**Web Engineering (LAB) 2**

**Name: Hamood Ahmed**

**ID: 022-20-119694**

**Program: BSCS**

**Subject Web Engineering (LAB)**

**Course Instructor: Sir Syed Atir Iftikhar**

Name: Hamood Ahmed Lab#2 STD ID: 19694

**Task1**

**9. Lab Task 1. Create a Login Form using HTML**

**Objective:**

a. Add Title (using h1 or h2 tag)

b. Add a div

c. Add a form element within the div

d. Add Login ID, Password textboxes and group them in a section (using div for section and<inputtype=”text” … for textboxes) and put the section under the form

e. Set Login ID and Password as mandatory inputs (using required element attribute)

f. Add OK and Cancel buttons (using <input type=”button” …..)

**Source Code:**

**Web Engineering**

Name: Hamood Ahmed Lab#2 STD ID: 19694

**Task1**

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metaname="viewport"content="width=device-width, initial-scale=1.0">

    <title>Login Form</title>

    <style>

      body {

            font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;

            margin: 50px;

        }

        .container {

            max-width: 320px;

            margin: auto;

            padding: 20px;

            border: 1pxsolid#ccc;

            border-radius: 5px;

            box-shadow: 0010pxrgba(0, 0, 0, 0.1);

        }

        .section {

            margin-bottom: 15px;

        }

        input[type="text"], input[type="password"] {

            width: 100%;

            padding: 10px;

            margin-top: 5px;

            border: 1pxsolid#ccc;

            border-radius: 4px;

        }

        input[type="button"] {

            padding: 10px15px;

            margin: 5px;

            border: none;

            border-radius: 4px;

            cursor: pointer;

        }

        input[type="button"].ok {

**Web Engineering**

Name: Hamood Ahmed Lab#2 STD ID: 19694

**Task1**

            background-color: #7b3fac; /\* Green \*/

            color: white;

        }

        input[type="button"].cancel {

            background-color: #3d51c7; /\* Red \*/

            color: white;

        }

    </style>

</head>

<body>

<h2>Login Form</h2>

<divclass="container">

    <form>

        <divclass="section">

            <labelfor="loginID">Login ID:</label>

            <inputtype="text"id="loginID"name="loginID"required>

        </div>

        <divclass="section">

            <labelfor="password">Password:</label>

            <inputtype="password"id="password"name="password"required>

        </div>

        <inputtype="button"value="OK"class="ok">

        <inputtype="button"value="Cancel"class="cancel">

    </form>

</div>

</body>

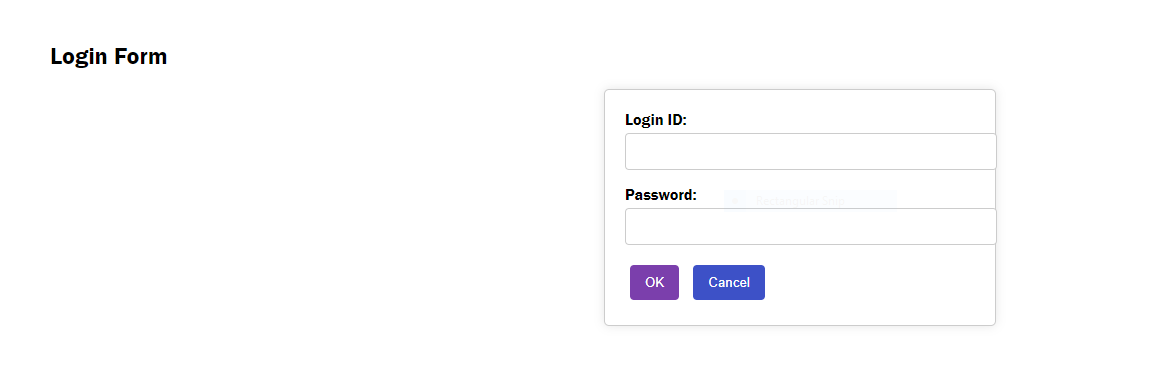
</html>

**Web Engineering**

Name: Hamood Ahmed Lab#2 STD ID: 19694

**Task1**

Output:

****

Name: Hamood Ahmed Lab#2 STD ID: 19694

**Task 2**

**Objective:**

2. Add styling to the Login Form using CSS

a. Create a separate style file and link it with HTML file within the <head> section using <link> tag

b. Add color to title, labels, inputs, and buttons text (color)

c. Add borders to the textboxes (pixels, color, and border style)

d. Right-align labels (using text-align of the parent/containing element)

e. Add border to the form element (same as step but with different color)

**Web Engineering**

f. Center-align the form on the screen (using width and margin-left and margin-right set as auto)

Name: Hamood Ahmed Lab#2 STD ID: 19694

**Task 2**

**Source Code:**

**HTML File:**

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metaname="viewport"content="width=device-width, initial-scale=1.0">

    <title>Login Form</title>

    <linkrel="stylesheet"href="style.css">

</head>

<body>

<h2>Login Form</h2>

<divclass="container">

    <form>

        <divclass="section">

            <labelfor="loginID">Login ID:</label>

            <inputtype="text"id="loginID"name="loginID"required>

        </div>

        <divclass="section">

            <labelfor="password">Password:</label>

            <inputtype="password"id="password"name="password"required>

        </div>

        <inputtype="button"value="OK"class="ok">

        <inputtype="button"value="Cancel"class="cancel">

    </form>

</div>

</body>

</html>

**Web Engineering**

Name: Hamood Ahmed Lab#2 STD ID: 19694

**Task 2**

**CSS Fie:**

body {

    font-family: Arial, sans-serif;

    margin: 0;

    padding: 50px;

    background-color: #f4f4f4;

}

h2 {

    color: #4CAF50;

    text-align: center;

}

.container {

    max-width: 400px;

    margin: 0auto;

    padding: 20px;

    border: 2pxsolid#4CAF50;

    border-radius: 5px;

    background-color: white;

    box-shadow: 0010pxrgba(0, 0, 0, 0.1);

}

.section {

    margin-bottom: 15px;

}

label {

    display: block;

    text-align: right;

    color: #333;

}

input[type="text"], input[type="password"] {

    width: calc(100%-22px);

    padding: 10px;

    margin-top: 5px;

    border: 1pxsolid#ccc;

    border-radius: 4px;

    color: #333;

}

input[type="button"] {

    padding: 10px15px;

    margin: 5px;

    border: none;

    border-radius: 4px;

    cursor: pointer;

    color: rgb(233, 227, 220);

}

input[type="button"].ok {

    background-color: #806316;

}

input[type="button"].cancel {

    background-color: #12503d;

}

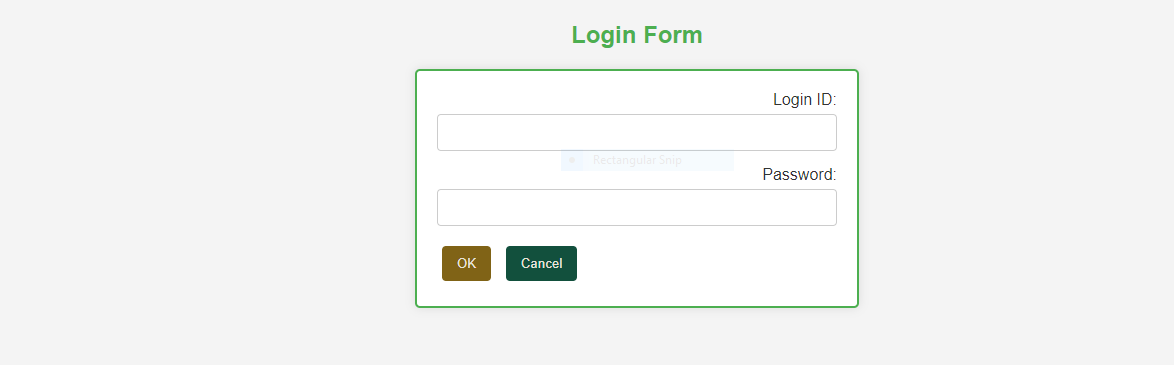
**Web Engineering**

Name: Hamood Ahmed Lab#2 STD ID: 19694

**Task 2**

Output:

**Web Engineering**

****

Name: Hamood Ahmed Lab#2 STD ID: 19694

**Task 3**

**Objective:**

3. Add interactions for its OK and Cancel buttons using JS

a. Add click event for both OK and Cancel buttons (using onclick element event

attribute)

b. When OK button is clicked, show alert with “Login is clicked”

message (using

alert(“yourMessage”))

c. When Cancel button is clicked, show alert with “Cancel is clicked” message (same as

step b)

**Source Code:**

**HTML File:**

<!DOCTYPEhtml>

<htmllang="en">

<head>

    <metacharset="UTF-8">

    <metaname="viewport"content="width=device-width, initial-scale=1.0">

    <title>Login Form</title>

    <linkrel="stylesheet"href="style.css">

    <script>

        functionhandleOkClick() {

            alert("Login is clicked");

        }

        functionhandleCancelClick() {

            alert("Cancel is clicked");

        }

    </script>

</head>

<body>

<h2>Login Form</h2>

<divclass="container">

    <form>

        <divclass="section">

            <labelfor="loginID">Login ID:</label>

            <inputtype="text"id="loginID"name="loginID"required>

        </div>

        <divclass="section">

            <labelfor="password">Password:</label>

            <inputtype="password"id="password"name="password"required>

        </div>

        <inputtype="submit"value="OK"class="ok"onclick="handleOkClick()">

        <inputtype="submit"value="Cancel"class="cancel"onclick="handleCancelClick()">

    </form>

</div>

</body>

</html>

**Web Engineering**

**CSS File:**

body {

    font-family: Arial, sans-serif;

    margin: 0;

    padding: 50px;

    background-color: #f4f4f4;

}

h2 {

    color: #4CAF50;

    text-align: center;

}

.container {

    max-width: 400px;

    margin: 0auto;

    padding: 20px;

    border: 2pxsolid#4CAF50;

    border-radius: 5px;

    background-color: white;

    box-shadow: 0010pxrgba(0, 0, 0, 0.1);

}

.section {

    margin-bottom: 15px;

}

label {

    display: block;

    text-align: right;

    color: #333;

}

input[type="text"], input[type="password"] {

    width: calc(100%-22px);

    padding: 10px;

    margin-top: 5px;

    border: 1pxsolid#ccc;

    border-radius: 4px;

    color: #333;

}

input[type="submit"] {

    padding: 10px15px;

    margin: 5px;

    border: none;

    border-radius: 4px;

    cursor: pointer;

    color: rgb(233, 227, 220);

