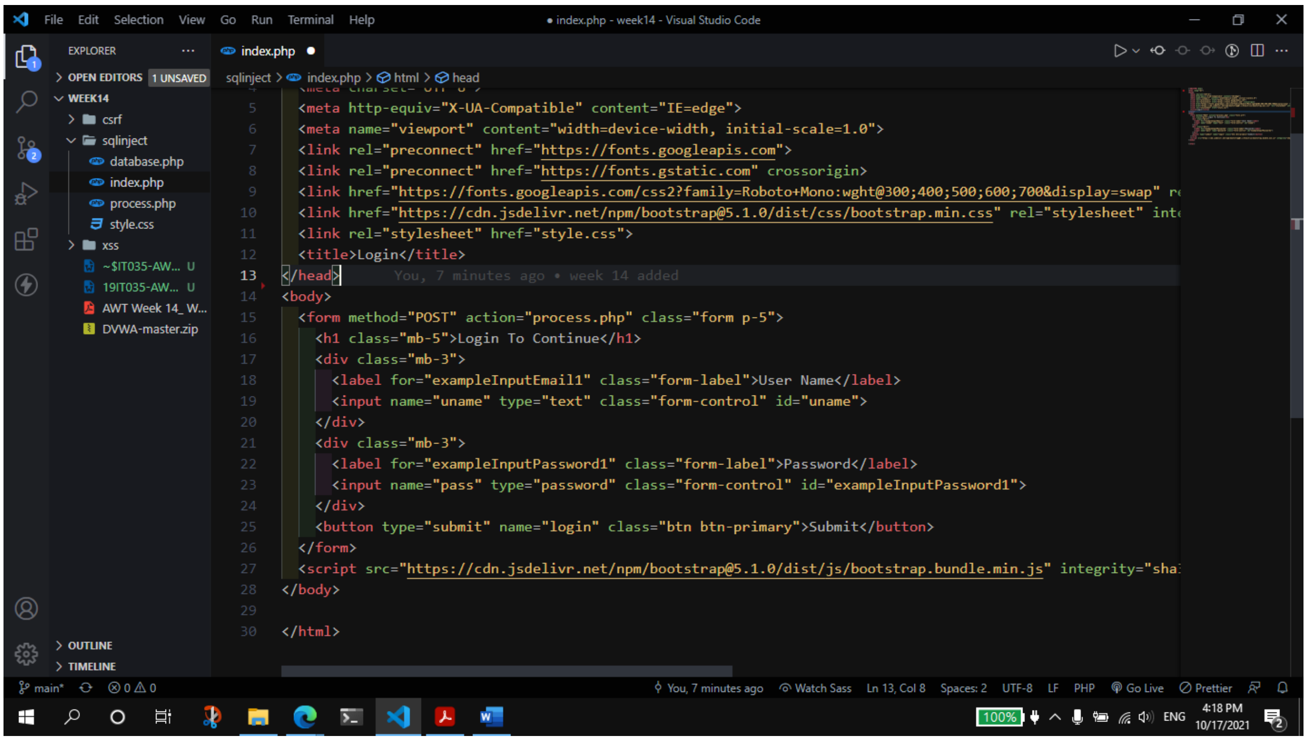
AWT Practical *Week - 14 Task*

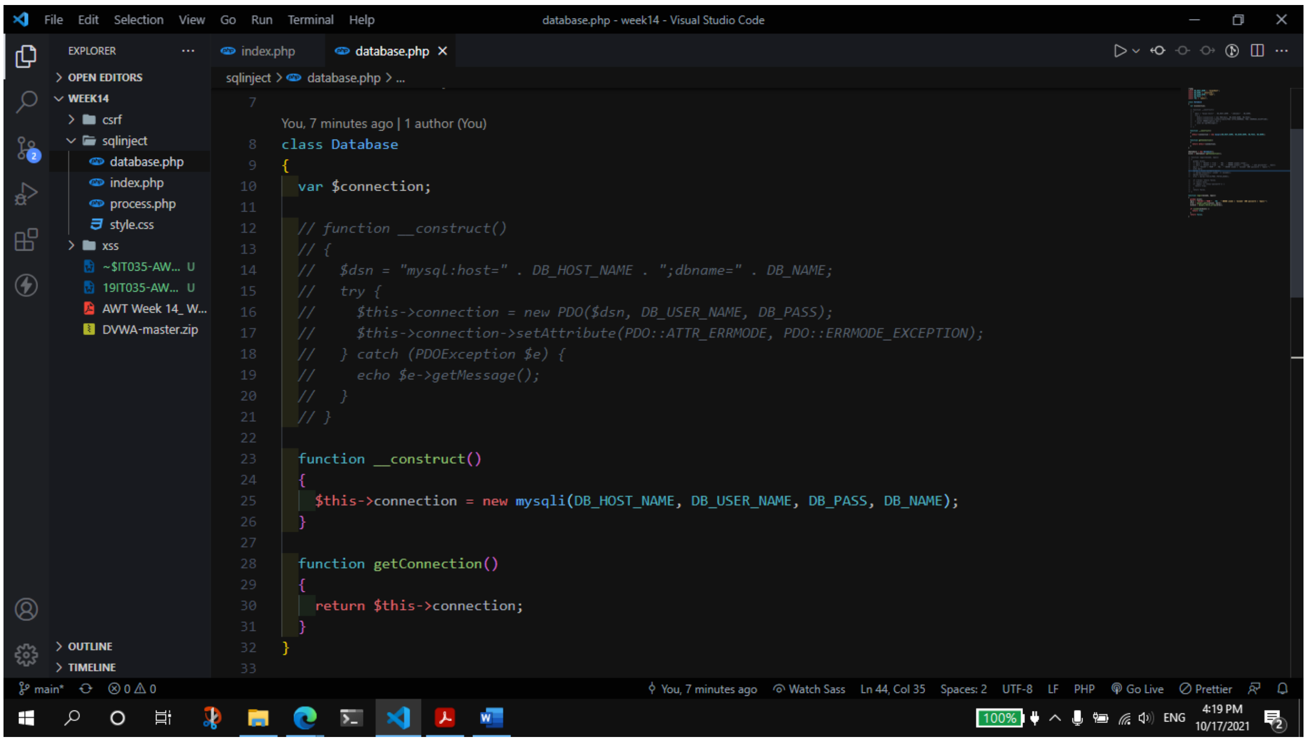
Web Securities (PHP)

1) Demonstrate SQL injection and Prevention in PHP.

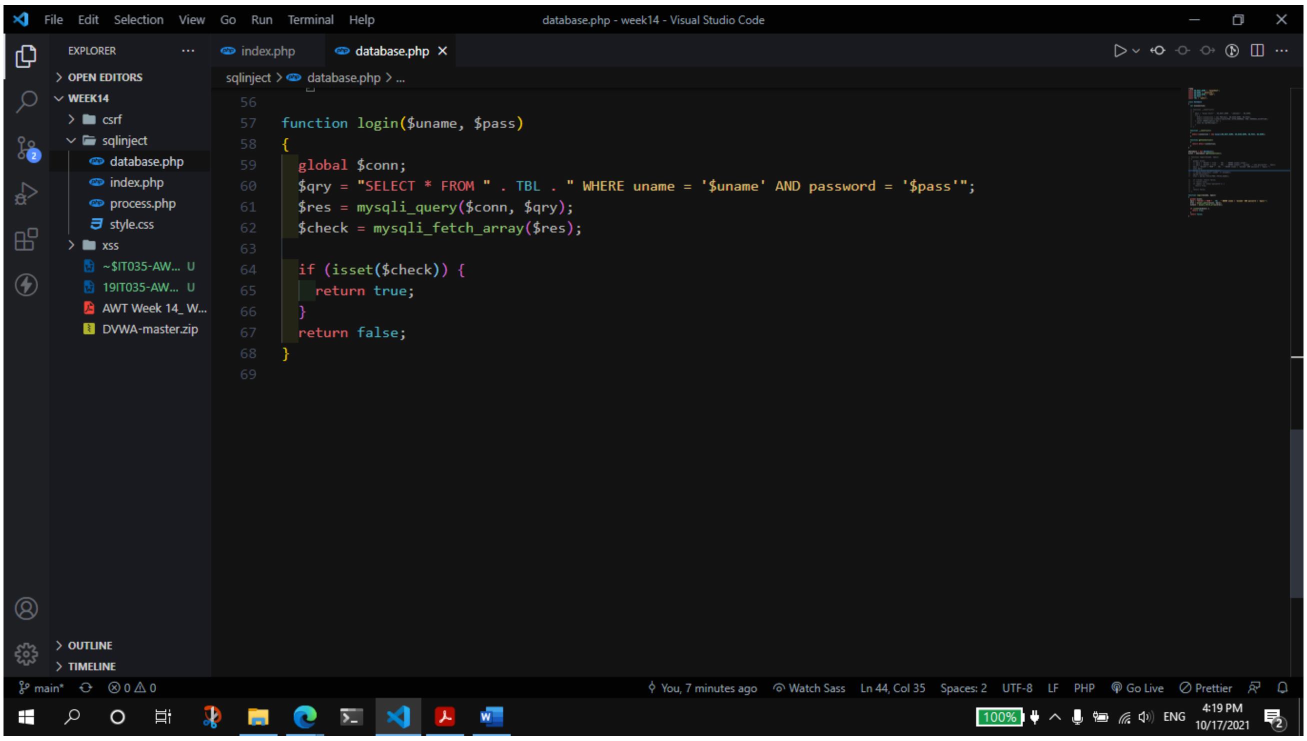
Code: Demo SQL Injection:



