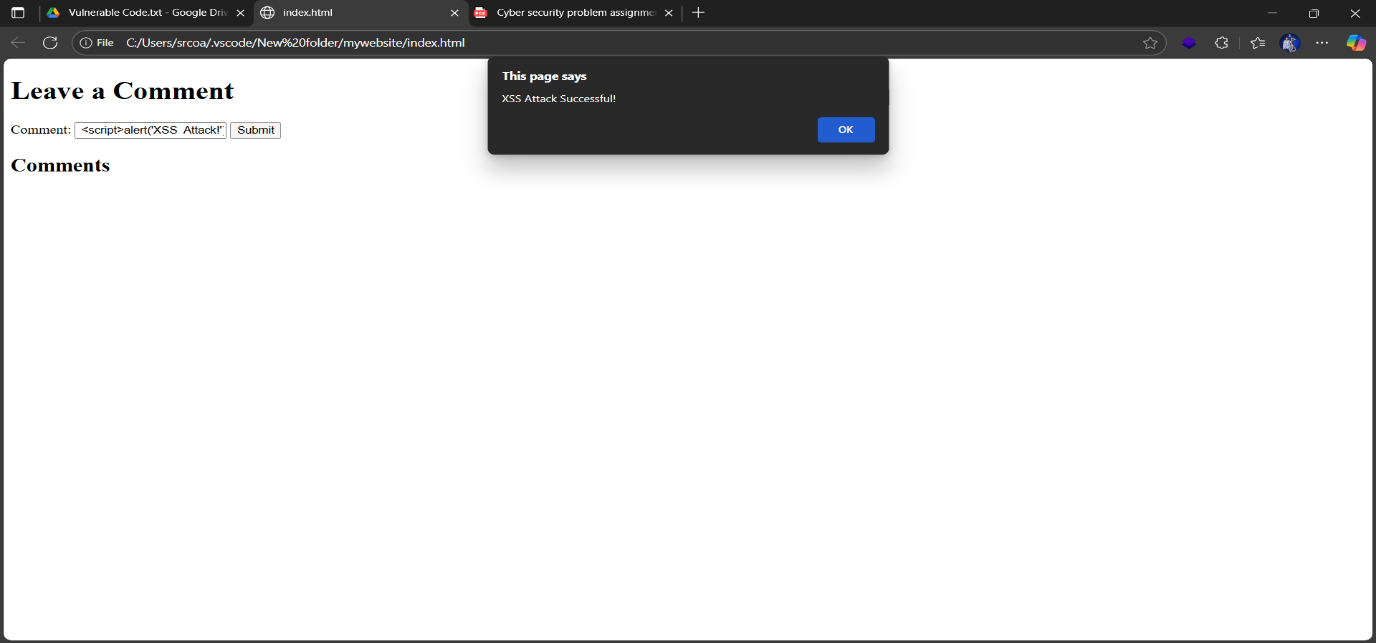
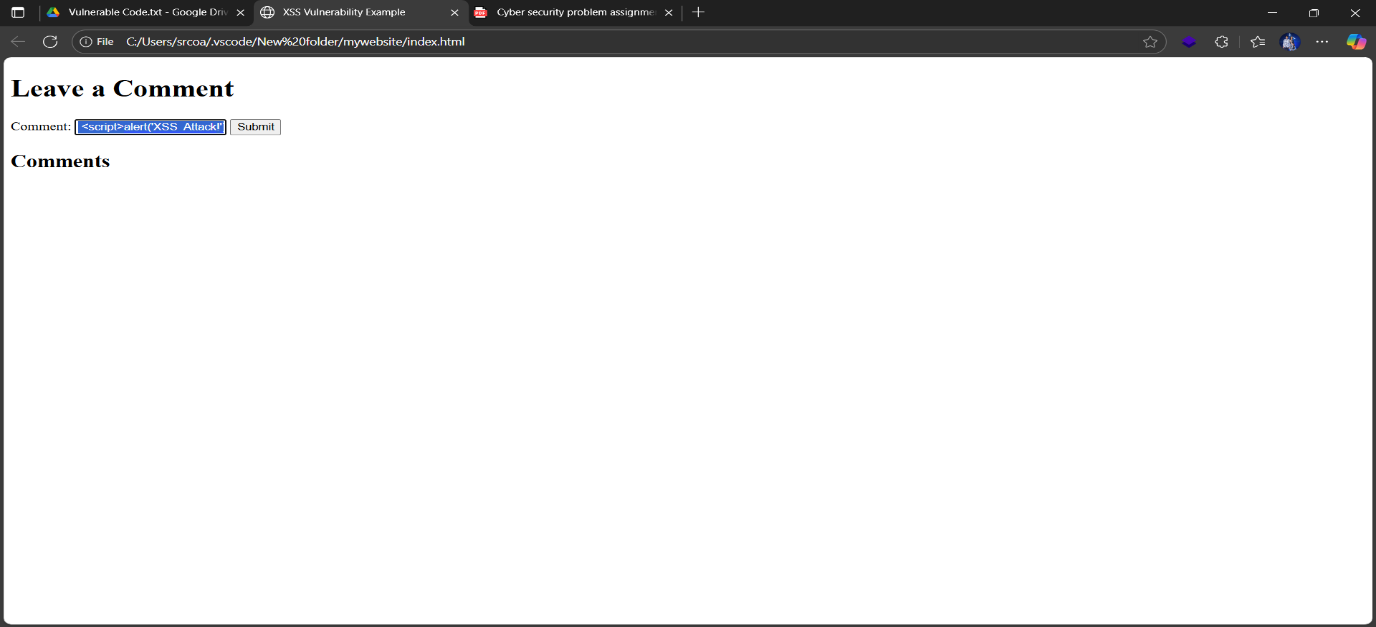
**ASSIGNMENT:-2 Solve the XSS Vulnerability**