}

input[type="submit"].ok {

    background-color: #806316;

}

input[type="submit"].cancel {

    background-color: #12503d;

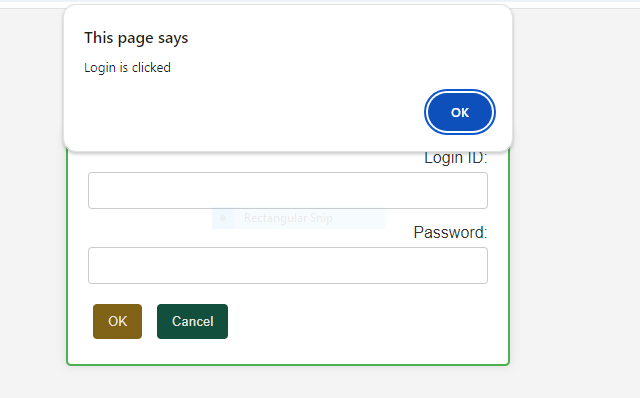
}

**Web Engineering**

Name: Hamood Ahmed Lab#2 STD ID: 19694

**Task 3**

Output:

****

**Web Engineering**

Name: Hamood Ahmed Lab#2 STD ID: 19694

**Task 4**

**Objective:**

10. Homework Tasks

1. Enhance the Login Form created in Lab Task above using HTML.

a. Add Remember Me Check Box

b. Add an image beside the form

**Source Code:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Login Form</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            margin: 0;

            padding: 50px;

            background-color: #6f561c;

        }

        h2 {

            color: #4CAF50;

            text-align: center;

        }

        .container {

            max-width: 400px;

            margin: 0 auto;

            padding: 20px;

            border: 2px solid #4CAF50;

            border-radius: 5px;

            background-color: rgb(179, 237, 229);

            box-shadow: 0 0 10px rgba(127, 39, 39, 0.1);

        }

        .section {

            margin-bottom: 15px;

        }

        label {

            display: block;

            text-align: right;

            color: #333;

        }

        input[type="text"], input[type="password"] {

            width: calc(100% - 22px);

            padding: 10px;

            margin-top: 5px;

            border: 1px solid #ccc;

            border-radius: 4px;

            color: #333;

        }

        input[type="checkbox"] {

            margin-left: 10px;

        }

        input[type="submit"] {

            padding: 10px 15px;

            margin: 5px;

            border: none;

            border-radius: 4px;

            cursor: pointer;

            color: rgb(233, 227, 220);

        }

        input[type="submit"].ok {

            background-color: #806316;

        }

        input[type="submit"].cancel {

            background-color: #12503d;

        }

        .form-container {

            display: flex;

            align-items: center;

        }

        .form-container img {

            width: 100px;

            height: auto;

            margin-right: 20px;

        }

    </style>

    <script>

        function handleOkClick() {

            alert("Login is clicked");

        }

        function handleCancelClick() {

            alert("Cancel is clicked");

        }

    </script>

</head>

<body>

<h2>Login Form</h2>

<div class="container">

    <div class="form-container">

        <img src="keyboard.png" alt="Login Image">

        <form>

            <div class="section">

                <label for="loginID">Login ID:</label>

                <input type="text" id="loginID" name="loginID" required>

            </div>

            <div class="section">

                <label for="password">Password:</label>

                <input type="password" id="password" name="password" required>

            </div>

            <div class="section">

                <input type="checkbox" id="rememberMe" name="rememberMe">

                <label for="rememberMe" style="display: inline; text-align: left;">Remember Me</label>

            </div>

            <input type="submit" value="OK" class="ok" onclick="handleOkClick()">

            <input type="submit" value="Cancel" class="cancel" onclick="handleCancelClick()">

        </form>

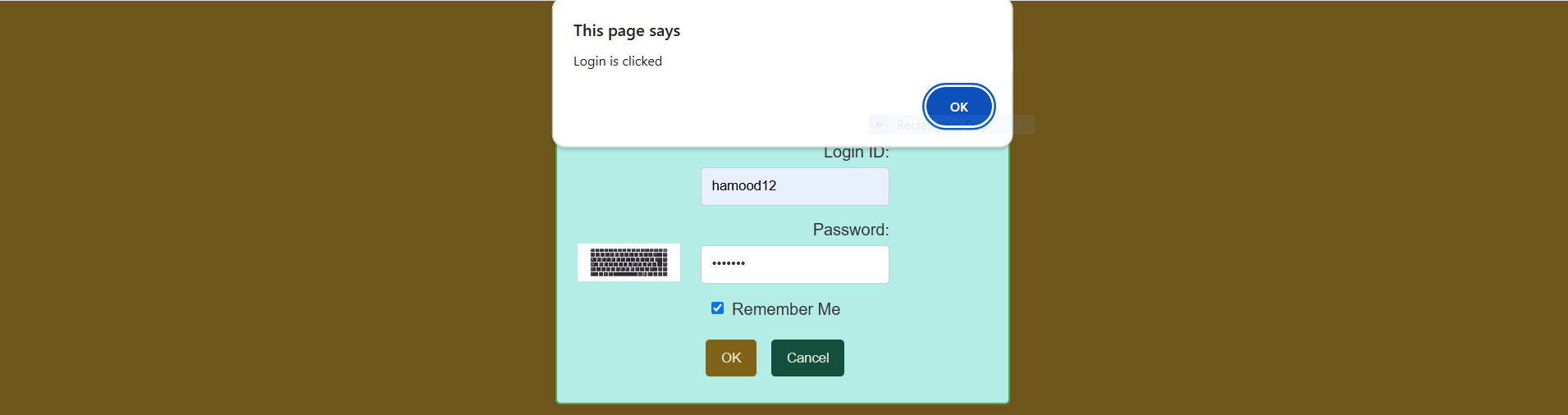
    </div>

</div>

</body>

</html>

Output:



**Web Engineering**

Name: Hamood Ahmed Lab#2 STD ID: 19694

**Task 5**

**Objective:**

2. Style the Login Form using CSS

a. Change the background-color of the textboxes when focus is set.

b. Set the image width and height

c. Position the image on the top-left corner using absolute and relative position combinations.

**Source Code:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Login Form Using CSS</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            margin: 0;

            padding: 50px;

            background-color: #dc643b;

        }

        h2 {

            color: #830c0c;

            text-align: center;

        }

        .container {

            max-width: 400px;

            margin: 0 auto;

            padding: 20px;

            border: 2px solid #4CAF50;

            border-radius: 5px;

            background-color: rgb(132, 136, 29);

            box-shadow: 0 0 10px rgba(127, 39, 39, 0.1);

            position: relative;

        }

        .section {

            margin-bottom: 15px;

        }

        label {

            display: block;

            text-align: right;

            color: #333;

        }

        input[type="text"], input[type="password"] {

            width: calc(100% - 22px);

            padding: 10px;

            margin-top: 5px;

            border: 1px solid #ccc;

            border-radius: 4px;

            color: #333;

            transition: background-color 0.3s ease;

        }

        input[type="text"]:focus, input[type="password"]:focus {

            background-color: #e0f7fa;

        }

        input[type="checkbox"] {

            margin-left: 10px;

        }

        input[type="submit"] {

            padding: 10px 15px;

            margin: 5px;

            border: none;

            border-radius: 4px;

            cursor: pointer;

            color: rgb(233, 227, 220);

        }

        input[type="submit"].ok {

            background-color: #806316;

        }

        input[type="submit"].cancel {

            background-color: #12503d;

        }

        .form-container {

            display: flex;

            align-items: center;

        }

        .form-container img {

            width: 80px;

            height: 80px;

            position: absolute;

            top: -40px;

            left: -40px;

            border-radius: 50%;

        }

    </style>

    <script>

        function handleOkClick() {

            alert("Login is clicked");

        }

        function handleCancelClick() {

            alert("Cancel is clicked");

        }

    </script>

</head>

<body>

<h2>Login Form</h2>

<div class="container">

    <div class="form-container">

        <img src="horse.png" alt="Login Image">

        <form>

            <div class="section">

                <label for="loginID">Login ID:</label>

                <input type="text" id="loginID" name="loginID" required>

            </div>

            <div class="section">

                <label for="password">Password:</label>

                <input type="password" id="password" name="password" required>

            </div>

            <div class="section">

                <input type="checkbox" id="rememberMe" name="rememberMe">

                <label for="rememberMe" style="display: inline; text-align: left;">Remember Me</label>

            </div>

            <input type="submit" value="OK" class="ok" onclick="handleOkClick()">

            <input type="submit" value="Cancel" class="cancel" onclick="handleCancelClick()">

        </form>

    </div>

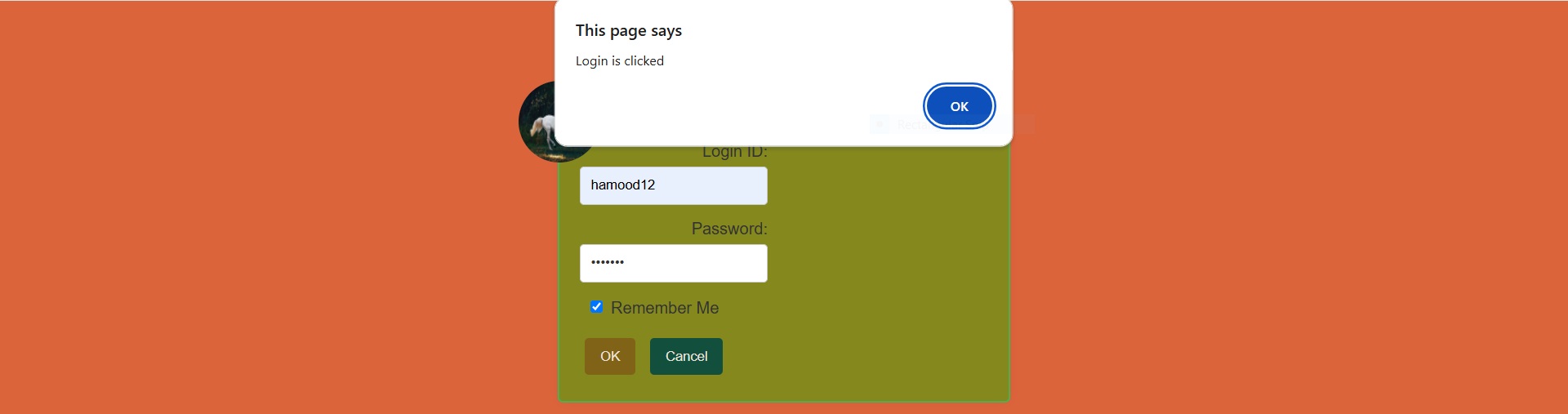
</div>

</body>

</html>

**Web Engineering**

Output:



Name: Hamood Ahmed Lab#2 STD ID: 19694

**Task 6**

**Objective:**

3. Extend the OK and Cancel button functionality by using JS

a. Create a function called findInputs() and return all the inputs in an array. Call the function from theOK button click and assign the output to an array.

b. Create a function called validateInputs() and return true or false after looping through each input element obtained from the findInputs() function and verifiying if mandatory Login ID and Passwordare filled in or not. Call the validateInputs() function from the OK button click after findInputs() function.

c. Show message to enter input if validateInput() output is false and return from the click event.

d. Create a function called login() and check if Remember Me is checked or not. If it is checked, showmessage in an alert as “Logged in successfully with Remember Me Marked as Set”, OTHERWISE,just show “Logged in Successfully”. Call the login() function on button click.

e. Create a function called clear() and clear all the inputs. Call the function from Cancel button when it is clicked

**Source Code:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>OK and Cancel Button Functionality By Using JS.</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            margin: 0;

            padding: 50px;

            background-color: #6659aa;

        }

        h2 {

            color: #ead61f;

            text-align: center;

        }

        .container {

            max-width: 400px;

            margin: 0 auto;

            padding: 20px;

            border: 2px solid #4CAF50;

            border-radius: 5px;

            background-color: rgb(124, 213, 120);

            box-shadow: 0 0 10px rgba(127, 39, 39, 0.1);

            position: relative;

        }

        .section {

            margin-bottom: 15px;

        }

        label {

            display: block;

            text-align: right;

            color: #333;

        }

        input[type="text"], input[type="password"] {

            width: calc(100% - 22px);

            padding: 10px;

            margin-top: 5px;

            border: 1px solid #ccc;

            border-radius: 4px;

            color: #333;

            transition: background-color 0.3s ease;

        }

        input[type="text"]:focus, input[type="password"]:focus {

            background-color: #e0f7fa;

        }

        input[type="checkbox"] {

            margin-left: 10px;

        }

        input[type="submit"] {

            padding: 10px 15px;

            margin: 5px;

            border: none;

            border-radius: 4px;

            cursor: pointer;

            color: rgb(233, 227, 220);

        }

        input[type="submit"].ok {

            background-color: #806316;

        }

        input[type="submit"].cancel {

            background-color: #12503d;

        }

        .form-container {

            display: flex;

            align-items: center;

        }

        .form-container img {

            width: 80px;

            height: 80px;

            position: absolute;

            top: -40px;

            left: -40px;

            border-radius: 50%;

        }

    </style>

    <script>

        function findInputs() {

            return document.querySelectorAll("input[type='text'], input[type='password'], input[type='checkbox']");

        }

        function validateInputs() {

            const inputs = findInputs();

            const loginID = inputs[0].value;

            const password = inputs[1].value;

            if (loginID === "" || password === "") {

                alert("Please enter both Login ID and Password.");

                return false;

            }

            return true;

        }

        function login() {

            const inputs = findInputs();

            const rememberMe = inputs[2].checked;

            if (rememberMe) {

                alert("Logged in successfully with Remember Me Marked as Set.");

            } else {

                alert("Logged in Successfully.");

            }

        }

        function clear() {

            const inputs = findInputs();

            inputs.forEach(input => {

                if (input.type !== "checkbox") {

                    input.value = "";

                } else {

                    input.checked = false;

                }

            });

        }

        function handleOkClick(event) {

            event.preventDefault();

            if (validateInputs()) {

                login();

            }

        }

        function handleCancelClick(event) {

            event.preventDefault();

            clear();

        }

    </script>

</head>

<body>

<h2>Login Form</h2>

<div class="container">

    <div class="form-container">

        <img src="horse.png" alt="Login Image">

        <form>

            <div class="section">

                <label for="loginID">Login ID:</label>

                <input type="text" id="loginID" name="loginID" required>

            </div>

            <div class="section">

                <label for="password">Password:</label>

                <input type="password" id="password" name="password" required>

            </div>

            <div class="section">

                <input type="checkbox" id="rememberMe" name="rememberMe">

                <label for="rememberMe" style="display: inline; text-align: left;">Remember Me</label>

            </div>

            <input type="submit" value="OK" class="ok" onclick="handleOkClick(event)">

            <input type="submit" value="Cancel" class="cancel" onclick="handleCancelClick(event)">

        </form>

    </div>

</div>

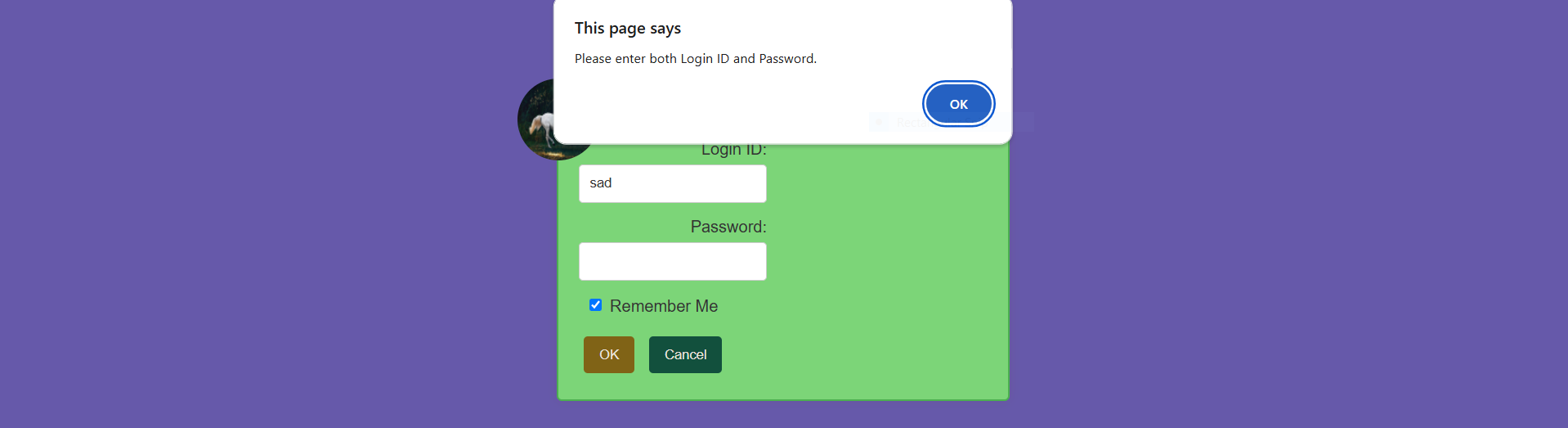
</body>

</html>

**Web Engineering**

Output

Snap 1(To Enter Login ID and Password):

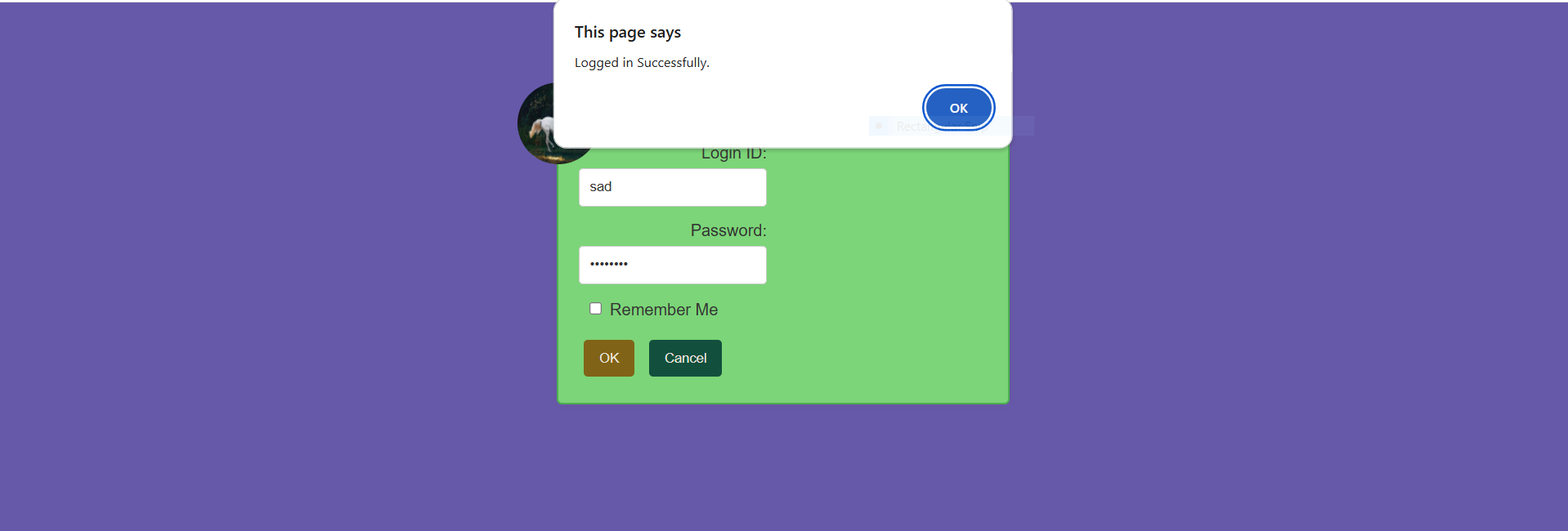


Snap 2(When Logged In Successfully):

**Web Engineering**

Name: Hamood Ahmed Lab#2 STD ID: 19694

**Task 6**



Snap 3(When Logged In Successfully With Remember Me Marked):



**Web Engineering**

**Web Engineering (LAB) 3**

**Name: Hamood Ahmed**

**ID: 022-20-119694**

**Program: BSCS**

**Subject: Web Engineering (LAB)**

**Course Instructor: Sir Syed Atir Iftikhar**

Name: Hamood Ahmed Lab#3 STD ID: 19694

**Task 1**

1. LabTask
   1. Extend theTask7 code and write a function to update a student batch year in the studentTable.
      1. Ask the Student # and Batch from the user using prompt alert. (You can ask from user in comma separated form)
      2. If users enter the data correctly,split the comma separated input.
      3. Find the row containing the‘student#’entered bythe user.
      4. Now,findthe‘batch’td element within the‘row’entered by the user.
      5. Finally update the batch entered by the user using ‘innerHTML’propertyof element.