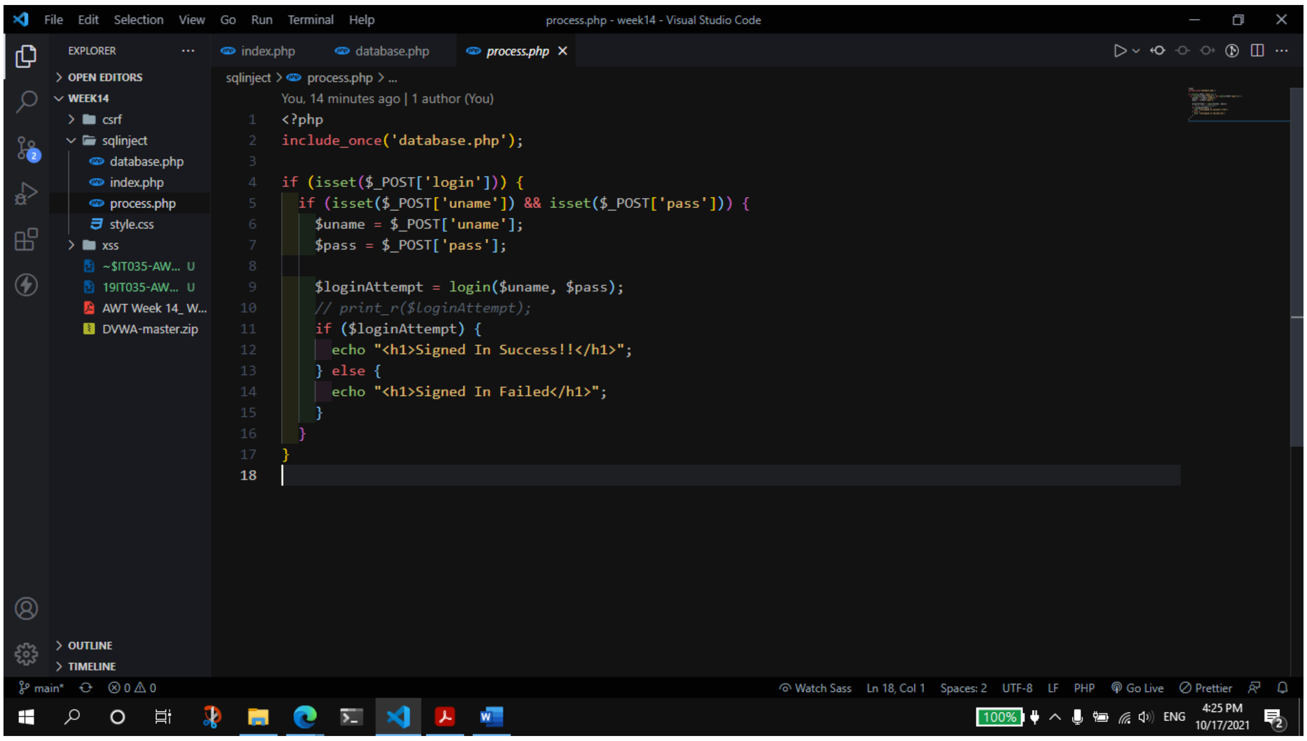
```
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=Roboto+Mono:wght@300;400;500;600;700&display=swap" re
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/css/bootstrap.min.css" rel="stylesheet" inte
    <link rel="stylesheet" href="style.css">
    <title>Login</title>
</head>    You, 7 minutes ago • week 14 added
<body>
    <form method="POST" action="process.php" class="form p-5">
        <h1 class="mb-5">Login To Continue</h1>
        <div class="mb-3">
            <label for="exampleInputEmail1" class="form-label">User Name</label>
            <input name="uname" type="text" class="form-control" id="uname">
        </div>
        <div class="mb-3">
            <label for="exampleInputPassword1" class="form-label">Password</label>
            <input name="pass" type="password" class="form-control" id="exampleInputPassword1">
        </div>
        <button type="submit" name="login" class="btn btn-primary">Submit</button>
    </form>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/js/bootstrap.bundle.min.js" integrity="sha
</body>

</html>
```



```
You, 7 minutes ago | 1 author (You)
class Database
{
    var $connection;

    // function __construct()
    // {
    //   $dsn = "mysql:host=" . DB_HOST_NAME . ";dbname=" . DB_NAME;
    //   try {
    //     $this->connection = new PDO($dsn, DB_USER_NAME, DB_PASS);
    //     $this->connection->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    //   } catch (PDOException $e) {
    //     echo $e->getMessage();
    //   }
    // }

    function __construct()
    {
        $this->connection = new mysqli(DB_HOST_NAME, DB_USER_NAME, DB_PASS, DB_NAME);
    }

    function getConnection()
    {
        return $this->connection;
    }
}
```

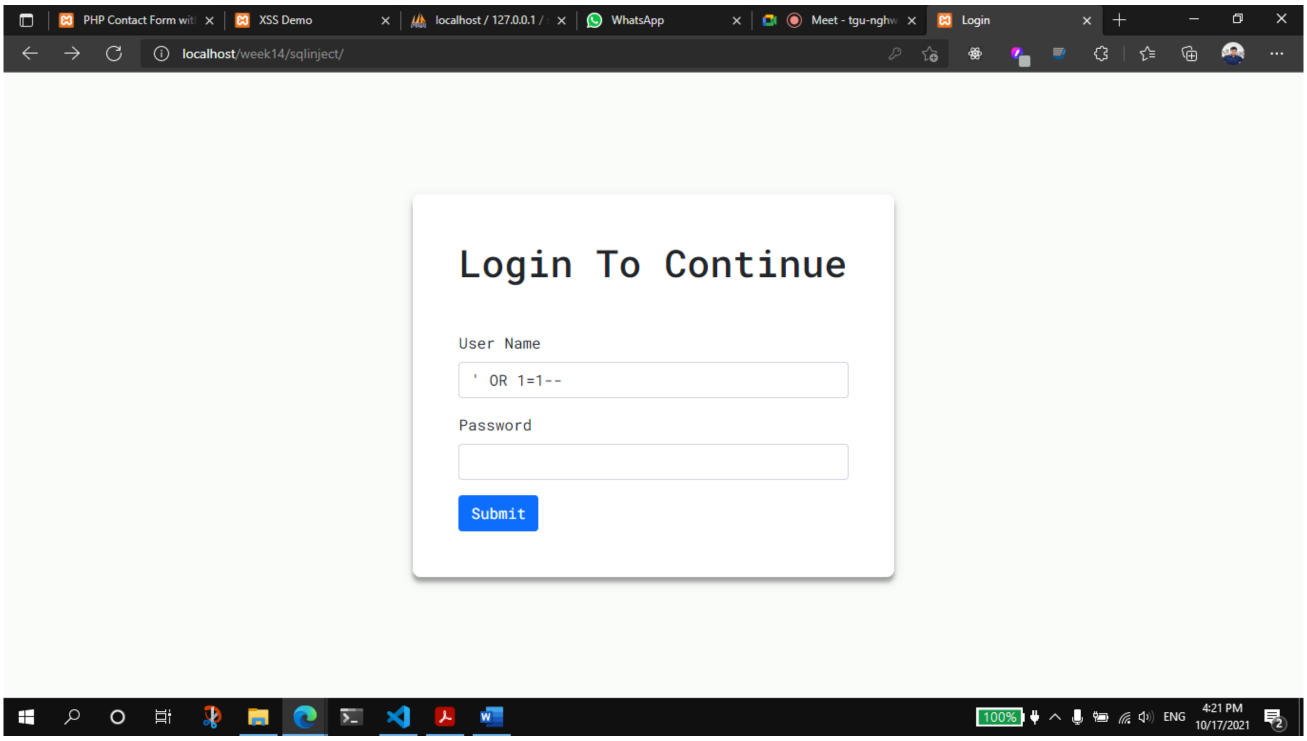

database.php - week14 - Visual Studio Code
sqlinject > database.php > ...

```php

function login($uname, $pass)
{
  global $conn;
  $qry = "SELECT * FROM " . TBL . " WHERE uname = '$uname' AND password = '$pass'";
  $res = mysqli_query($conn, $qry);
  $check = mysqli_fetch_array($res);

  if (isset($check)) {
    return true;
  }
  return false;
}