Screenshot 1: show the Alert On Vulnerable programme



Screenshot 2 : input Vulnerable code in comment field

This code is vulnerable because Its show XSS Script Alert

**\*\* Vulnerability Explanation \*\***

**Problem Areas:**

1. InnerHTML usage:

• var comment = document.getElementById('comment').value;

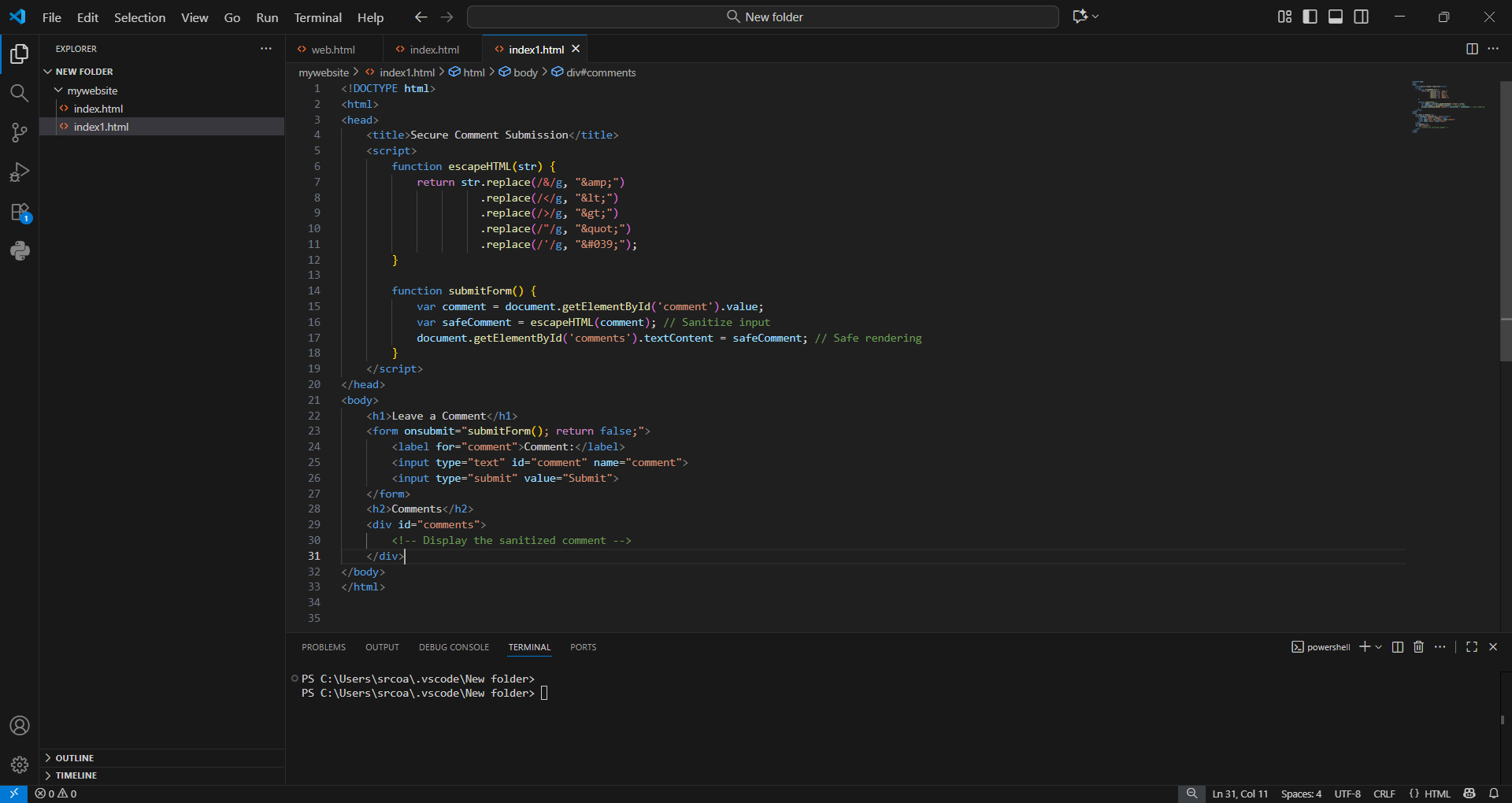
This directly injects user input into the DOM, allowing malicious scripts like <script>alert(‘XSS’);</script> to execute.

2. document.write(); with script injection:

• document.write(scriptTag);

This forcibly writes a script tag into the document, which is a classic XSS vector.

**\*\* Secure Version (XSS Mitigation) \*\***

****

**Screenshot 3 : xss mitigation code on Vscode**

**Full Code :**

<!DOCTYPE html>

<html>

<head>

    <title>Secure Comment Submission</title>

    <script>

        function escapeHTML(str) {

            return str.replace(/&/g, "&amp;")

                      .replace(/</g, "&lt;")

                      .replace(/>/g, "&gt;")

                      .replace(/"/g, "&quot;")

                      .replace(/'/g, "&#039;");

        }

        function submitForm() {

            var comment = document.getElementById('comment').value;