**Code:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Lab 3 Tasks - Task 7 Code Extension</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            background-color: #f4f4f4;

            margin: 20px;

        }

        table {

            width: 100%;

            border-collapse: collapse;

            margin-bottom: 20px;

        }

        th, td {

            border: 1px solid #770d0d;

            text-align: left;

            padding: 8px;

        }

        th {

            background-color: #925917;

            color: white;

        }

        tr:nth-child(even) {

            background-color: #c56e45;

        }

        tr:hover {

            background-color: #d8d65a;

        }

        input[type="button"] {

            padding: 10px 15px;

            margin-right: 5px;

            background-color: #925917;

            color: white;

            border: none;

            cursor: pointer;

        }

        input[type="button"]:hover {

            background-color: #9238c5;

        }

    </style>

</head>

<body>

<table id="studentNames" class="students">

    <thead>

        <tr>

            <th>Student #</th>

            <th>Student Name</th>

            <th>Batch</th>

            <th>CNIC</th>

            <th>Action</th>

        </tr>

    </thead>

    <tbody>

        <tr id="row1" class="row">

            <td>Student 1</td>

            <td>Abdul Rehman</td>

            <td>2021</td>

            <td>42104-1786542-9</td>

            <td><button onclick="updateStudent('row1')">Update</button></td>

        </tr>

        <tr id="row2" class="row">

            <td>Student 2</td>

            <td>Hamood Ahmed</td>

            <td>2020</td>

            <td>42101-1426956-3</td>

            <td><button onclick="updateStudent('row2')">Update</button></td>

        </tr>

        <tr id="row3" class="row">

            <td>Student 3</td>

            <td>Manzoor Ahmed Rana</td>

            <td>2018</td>

            <td>42303-9653214-8</td>

            <td><button onclick="updateStudent('row3')">Update</button></td>

        </tr>

    </tbody>

</table>

<br>

<input type="button" value="Add New Student" onclick="addNewStudent();" />

<input type="button" value="Delete Student" onclick="deleteStudent();" />

<input type="button" value="Update Student Batch & CNIC" onclick="updateStudentBatch();" />

<script>

    function addNewStudent() {

        const body = document.querySelector('tbody');

        const row = document.createElement('tr');

        const tdStudentID = document.createElement('td');

        const tdStudentName = document.createElement('td');

        const tdBatch = document.createElement('td');

        const tdCNIC = document.createElement('td');

        const tdAction = document.createElement('td');

        const studentId = (Math.round(Math.random() \* 100)) + 4;

        const studentName = "New Student " + studentId;

        const batch = (Math.round(Math.random() \* 5)) + 2018;

        const cnic = Math.floor(10000 + Math.random() \* 90000) + "-" + Math.floor(1000000 + Math.random() \* 9000000) + "-" + Math.floor(1 + Math.random() \* 9);

        row.id = 'row' + studentId;

        tdStudentID.innerHTML = 'Student ' + studentId;

        tdStudentName.innerHTML = studentName;

        tdBatch.innerHTML = batch;

        tdCNIC.innerHTML = cnic;

        tdAction.innerHTML = '<button onclick="updateStudent(\'' + row.id + '\')">Update</button>';

        row.appendChild(tdStudentID);

        row.appendChild(tdStudentName);

        row.appendChild(tdBatch);

        row.appendChild(tdCNIC);

        row.appendChild(tdAction);

        body.appendChild(row);

    }

    function deleteStudent() {

        const studentId = prompt('Please enter Student # to delete:');

        const row = document.getElementById('row' + studentId);

        if (row) {

            const body = document.querySelector('tbody');

            body.removeChild(row);

        } else {

            alert('Student not found or invalid input.');

        }

    }

    function updateStudent(rowId) {

        const row = document.getElementById(rowId);

        if (row) {

            const updatedName = prompt('Enter updated student name:');

            const updatedBatch = prompt('Enter updated batch:');

            const updatedCNIC = prompt('Enter updated CNIC:');

            row.querySelector('td:nth-child(2)').textContent = updatedName;

            row.querySelector('td:nth-child(3)').textContent = updatedBatch;

            row.querySelector('td:nth-child(4)').textContent = updatedCNIC;

        } else {

            alert('Student not found or invalid input.');

        }

    }

    function updateStudentBatch() {

        const input = prompt('Enter Student #, new Batch, and CNIC (e.g., "1,2022,12345-6789012-3")');

        if (input) {

            const [studentNumber, newBatch, newCNIC] = input.split(',');

            const row = document.getElementById('row' + studentNumber.trim());

            if (row) {

                const batchCell = row.querySelector('td:nth-child(3)');

                const cnicCell = row.querySelector('td:nth-child(4)');

                batchCell.innerHTML = newBatch.trim();

                cnicCell.innerHTML = newCNIC.trim();

            } else {

                alert('Student not found or invalid input.');

            }

        }

    }

</script>

</body>

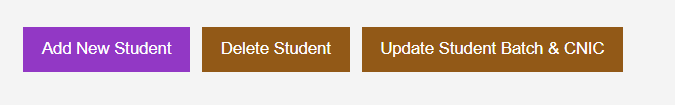
</html>

**Web Engineering**

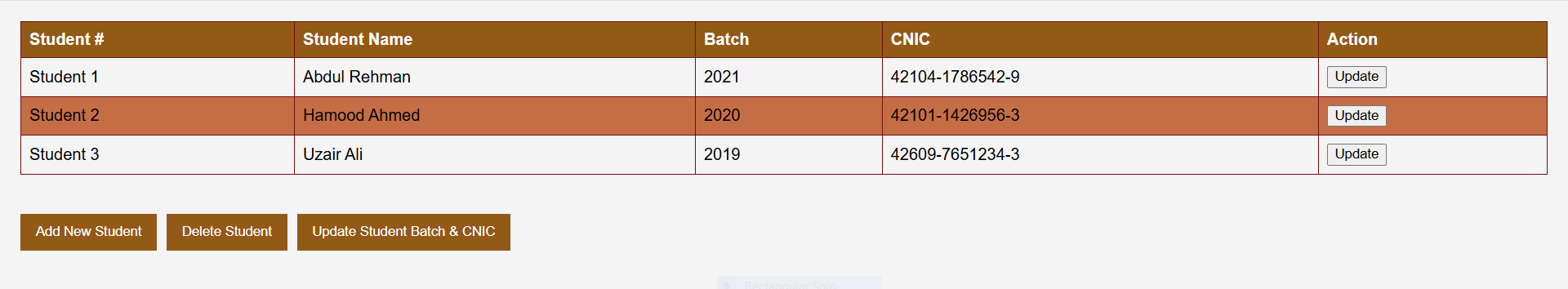
**Output (Add,Update and Delete Student):**