```


process.php - week14 - Visual Studio Code
sqlinject > process.php > ...
You, 14 minutes ago | 1 author (You)

```php
<?php
include_once('database.php');

if (isset($_POST['login'])) {
  if (isset($_POST['uname']) && isset($_POST['pass'])) {
    $uname = $_POST['uname'];
    $pass = $_POST['pass'];

    $loginAttempt = login($uname, $pass);
    // print_r($loginAttempt);
    if ($loginAttempt) {
      echo "<h1>Signed In Success!!</h1>";
    } else {
      echo "<h1>Signed In Failed</h1>";
    }
  }
}

```

Output:



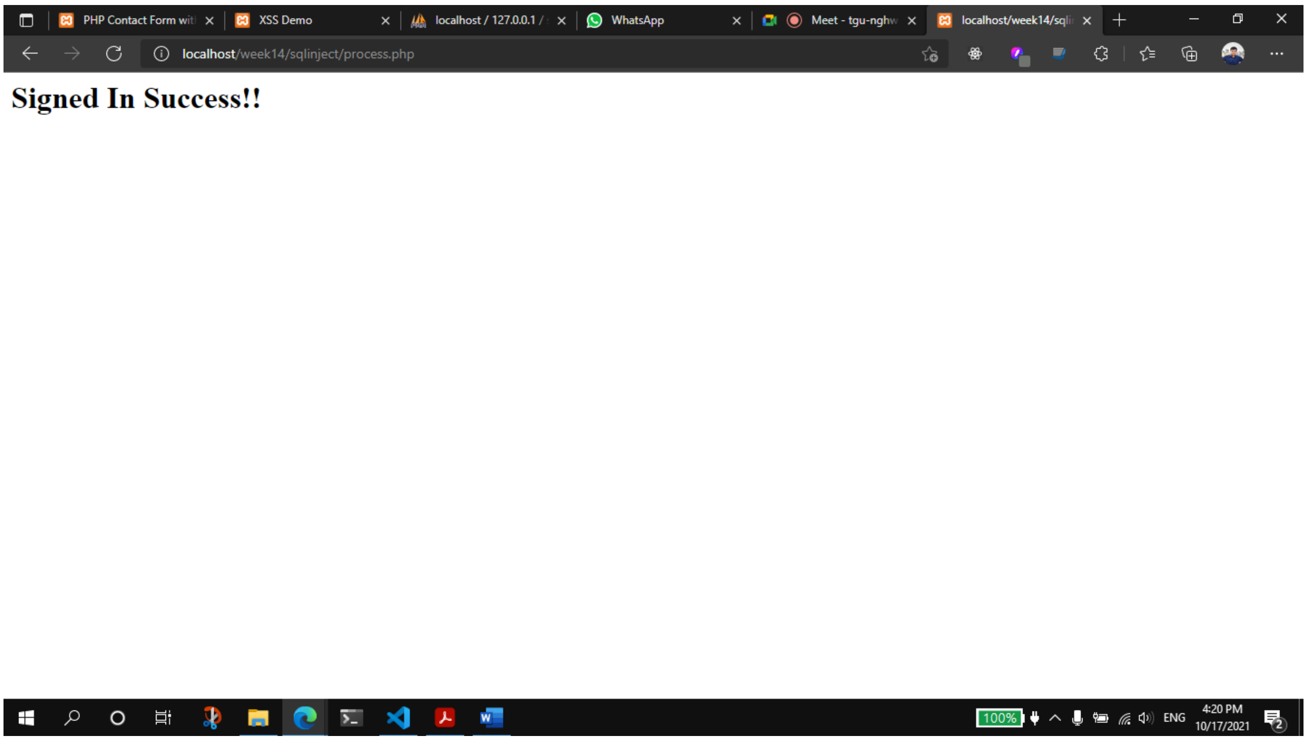
Login To Continue

User Name

' OR 1=1--

Password

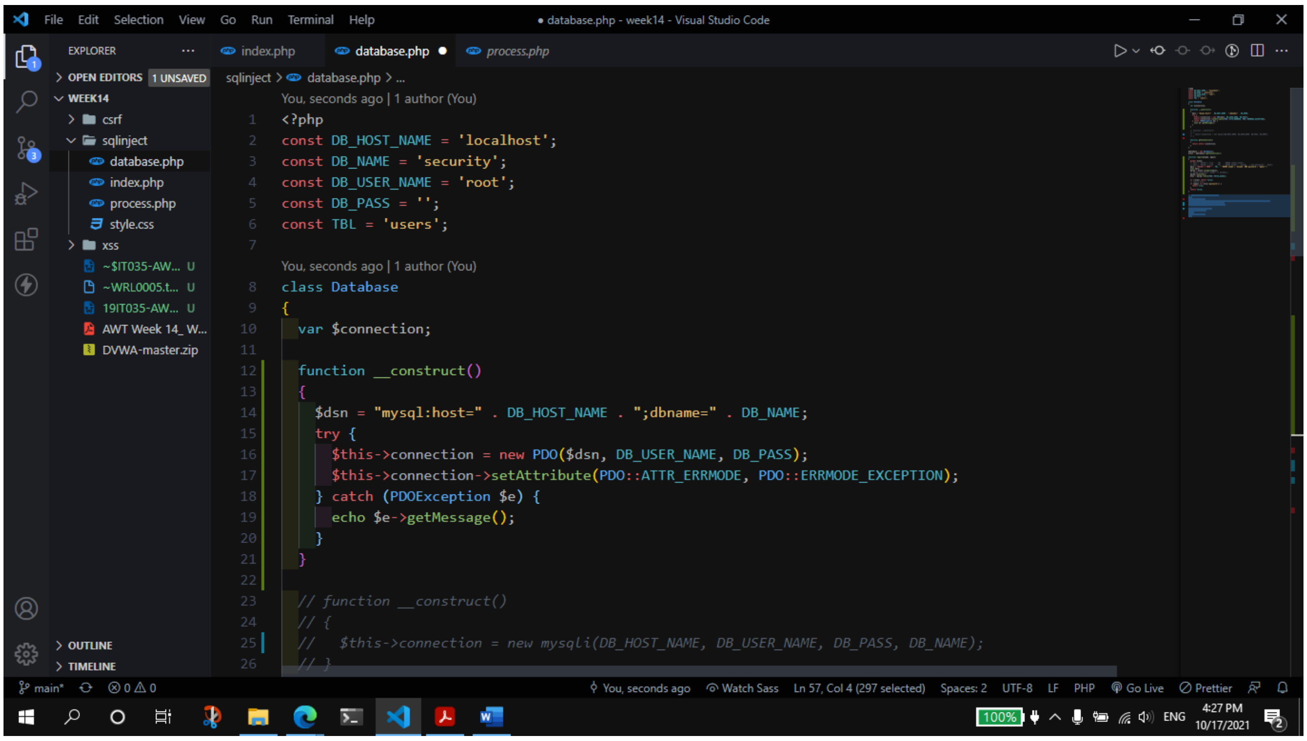
Submit



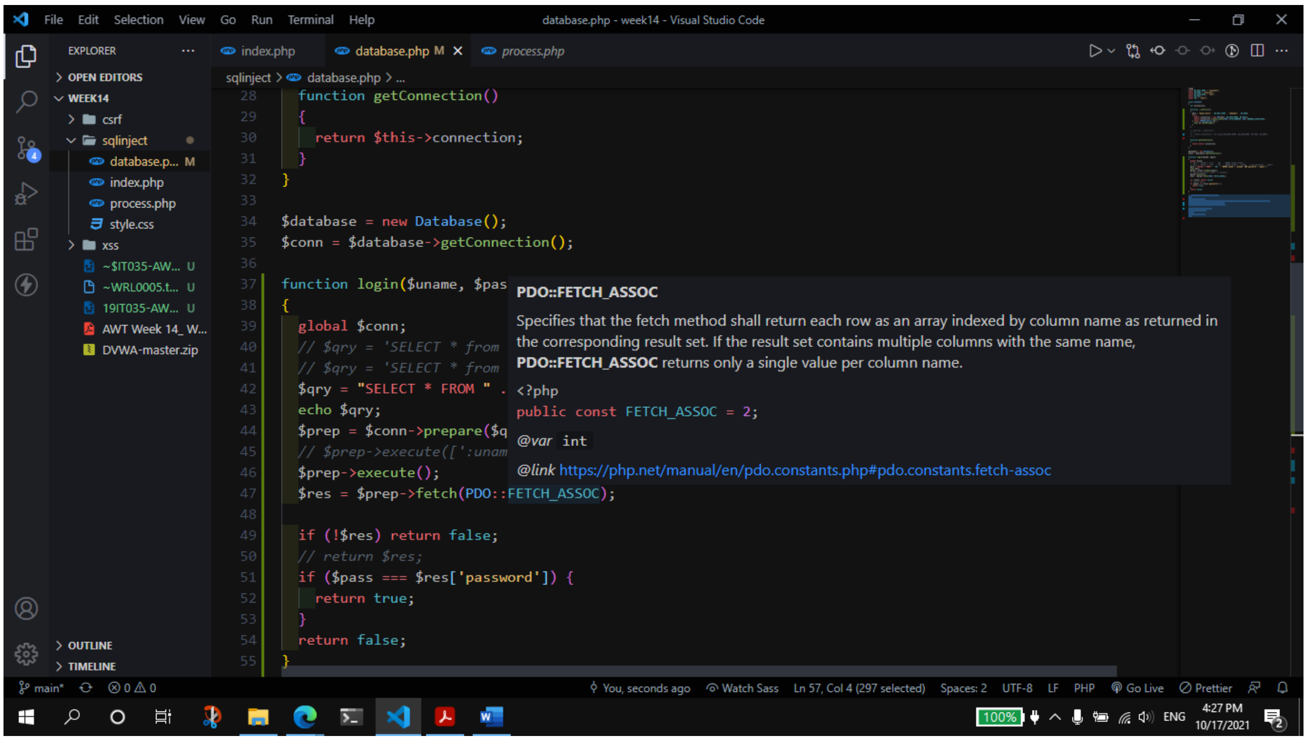
Signed In Success!!