            var safeComment = escapeHTML(comment); // Sanitize input

            document.getElementById('comments').textContent = safeComment; // Safe rendering

        }

    </script>

</head>

<body>

    <h1>Leave a Comment</h1>

    <form onsubmit="submitForm(); return false;">

        <label for="comment">Comment:</label>

        <input type="text" id="comment" name="comment">

        <input type="submit" value="Submit">

    </form>

    <h2>Comments</h2>

    <div id="comments">

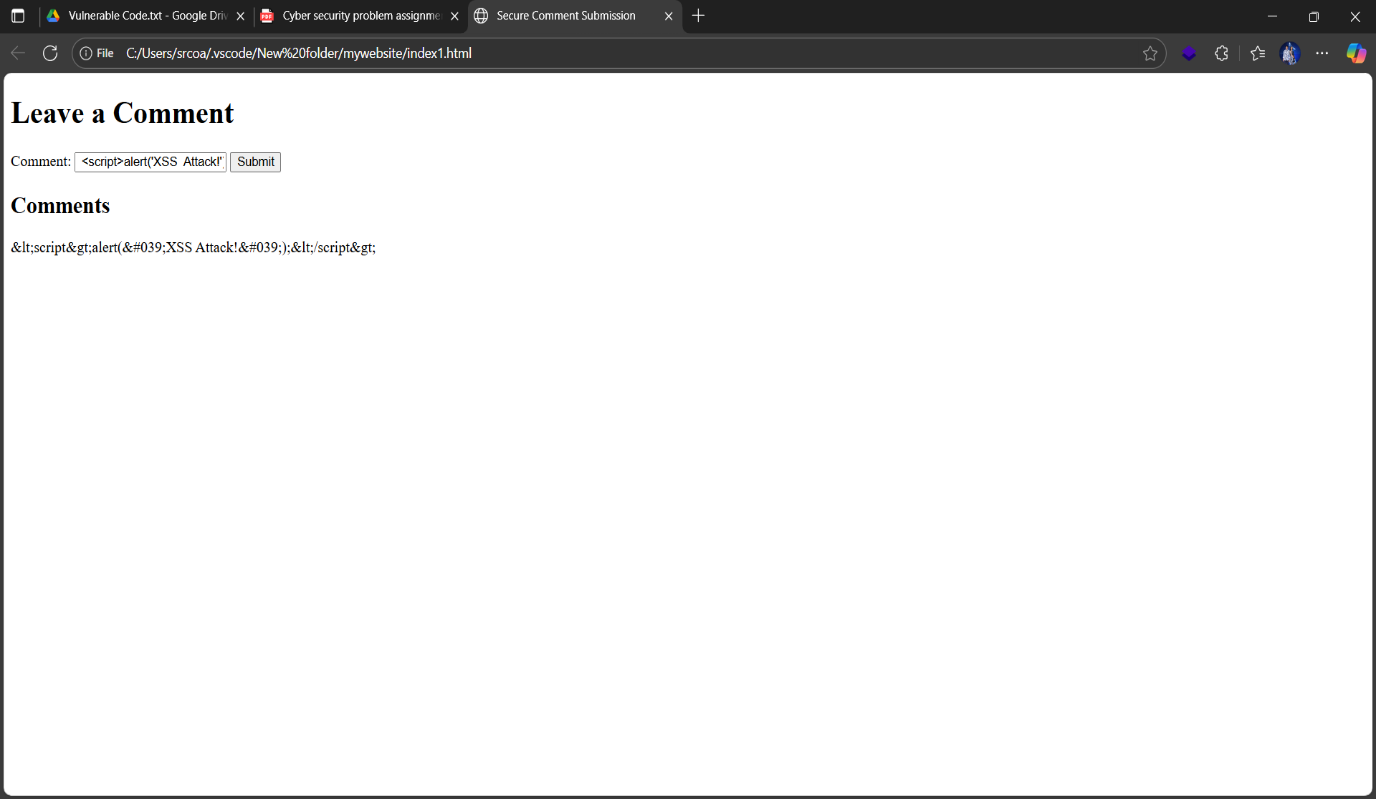
        <!-- Display the sanitized comment -->

    </div>

</body>

</html>

**Screenshot 4 : xss mitigation Code**

****

**Screenshot 5 : XSS Vulnerability Fixed Input Comment Field Sanitized**

**Alert not shown**

**@@ Why This Fix Works:**

• Escaping HTML characters prevents script tags from being interpreted by the browser.

• Using TextContant instead of innerhtml ensures that even if malicious input slips through, it’s treated as plain text—not executable code.

• Avoiding document.write() eliminates a major XSS vector