Name: Hamood Ahmed Lab#3 STD ID: 19694

**Task 1**



**When Updated To Add New Record:**



**To Change/Delete Any New Record:**

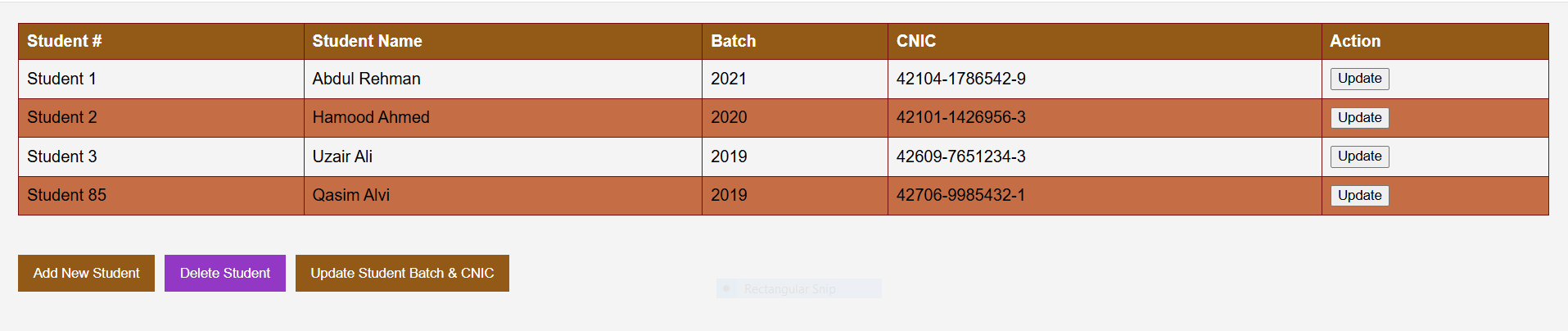


**For Changing/Deleting Any New Record Student 78 (Ali Zaidi) is Deleted:**

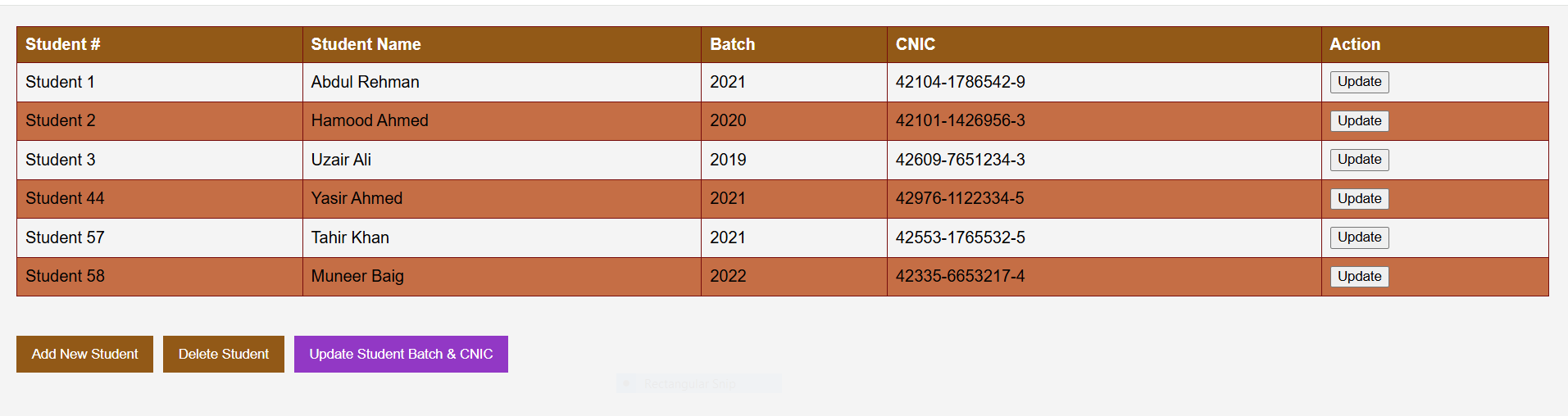
**Web Engineering**

Name: Hamood Ahmed Lab#3 STD ID: 19694

**Task 1**

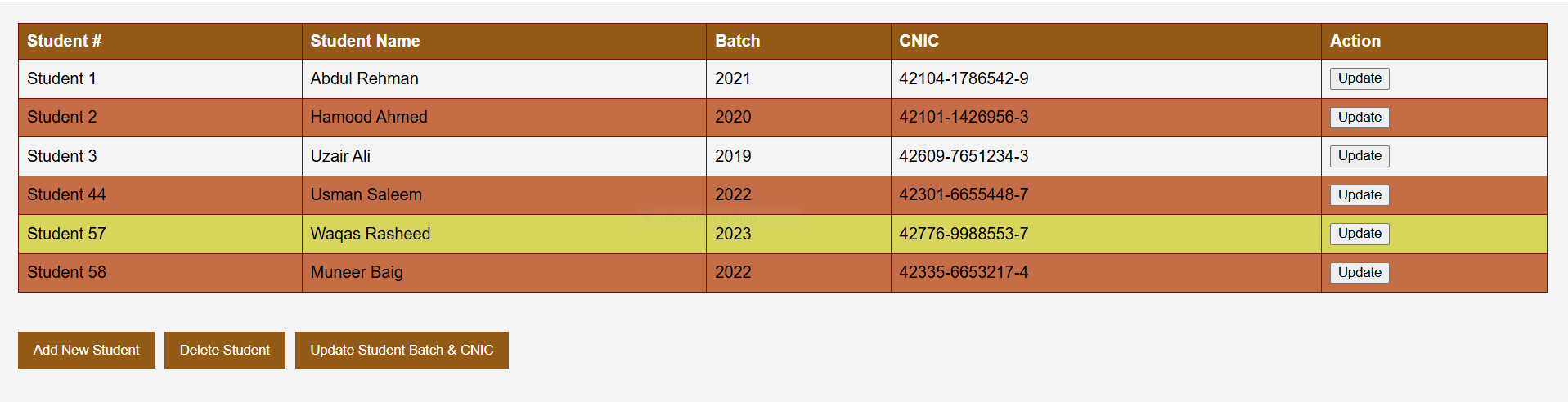
****

**To Update/Change any New Record:**

****

**For Updating/Changing Any New Record Student # 44,57 Record is Changed/Updated:**

**Web Engineering**



Name: Hamood Ahmed Lab#3 STD ID: 19694

**Task 1**

Name: Hamood Ahmed Lab#3 STD ID: 19694

**Task 2**

### Homework Tasks

* 1. Create a new *Student Profile* page
     1. Add *Title* using suitable *h1* or *h2* tag
     2. Add an introduction paragraph along with image using p and img tag srespectively
     3. Add a table listing down courses taken using table tag
        1. Add CourseId, CourseName, CreditHours, and year taken in
        2. Add at least four courses using row tag
        3. At the bottom of table,create a div tag and add following buttons
           1. Search and highlight the course row input by the user (using window.prompt)
           2. Add new Course
           3. Update Course
           4. Remove Course

### Add Interests section

* + - 1. Add link using a tag pointing to UIT university homepage.

**Code:**

**Web Engineering**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>UIT University HomePage</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            background-color: #e3885b;

            margin: 20px;

            color: #333;

        }

        h1, h2 {

            color: #7b1289;

        }

        .profile-container {

            max-width: 800px;

            margin: auto;

            padding: 20px;

            background-color: #fff;

            border-radius: 8px;

            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

        }

        .intro img {

            width: 150px;

            border-radius: 50%;

            float: right;

            margin: 10px;

        }

        table {

            width: 100%;

            border-collapse: collapse;

            margin-top: 20px;

        }

        th, td {

            border: 1px solid #1b5e51;

            padding: 10px;

            text-align: left;

        }

        th {

            background-color: #b43e83;

            color: white;

        }

        tr:nth-child(even) {

            background-color: #f9f9f9;

        }

        .button-container {

            margin-top: 15px;

            text-align: center;

        }

        button {

            padding: 10px 20px;

            margin: 5px;

            background-color: #b43e83;

            color: white;

            border: none;

            cursor: pointer;

            border-radius: 4px;

        }

        button:hover {

            background-color: #535217;

        }

        .interests {

            margin-top: 20px;

            font-size: 16px;

        }

        .interests a {

            color: #6d224e;

            text-decoration: none;

        }

        .interests a:hover {

            text-decoration: underline;

        }

    </style>

</head>

<body>

<div class="profile-container">

    <h1>Student Profile</h1>

    <div class="intro">

        <p>Hello, I am a Passionate Student about Technology and Transformation.

        I am currently continuing my degree and I am actively involved in various Projects and Courses that enhance my learning experience.</p>

        <img src="usman logo.png" alt="Usman University Logo">

    </div>

    <h2>Courses Taken</h2>

    <table id="courseTable">

        <thead>

            <tr>

                <th>Course ID</th>

                <th>Course Name</th>

                <th>Credit Hours</th>

                <th>Year Taken</th>

            </tr>

        </thead>

        <tbody>

            <tr>

                <td>CSE101</td>

                <td>Introduction to Computing</td>

                <td>3</td>

                <td>2021</td>

            </tr>

            <tr>

                <td>CSE102</td>

                <td>Object Oriented Programming</td>

                <td>4</td>

                <td>2022</td>

            </tr>

            <tr>

                <td>CSE203</td>

                <td>Data Structures & Algorithms</td>

                <td>3</td>

                <td>2023</td>

            </tr>

            <tr>

                <td>CSE304</td>

                <td>Operating Systems</td>

                <td>3</td>

                <td>2024</td>

            </tr>

        </tbody>

    </table>

    <div class="button-container">

        <button onclick="searchCourse()">Search Course</button>

        <button onclick="addCourse()">Add New Course</button>

        <button onclick="updateCourse()">Update Course</button>

        <button onclick="removeCourse()">Remove Course</button>

    </div>

    <div class="interests">

        <h2>Interests</h2>

        <p>Learn more about <a href="https://uit.edu/" target="\_blank">UIT University</a> and its exciting programs.</p>

    </div>

</div>

<script>

    function searchCourse() {

        const courseId = prompt("Enter the Course ID to search:");

        const rows = document.querySelectorAll("#courseTable tbody tr");

        let found = false;

        rows.forEach(row => {

            row.style.backgroundColor = "";

            if (row.cells[0].textContent === courseId) {

                row.style.backgroundColor = "#ffff99";

                found = true;

            }

        });

        if (!found) {

            alert("Course not found.");

        }

    }

    function addCourse() {

        const courseId = prompt("Enter Course ID:");

        const courseName = prompt("Enter Course Name:");

        const creditHours = prompt("Enter Credit Hours:");

        const yearTaken = prompt("Enter Year Taken:");

        const tableBody = document.querySelector("#courseTable tbody");

        const newRow = document.createElement("tr");

        newRow.innerHTML = `

            <td>${courseId}</td>

            <td>${courseName}</td>

            <td>${creditHours}</td>

            <td>${yearTaken}</td>

        `;

        tableBody.appendChild(newRow);

    }

    function updateCourse() {

        const courseId = prompt("Enter the Course ID to update:");

        const rows = document.querySelectorAll("#courseTable tbody tr");

        let found = false;

        rows.forEach(row => {

            if (row.cells[0].textContent === courseId) {

                const newCourseName = prompt("Enter new Course Name:");

                const newCreditHours = prompt("Enter new Credit Hours:");

                const newYearTaken = prompt("Enter new Year Taken:");

                row.cells[1].textContent = newCourseName;

                row.cells[2].textContent = newCreditHours;

                row.cells[3].textContent = newYearTaken;

                found = true;

            }

        });

        if (!found) {

            alert("Course not found.");

        }

    }

    function removeCourse() {

        const courseId = prompt("Enter the Course ID to remove:");

        const rows = document.querySelectorAll("#courseTable tbody tr");

        let found = false;

        rows.forEach(row => {

            if (row.cells[0].textContent === courseId) {

                row.remove();

                found = true;

            }

        });

        if (!found) {

            alert("Course not found.");