Prevent SQL Injection:

- Should not use the GET method, for submitting form, for sensitive data.
- Should avoid using the older functional approaches for, DB connection, instead use the PDO approach which by default add, single quotes to the input.
- Should use form validation, restrict user to use certain special character.



```php
<?php
const DB_HOST_NAME = 'localhost';
const DB_NAME = 'security';
const DB_USER_NAME = 'root';
const DB_PASS = '';
const TBL = 'users';

class Database
{
  var $connection;

  function __construct()
  {
    $dsn = "mysql:host=" . DB_HOST_NAME . ";dbname=" . DB_NAME;
    try {
      $this->connection = new PDO($dsn, DB_USER_NAME, DB_PASS);
      $this->connection->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    } catch (PDOException $e) {
      echo $e->getMessage();
    }
  }

  // function __construct()
  // {
  //   $this->connection = new mysqli(DB_HOST_NAME, DB_USER_NAME, DB_PASS, DB_NAME);
  // }
```



```php
  function getConnection()
  {
    return $this->connection;
  }
}

$database = new Database();
$conn = $database->getConnection();

function login($uname, $pas
{
  global $conn;
  // $qry = 'SELECT * from
  // $qry = 'SELECT * from
  $qry = "SELECT * FROM " .
  echo $qry;
  $prep = $conn->prepare($q
  // $prep->execute([':unam
  $prep->execute();
  $res = $prep->fetch(PDO::FETCH_ASSOC);

  if (!$res) return false;
  // return $res;
  if ($pass === $res['password']) {
    return true;
  }
  return false;
}
```

PDO::FETCH_ASSOC

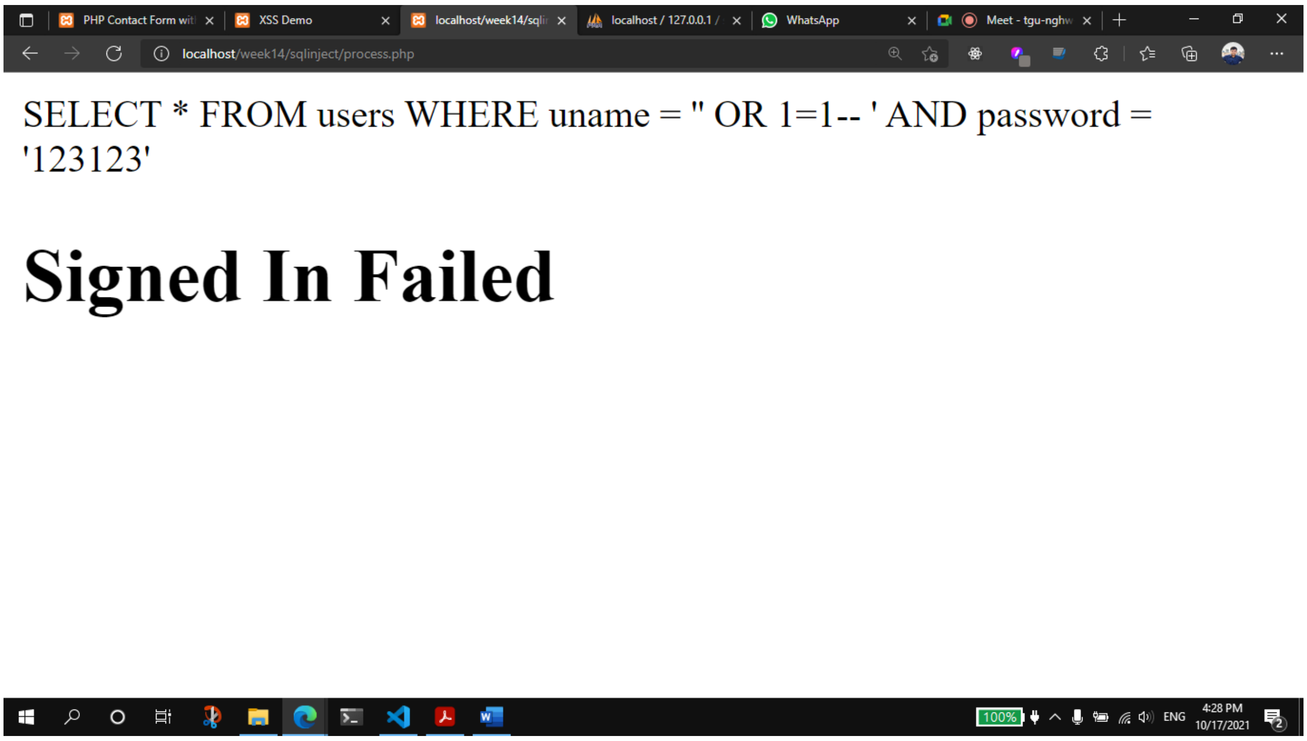
Specifies that the fetch method shall return each row as an array indexed by column name as returned in the corresponding result set. If the result set contains multiple columns with the same name, **PDO::FETCH_ASSOC** returns only a single value per column name.

```php
<?php
public const FETCH_ASSOC = 2;
```

@var int

@link https://php.net/manual/en/pdo.constants.php#pdo.constants.fetch-assoc

Output:



SELECT * FROM users WHERE uname = '' OR 1=1-- ' AND password = '123123'

Signed In Failed

2) What is CSRF Attack? Explain with examples with solutions.

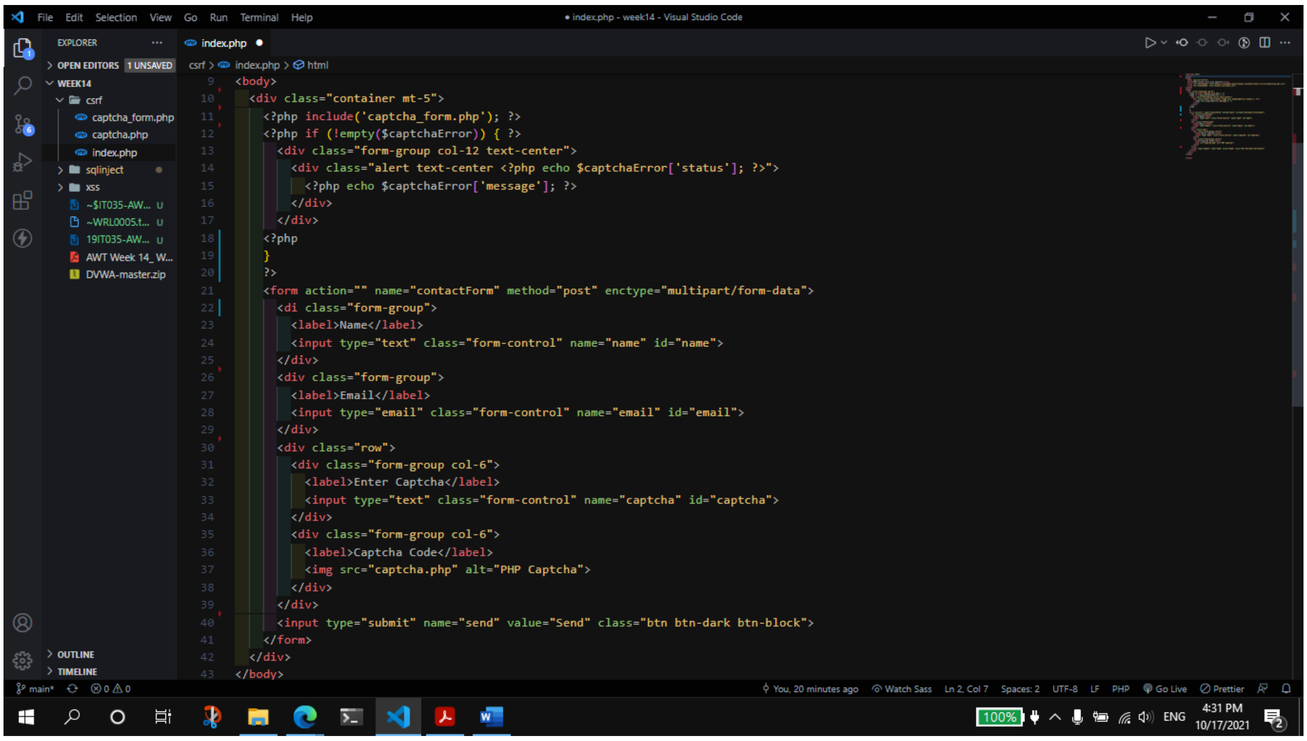
a. Implement Image Captcha Using PHP to Prevent CSRF Attack.

- Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated.
- If the victim is a normal user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth. If the victim is an administrative account, CSRF can compromise the entire web application.
- A CSRF attack is limited to the permissions of the targeted end user. An end user with limited permissions can be forced into changing email addresses, or transferring funds, while an admin account can be forced to compromise an entire web application.\
- For example, a user might receive an email or a text message with a link, which deploys malware or injects malicious code into a web page. Once the user clicks the link, attackers use the malware or injected code to send requests to the web application on the user's behalf.

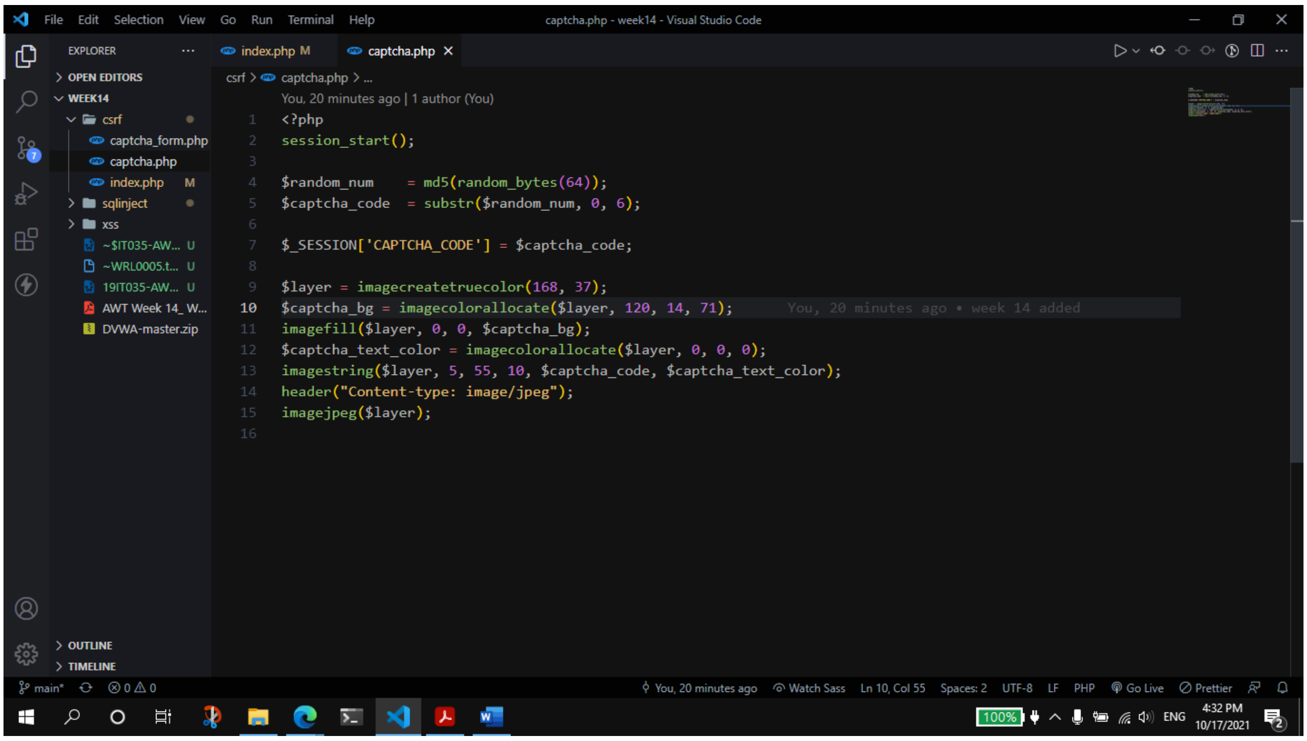
Solutions:

- Captcha
- Anti-CSRF Token
- Use Same site flag in cookies
- Multifactor authentication

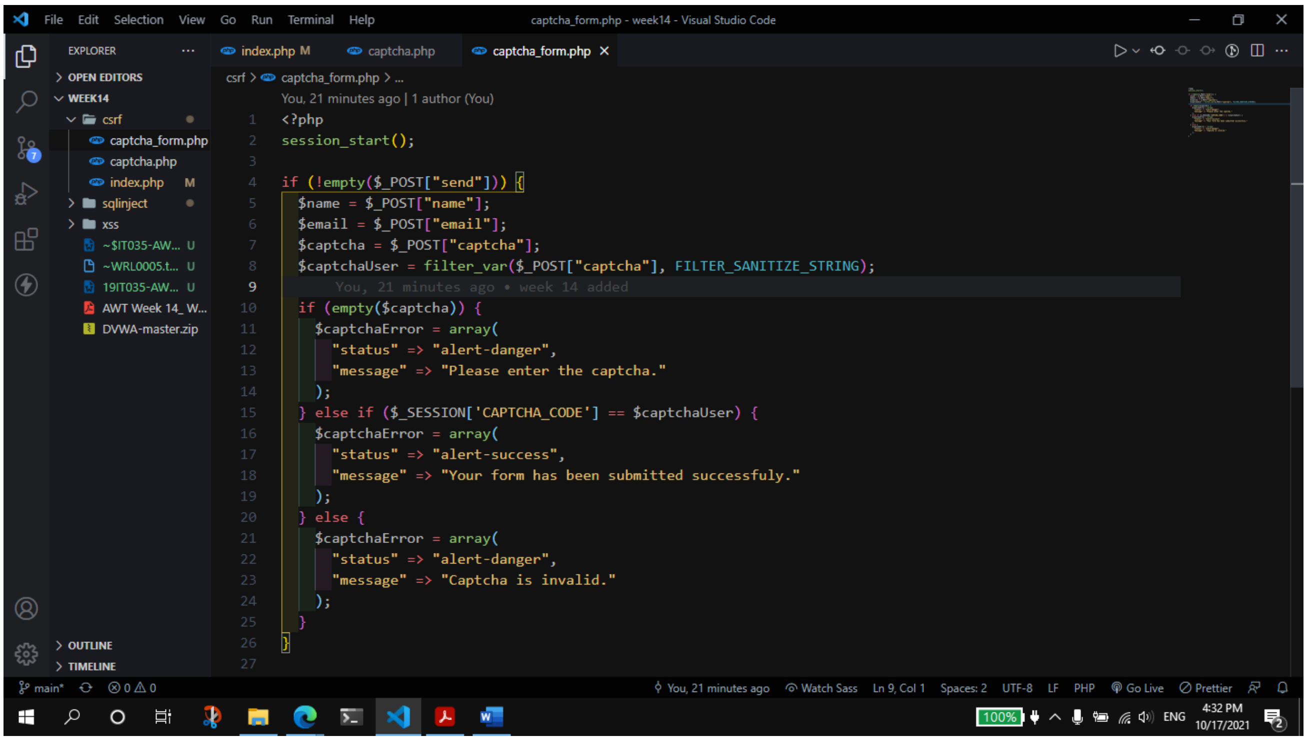
Code:



```
<body>
  <div class="container mt-5">
    <?php include('captcha_form.php'); ?>
    <?php if (!empty($captchaError)) { ?>
      <div class="form-group col-12 text-center">
        <div class="alert text-center <?php echo $captchaError['status']; ?>">
          <?php echo $captchaError['message']; ?>
        </div>
      </div>
    <?php
    }
    ?>
    <form action="" name="contactForm" method="post" enctype="multipart/form-data">
      <di class="form-group">
        <label>Name</label>
        <input type="text" class="form-control" name="name" id="name">
      </div>
      <div class="form-group">
        <label>Email</label>
        <input type="email" class="form-control" name="email" id="email">
      </div>
      <div class="row">
        <div class="form-group col-6">
          <label>Enter Captcha</label>
          <input type="text" class="form-control" name="captcha" id="captcha">
        </div>
        <div class="form-group col-6">
          <label>Captcha Code</label>
          <img src="captcha.php" alt="PHP Captcha">
        </div>
      </div>
      <input type="submit" name="send" value="Send" class="btn btn-dark btn-block">
    </form>
  </div>
</body>
```

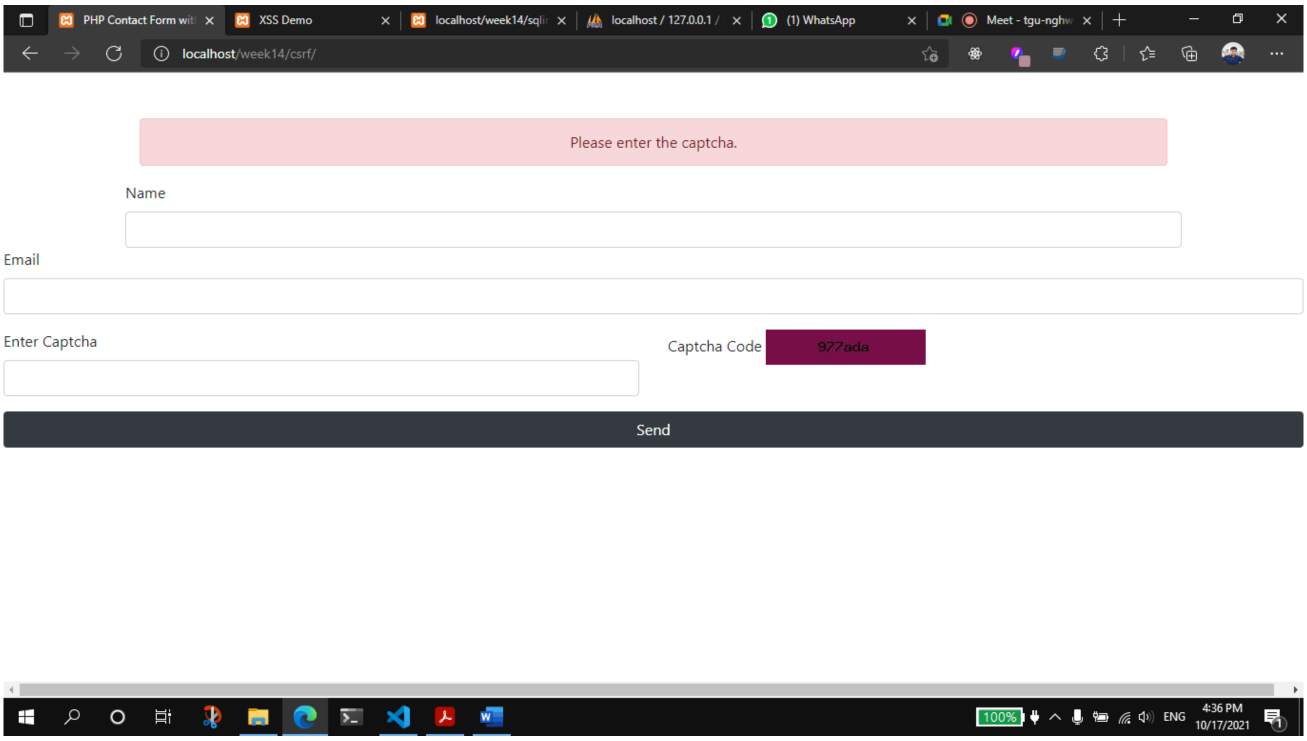


```
<?php
session_start();

$random_num    = md5(random_bytes(64));
$captcha_code  = substr($random_num, 0, 6);

$_SESSION['CAPTCHA_CODE'] = $captcha_code;

$layer = imagecreatetruecolor(168, 37);
$captcha_bg = imagecolorallocate($layer, 120, 14, 71);
imagefill($layer, 0, 0, $captcha_bg);
$captcha_text_color = imagecolorallocate($layer, 0, 0, 0);
imagestring($layer, 5, 55, 10, $captcha_code, $captcha_text_color);
header("Content-type: image/jpeg");
imagejpeg($layer);
```



```php
<?php
session_start();

if (!empty($_POST["send"])) {
  $name = $_POST["name"];
  $email = $_POST["email"];
  $captcha = $_POST["captcha"];
  $captchaUser = filter_var($_POST["captcha"], FILTER_SANITIZE_STRING);

  if (empty($captcha)) {
    $captchaError = array(
      "status" => "alert-danger",
      "message" => "Please enter the captcha."
    );
  } else if ($_SESSION['CAPTCHA_CODE'] == $captchaUser) {
    $captchaError = array(
      "status" => "alert-success",
      "message" => "Your form has been submitted successfuly."
    );
  } else {
    $captchaError = array(
      "status" => "alert-danger",
      "message" => "Captcha is invalid."
    );
  }
}
```

Output:



localhost/week14/csrf/

Please enter the captcha.

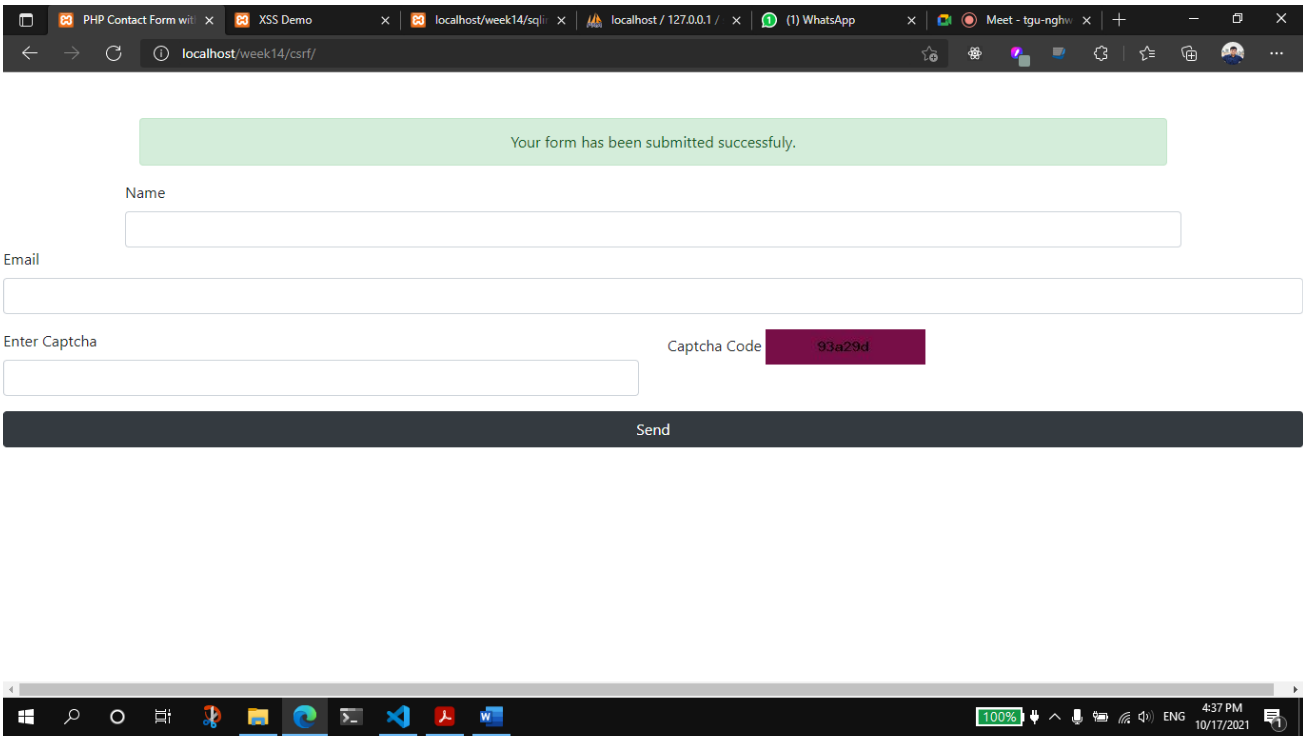
Name

Email

Enter Captcha

Captcha Code 977ada

Send



Your form has been submitted successfuly.