        }

    }

</script>

</body>

</html>

**Web Engineering**

Name: Hamood Ahmed Lab#3 STD ID: 19694

**Task 2**

**Output (UIT University HomePage):**

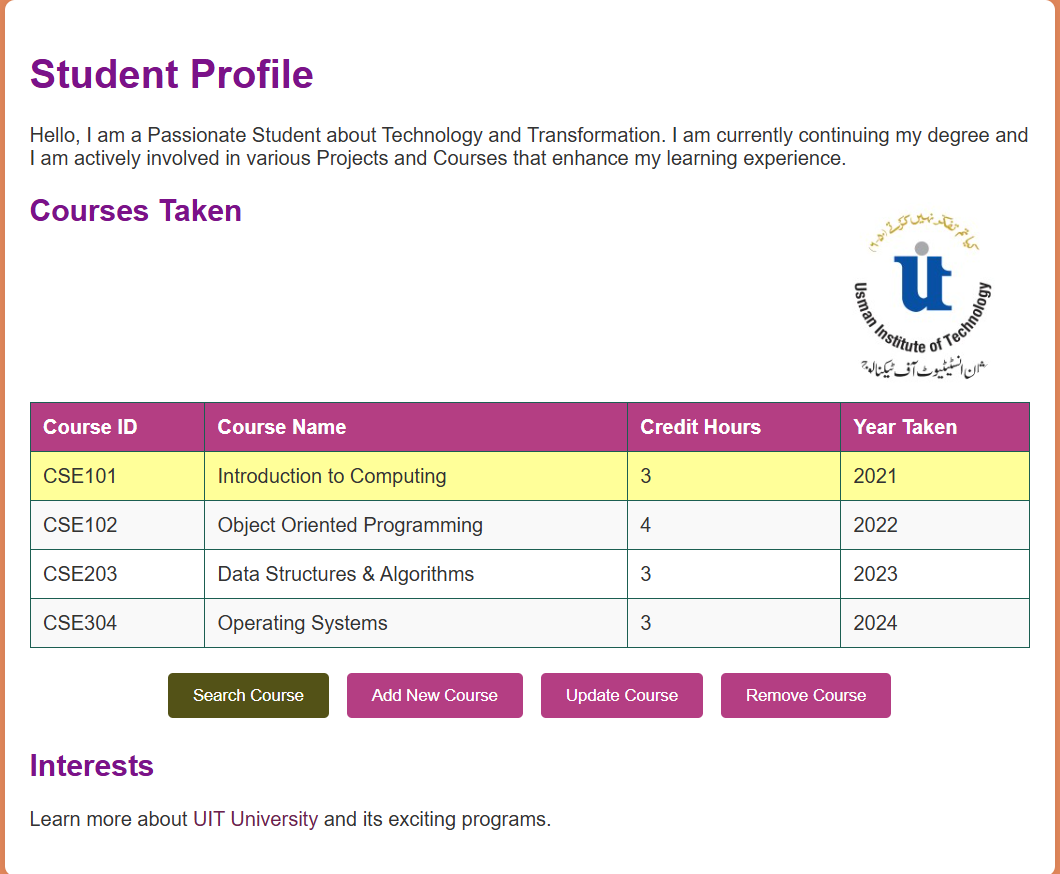
****

**Web Engineering**

Name: Hamood Ahmed Lab#3 STD ID: 19694

**Task 2**

**To Search Any Course From HomePage(CSE101 Introduction To Computing Course is Searched):**

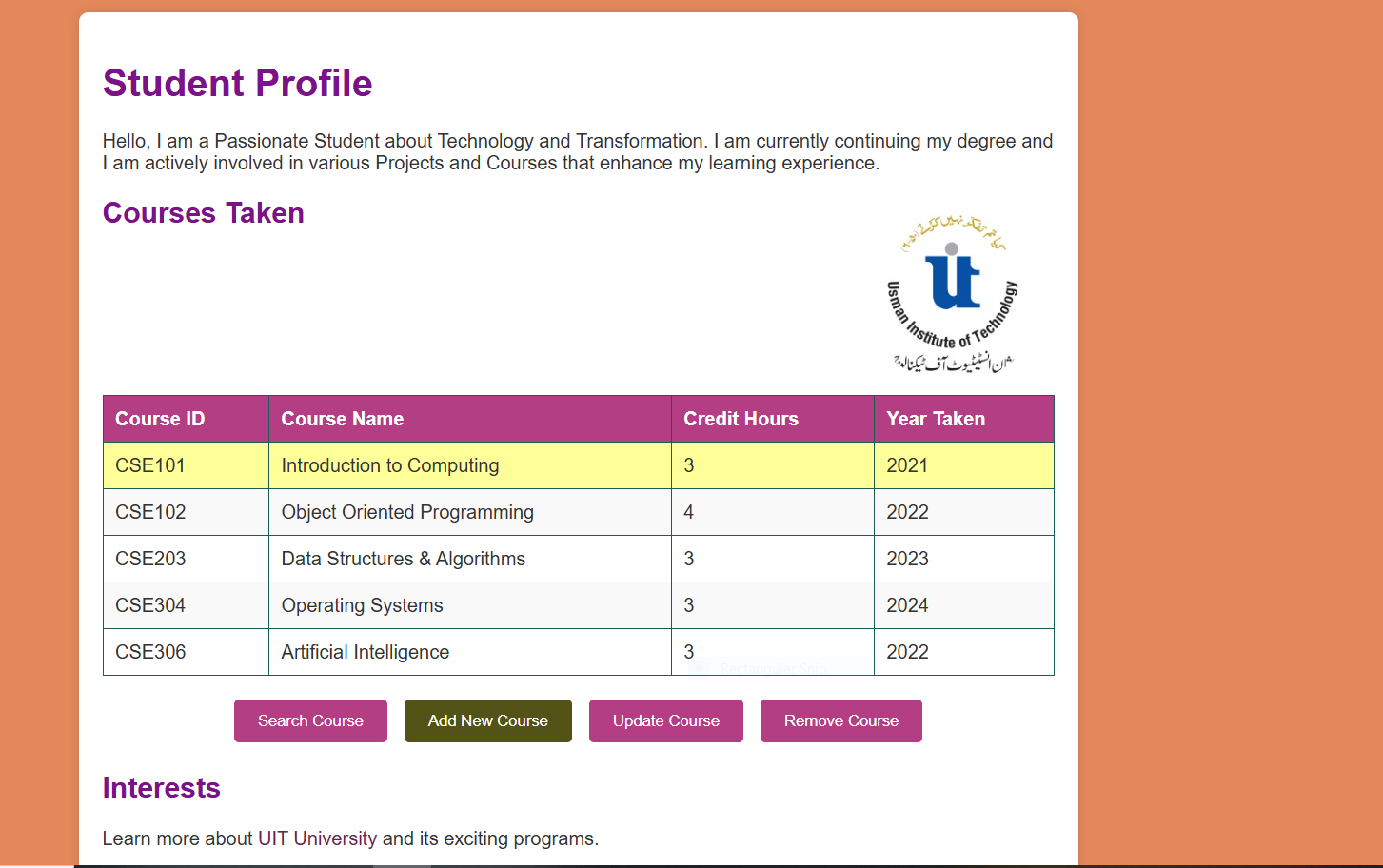
****

**Web Engineering**

**To Add New Course From HomePage(CSE306 Artificial Intelligence Course is Added):**

Name: Hamood Ahmed Lab#3 STD ID: 19694

**Task 2**

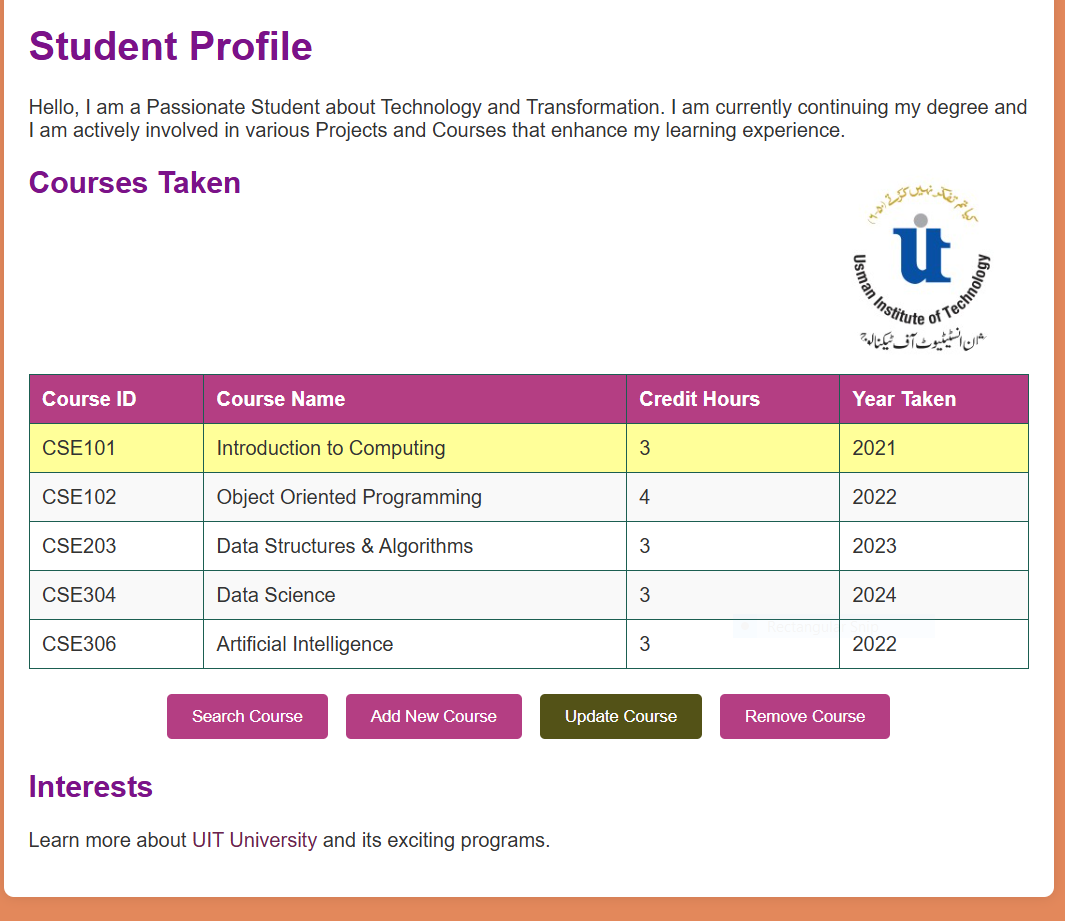
****

**Web Engineering**

**To Update Course From HomePage(CSE304 From Operating Systems to Data Science Course is Updated or Changed):**

Name: Hamood Ahmed Lab#3 STD ID: 19694

**Task 2**



**Web Engineering**

**To Remove Course From HomePage(CSE 102 Object Oriented Programming Course is Removed):**

Name: Hamood Ahmed Lab#3 STD ID: 19694

**Task 2**



**Web Engineering**

**Web Engineering (LAB) 4**

**Name: Hamood Ahmed**

**ID: 022-20-119694**

**Program: BSCS**

**Subject: Web Engineering (LAB)**

**Course Instructor: Sir Syed Atir Iftikhar**

Name: Hamood Ahmed Lab#4 STD ID: 19694

**Task 1**

Lab Task

Use jQuery to filter/search for specific elements.Perform a case-insensitive search for items in a table.Type something in the input field to search from the table:



### You can also use bootstrap CSS to show the proper styling of table using the given below link

<https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css>

**Web Engineering**

Code:

<!DOCTYPEhtml>

<htmllang="en">

<head>

  <metacharset="UTF-8">

<title>Table Search Filter</title>

  <!-- Bootstrap CSS -->

  <linkrel="stylesheet"href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">

  <!-- jQuery Library -->

  <scriptsrc="https://code.jquery.com/jquery-3.6.0.min.js"></script>

  <script>

    $(document).ready(function() {

      // Listen for changes in the search input field

      $("#searchInput").on("keyup", function() {

        // Get the search value and convert it to lowercase for case-insensitive search

        varvalue = $(this).val().toLowerCase();

        $("#dataTable tr").filter(function() {

          $(this).toggle($(this).text().toLowerCase().indexOf(value) > -1);

        });

      });

    });