Name

Email

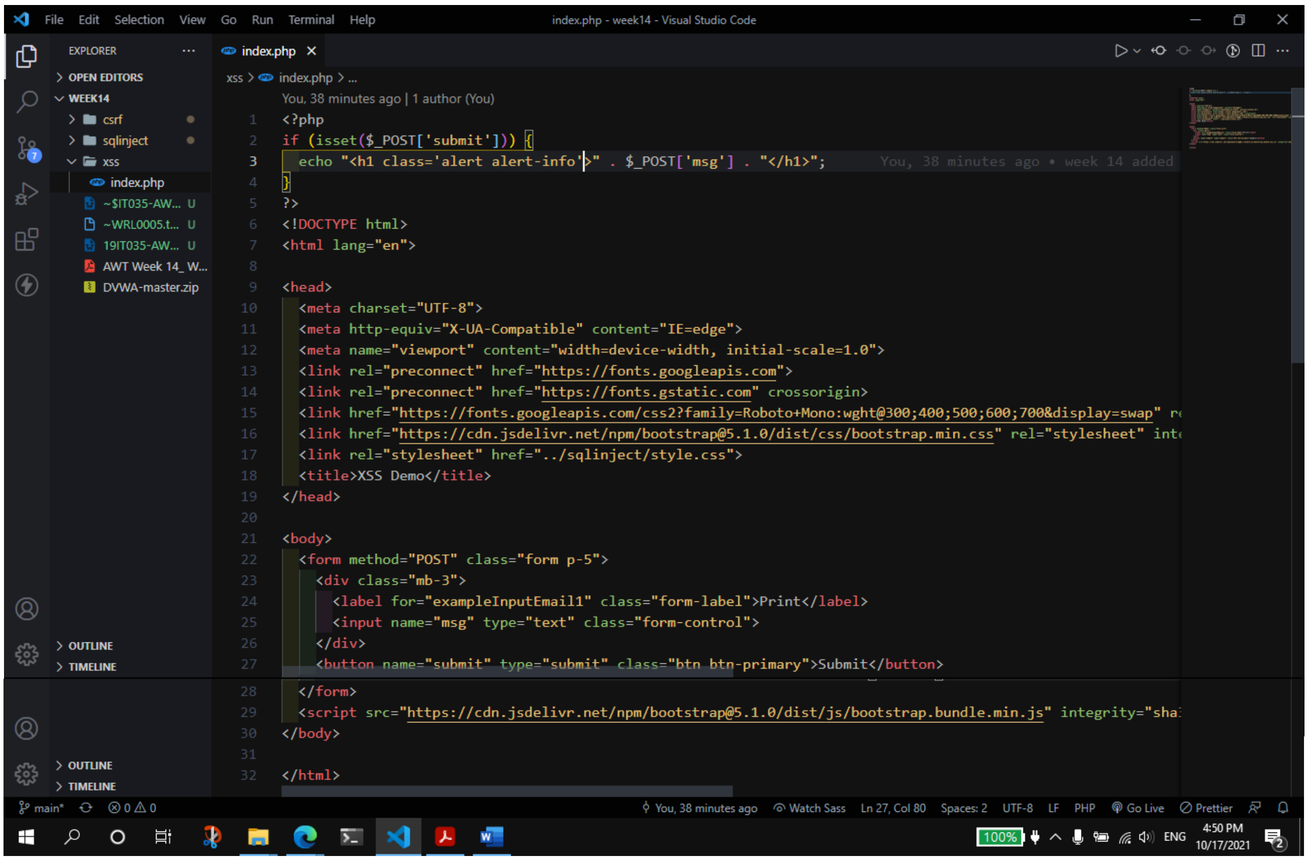
Enter Captcha

Captcha Code 93a29d

Send

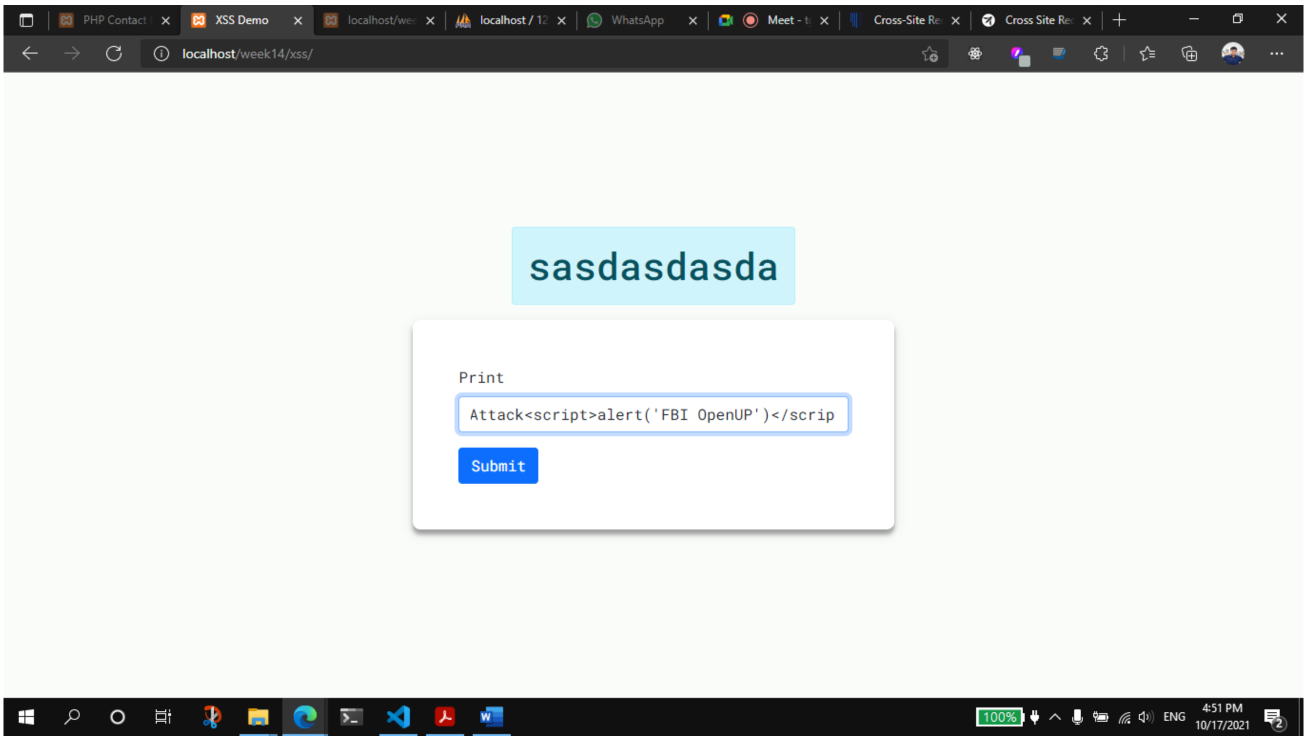
3) Demonstrate and Prevent XSS Attack in PHP App.

Code:



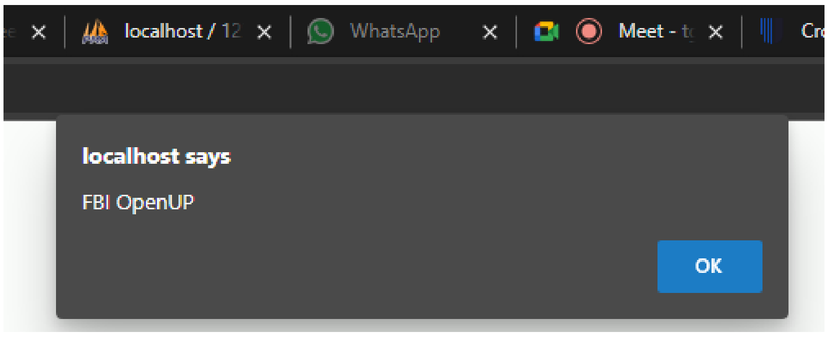
```
<?php
if (isset($_POST['submit'])) {
  echo "<h1 class='alert alert-info'>" . $_POST['msg'] . "</h1>";
}
?>
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  <link href="https://fonts.googleapis.com/css2?family=Roboto+Mono:wght@300;400;500;600;700&display=swap" r
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/css/bootstrap.min.css" rel="stylesheet" int
  <link rel="stylesheet" href="../sqlinject/style.css">
  <title>XSS Demo</title>
</head>

<body>
  <form method="POST" class="form p-5">
    <div class="mb-3">
      <label for="exampleInputEmail1" class="form-label">Print</label>
      <input name="msg" type="text" class="form-control">
    </div>
    <button name="submit" type="submit" class="btn btn-primary">Submit</button>
  </form>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/js/bootstrap.bundle.min.js" integrity="sha
</body>

</html>
```

Output:

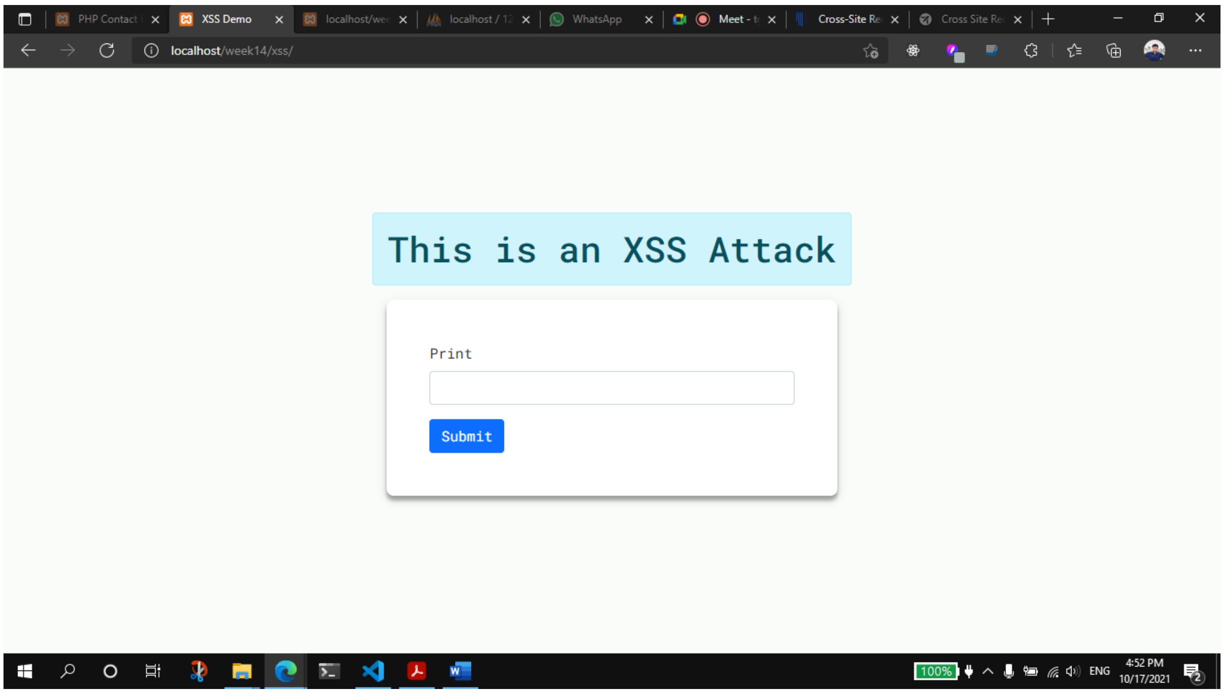


sasdasdasda

Attack<script>alert('FBI OpenUP')</scrip

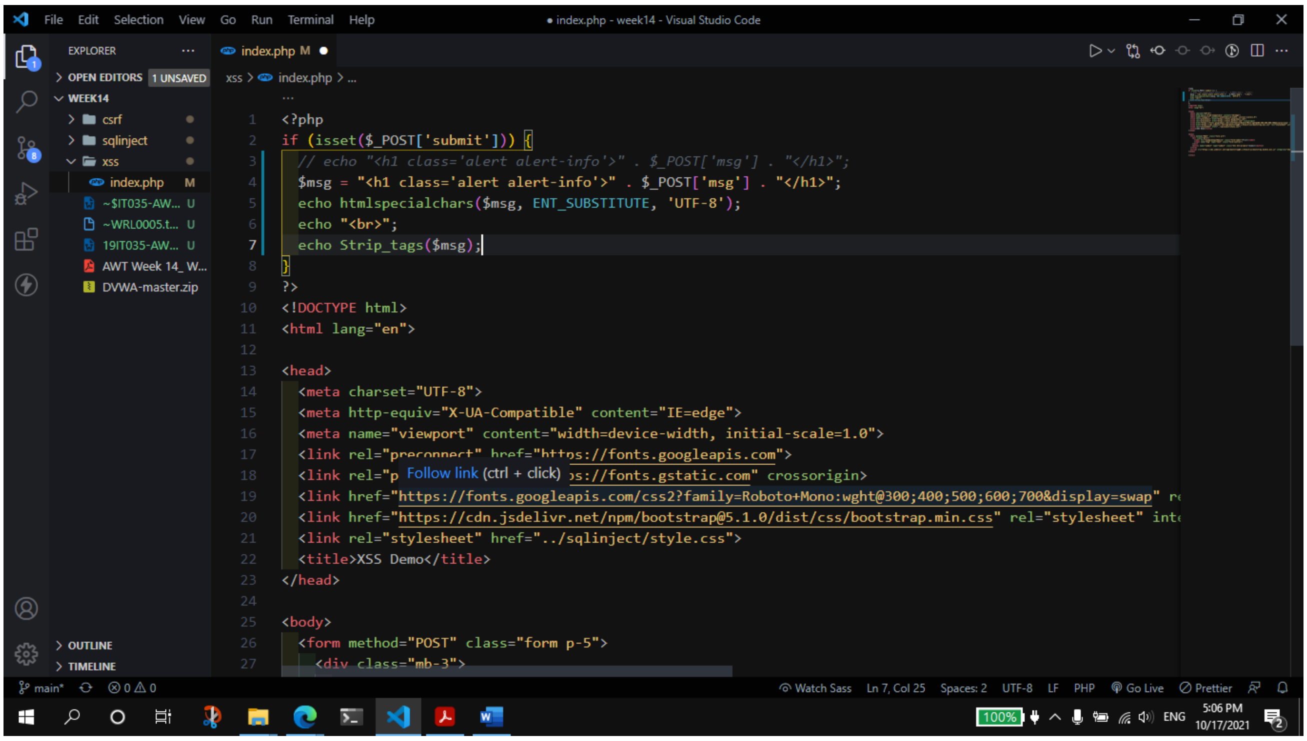

localhost says

FBI OpenUP

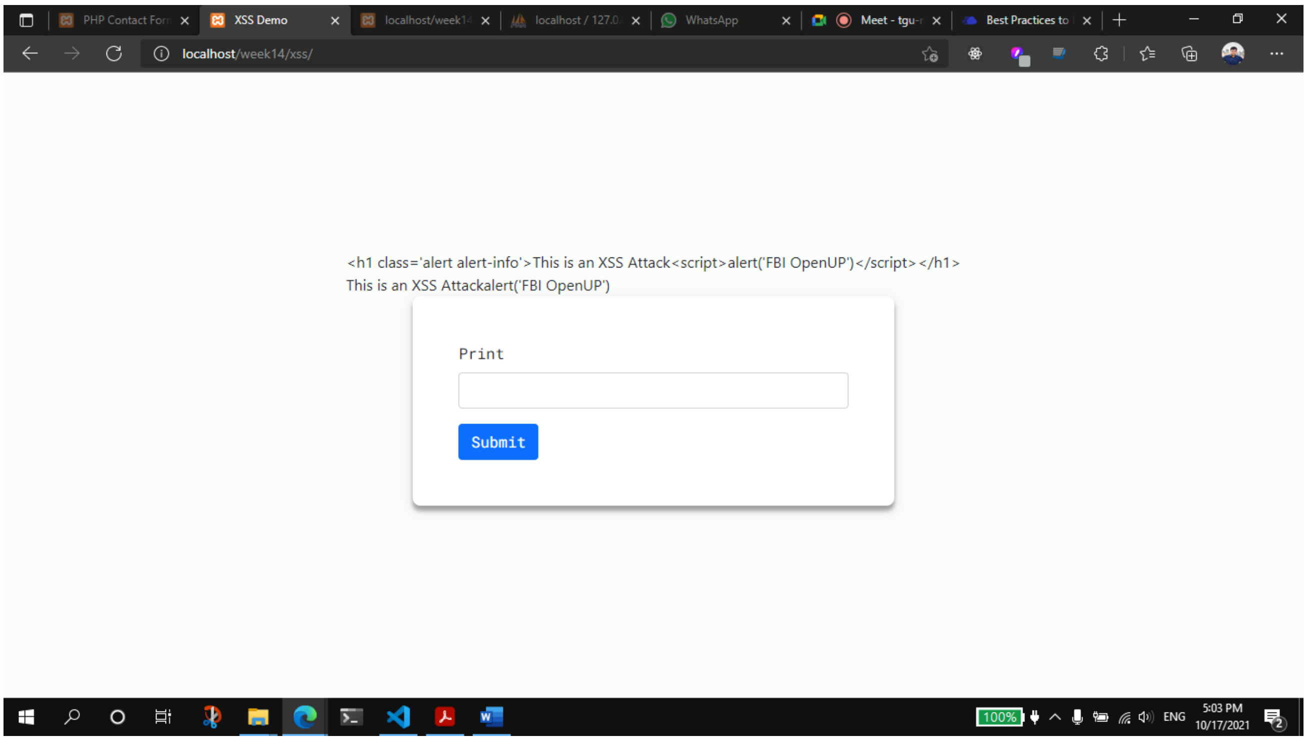


This is an XSS Attack

Prevention:



```
<?php
if (isset($_POST['submit'])) {
  // echo "<h1 class='alert alert-info'>" . $_POST['msg'] . "</h1>";
  $msg = "<h1 class='alert alert-info'>" . $_POST['msg'] . "</h1>";
  echo htmlspecialchars($msg, ENT_SUBSTITUTE, 'UTF-8');
  echo "<br>";
  echo Strip_tags($msg);
}
?>
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="p Follow link (ctrl + click) ps://fonts.gstatic.com" crossorigin>
  <link href="https://fonts.googleapis.com/css2?family=Roboto+Mono:wght@300;400;500;600;700&display=swap" r
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/css/bootstrap.min.css" rel="stylesheet" int
  <link rel="stylesheet" href="../sqlinject/style.css">
  <title>XSS Demo</title>
</head>

<body>
  <form method="POST" class="form p-5">
    <div class="mb-3">
```



localhost/week14/xss/

```
<h1 class='alert alert-info'>This is an XSS Attack<script>alert('FBI OpenUP')</script></h1>
This is an XSS Attackalert('FBI OpenUP')
```

- Whenever need to print something should use htmlspecialchars() function to print, this function returns the string, and the code does not execute In this string.
- Should use strip_tag() function to print the msg, this function remove the HTML tags and returns the string.