  </script>

</head>

<body>

<divclass="container mt-3">

  <h2>Employee Table</h2>

  <!-- Search input field -->

  <inputid="searchInput"type="text"class="form-control mb-3"placeholder="Search">

  <!-- Data table -->

  <table class="table table-bordered table-striped">

    <thead>

      <tr>

        <th>Name</th>

        <th>Position</th>

        <th>Office</th>

        <th>Age</th>

        <th>Start date</th>

        <th>Salary</th>

      </tr>

    </thead>

    <tbody id="dataTable">

      <tr>

        <td>Tiger Nixon</td>

        <td>System Architect</td>

        <td>Edinburgh</td>

        <td>61</td>

        <td>2011/04/25</td>

        <td>$320,800</td>

      </tr>

      <tr>

        <td>Garrett Winters</td>

        <td>Accountant</td>

        <td>Tokyo</td>

        <td>63</td>

        <td>2011/07/25</td>

        <td>$170,750</td>

      </tr>

      <tr>

        <td>Ashton Cox</td>

        <td>Junior Technical Author</td>

        <td>San Francisco</td>

        <td>66</td>

        <td>2009/01/12</td>

        <td>$86,000</td>

      </tr>

      <tr>

        <td>Cedric Kelly</td>

        <td>Senior Javascript Developer</td>

        <td>Edinburgh</td>

        <td>22</td>

        <td>2012/03/29</td>

        <td>$433,060</td>

      </tr>

      <tr>

        <td>Airi Satou</td>

        <td>Accountant</td>

        <td>Tokyo</td>

        <td>33</td>

        <td>2008/11/28</td>

        <td>$162,700</td>

      </tr>

      <tr>

        <td>Brielle Williamson</td>

        <td>Integration Specialist</td>

        <td>New York</td>

        <td>61</td>

        <td>2012/12/02</td>

        <td>$372,000</td>

      </tr>

      <tr>

        <td>Herrod Chandler</td>

        <td>Sales Assistant</td>

        <td>San Francisco</td>

        <td>59</td>

        <td>2012/08/06</td>

        <td>$137,500</td>

      </tr>

    </tbody>

  </table>

</div>

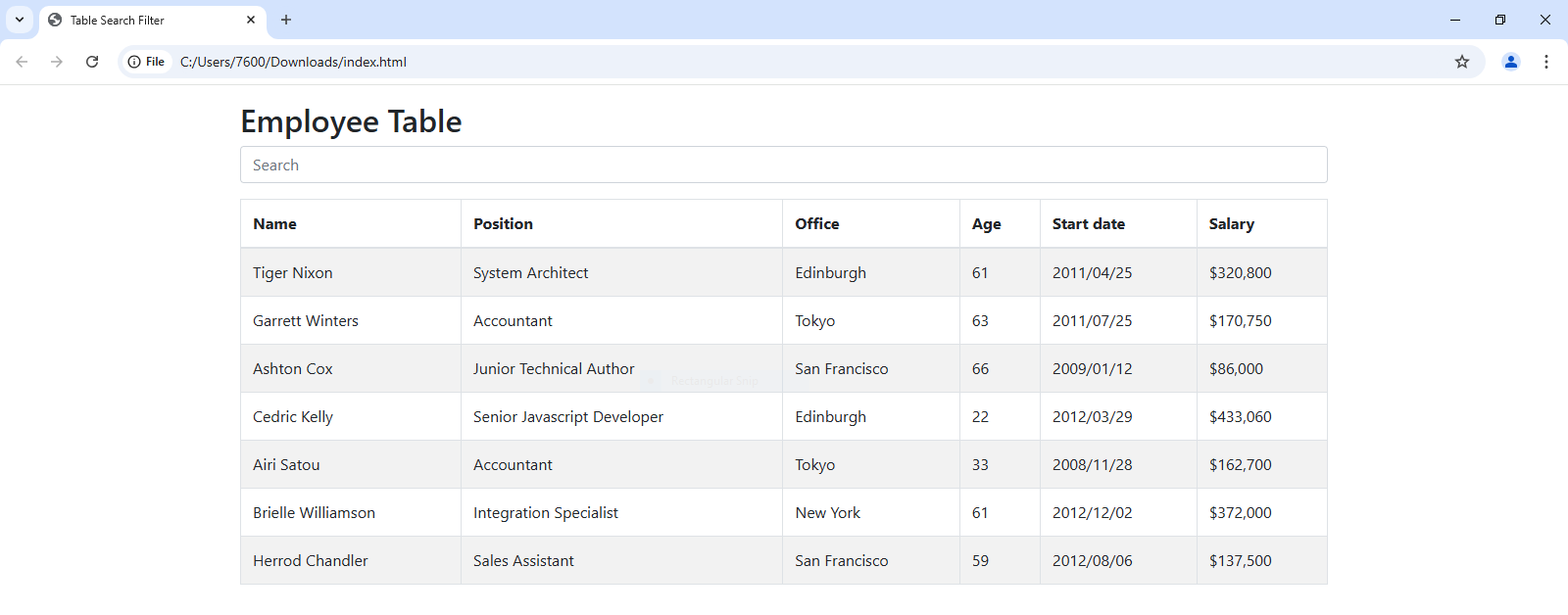
</body>

</html>

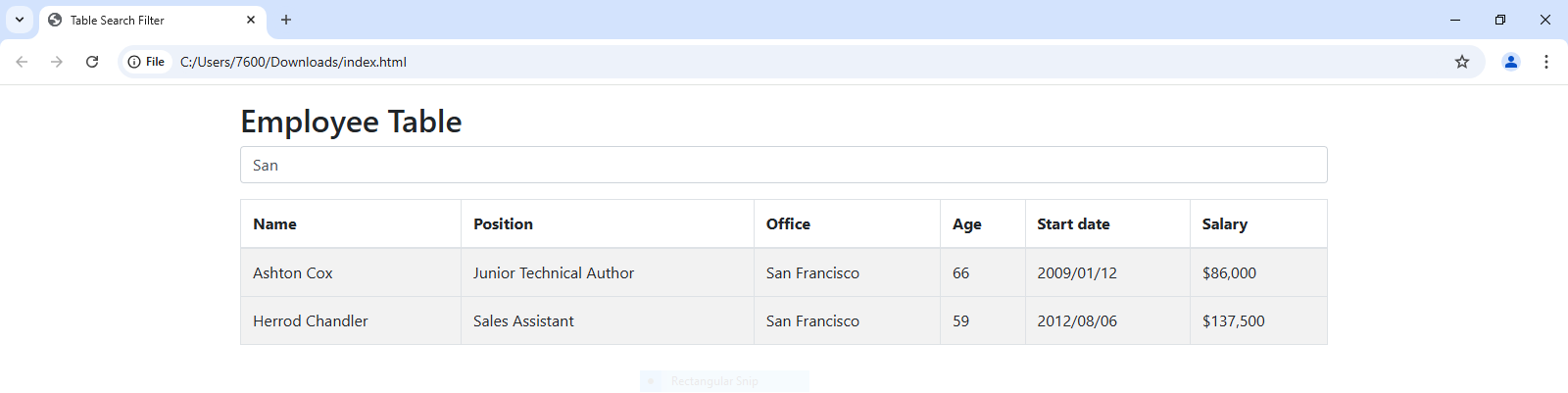
Name: Hamood Ahmed Lab#4 STD ID: 19694

**Task 1**

Output (Employee Table):

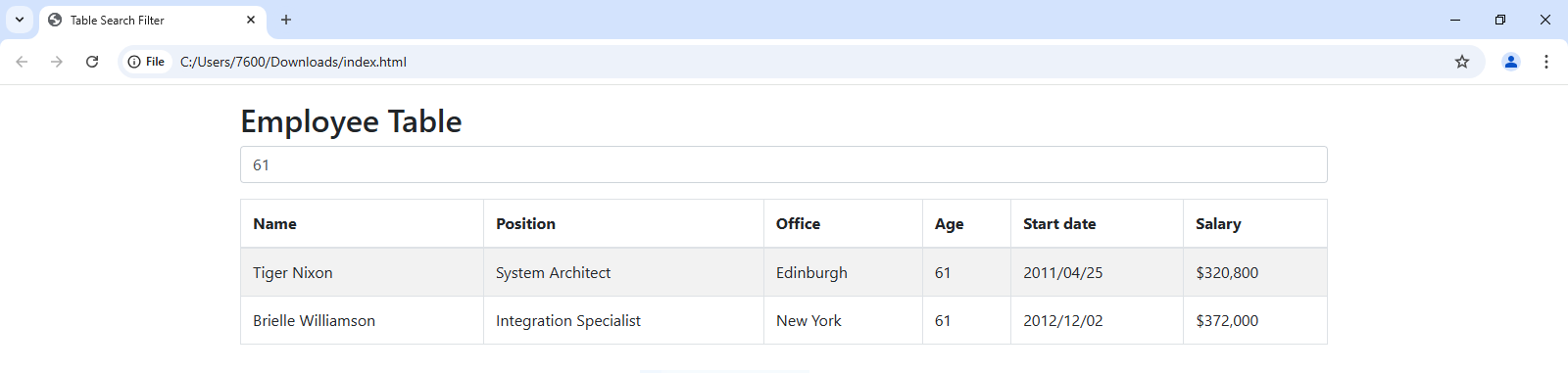


(Table Searched With Employee Office Address):



(Table Searched With Employee Age):

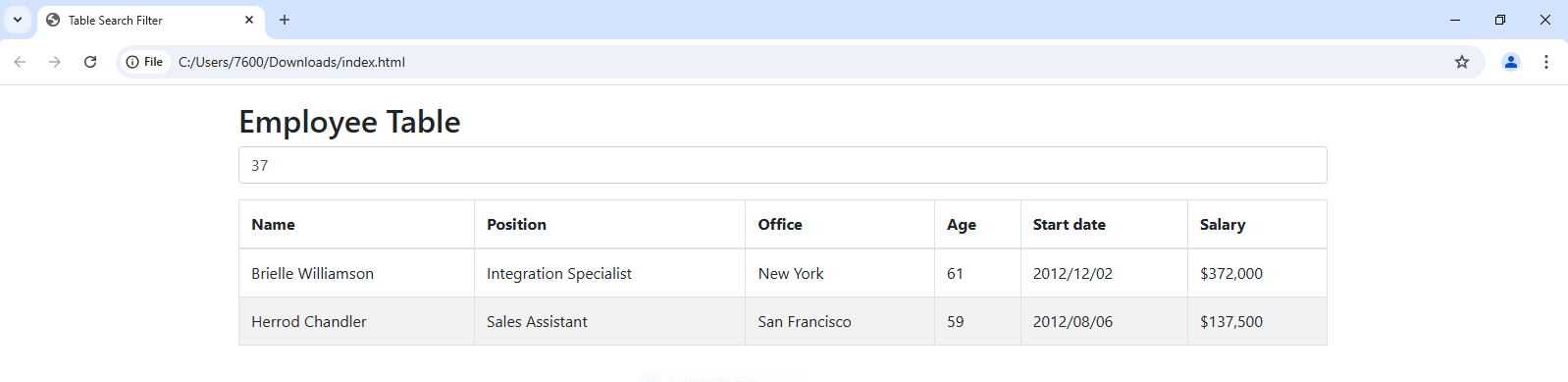
**Web Engineering**



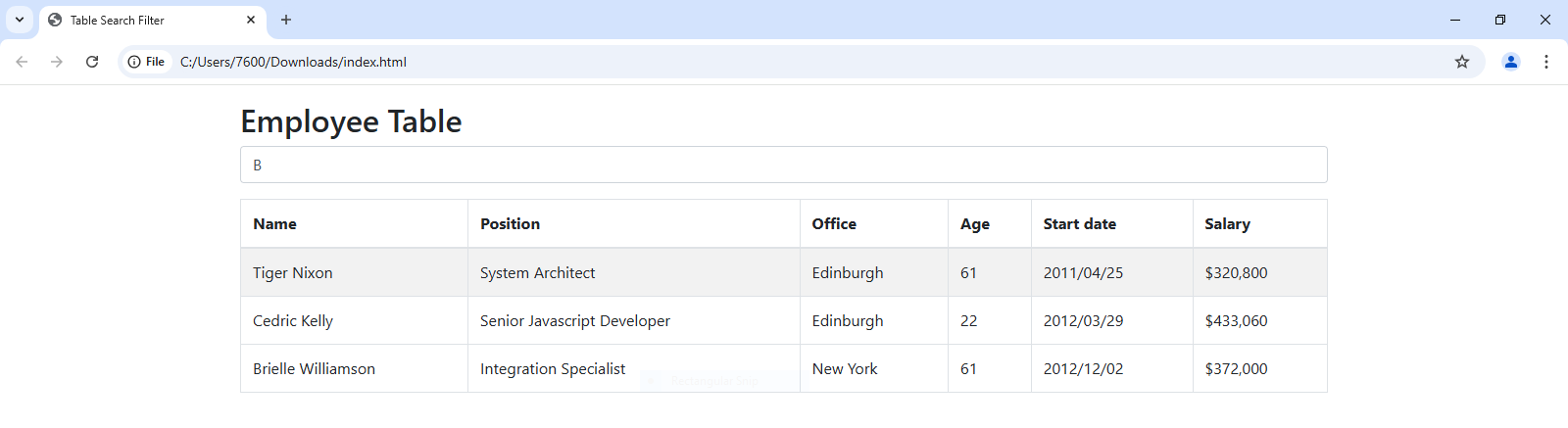
(Table Searched With Employee Salary):

Name: Hamood Ahmed Lab#4 STD ID: 19694

**Task 1**

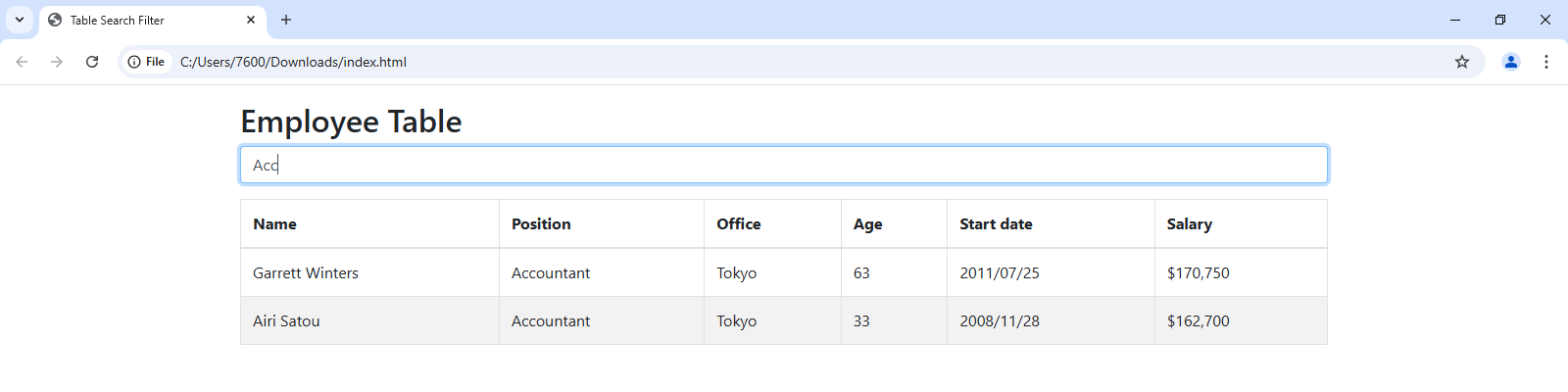


(Table Searched With Employee Name):



(Table Searched With Employee Position):

**Web Engineering**



Name: Hamood Ahmed Lab#4 STD ID: 19694

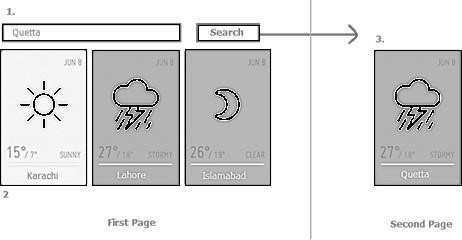
**Task 2**

**Home Task:**

## Replica of Minimeteo

### **Don’t** Use any JavaScript popup boxes.Use something else to show error!

Design Guide & App Information Link: <https://goo.gl/espVRo>

Create/Design a 2pages,minimal design,and weather information responsive application for Pakistan’s major cities.

## Process Flow.

### Onload/start,

**Web Engineering**

Name: Hamood Ahmed Lab#4 STD ID: 19694

**Task 2**

1. An Ajax request is created & send to any open/free weather API & return response of **3 major cities of Pakistan.**

### On successful response, populate/set response data in weather card as shown in **[2]** on1st/Main page.

1. Also store response in local storage for future processing.

### On error response[case like internet isn’t working etc.],just show error & reuse last local storage values if exists.

* On search click**[1]**,

### An Ajax request is created & send to any open/free weather API with provided city name & return response of **particular city**.

1. On successful response, populate/set response data in weather card as shown in **[3]** on2ndpage.

## Follow above point c & d

### Weather card is consist of,

1. City Name

### Temperature

1. Date

### Weather Icon

Weather status

**Code:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Colorful Weather App</title>

    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">

    <style>

        body {

            font-family: Arial, sans-serif;

            background: linear-gradient(to right, #c9303c, #e25396);

            color: #f3f3ea;

        }

        .weather-card {

            border: 1px solid #ccc;

            padding: 15px;

            margin: 10px;

            width: 150px;

            text-align: center;

            border-radius: 8px;

            color: white;

            transition: transform 0.3s;

        }

        .weather-card:hover {

            transform: scale(1.05);

        }

        .sunny { background-color: #be7026; }

        .cloudy { background-color: #204d88; }

        .rain { background-color: #691669; }

        .stormy { background-color: #0d5751; }

        .clear { background-color: #0d583c; }

        #error-message { color: rgb(145, 66, 52); display: none; text-align: center; }

        #secondPage { display: none; }

        .container { max-width: 500px; }

        .btn-primary { width: 100%; }

        h3 { color: white; }

    </style>

</head>

<body>

    <div class="container mt-3">

        <div class="input-group mb-3">

            <input id="searchInput" type="text" class="form-control" placeholder="Enter city name (e.g., Quetta)">

            <div class="input-group-append">

                <button id="searchButton" class="btn btn-primary">Search</button>

            </div>

        </div>

        <div id="error-message">Error: Unable to fetch data. Showing cached data if available.</div>

        <!-- Main Weather Cards for Major Cities -->

        <div id="mainPage">

            <h3 class="text-center">Weather in Major Cities</h3>

            <div class="d-flex justify-content-center flex-wrap" id="weatherCards"></div>

        </div>

        <!-- Second Page for Search Results -->

        <div id="secondPage">

            <h3 class="text-center">Search Result</h3>

            <div id="searchResult" class="d-flex justify-content-center flex-wrap"></div>

            <button class="btn btn-secondary mt-3" onclick="goBack()">Back to Main Page</button>

        </div>

    </div>

    <!-- jQuery Library -->

    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

    <script>

        $(document).ready(function() {

            const cities = ["Karachi", "Lahore", "Islamabad", "Quetta"];

            const mockWeatherData = {

                "Karachi": { name: "Karachi", main: { temp: 30 }, weather: [{ main: "Sunny", icon: "01d" }] },

                "Lahore": { name: "Lahore", main: { temp: 25 }, weather: [{ main: "Cloudy", icon: "02d" }] },

                "Islamabad": { name: "Islamabad", main: { temp: 22 }, weather: [{ main: "Rain", icon: "09d" }] },

                "Quetta": {name: "Quetta", main: { temp: 27 }, weather: [{ main: "Stormy", icon: "04d" }] }

            };

            const pakistanCities = ["karachi", "lahore", "islamabad", "quetta"];

            function getMockWeather(city) {

                return mockWeatherData[city] || { name: city, main: { temp: 20 }, weather: [{ main: "Clear", icon: "01n" }] };

            }

            function getWeatherClass(weather) {

                switch(weather.toLowerCase()) {

                    case "sunny": return "sunny";

                    case "cloudy": return "cloudy";

                    case "rain": return "rain";

                    case "stormy": return "stormy";

                    default: return "clear";

                }

            }

            function displayWeather(data, container) {

                const weatherClass = getWeatherClass(data.weather[0].main);

                const weatherCard = `

                    <div class="weather-card ${weatherClass}">

                        <h5>${data.name}</h5>

                        <p>${data.main.temp}°C</p>

                        <p>${new Date().toLocaleDateString()}</p>

                        <img src="http://openweathermap.org/img/w/${data.weather[0].icon}.png" alt="${data.weather[0].main}">

                        <p>${data.weather[0].main}</p>

                    </div>

                `;

                container.append(weatherCard);

            }

            function loadMainPage() {

                $("#weatherCards").empty();

                $("#error-message").hide();

                cities.forEach(city => {

                    const data = getMockWeather(city);

                    displayWeather(data, $("#weatherCards"));

                });

            }

            $("#searchButton").click(function() {

                const city = $("#searchInput").val().trim();

                if (!city) {

                    $("#error-message").text("Please enter a city name.").show();

                    return;

                }

                $("#mainPage").hide();

                $("#secondPage").show();

                $("#searchResult").empty();

                $("#error-message").hide();

                const lowerCaseCity = city.toLowerCase();

                if (pakistanCities.includes(lowerCaseCity)) {

                    cities.forEach(city => {

                        const data = getMockWeather(city);

                        displayWeather(data, $("#searchResult"));

                    });

                } else {

                    const data = getMockWeather(city);

                    displayWeather(data, $("#searchResult"));

                }

            });

            window.goBack = function() {

                $("#secondPage").hide();

                $("#mainPage").show();

                $("#error-message").hide();

            }

            loadMainPage();

        });

    </script>

</body>

</html>

Name: Hamood Ahmed Lab#4 STD ID: 19694

**Task 2**

**Output (Weather App):**



**Other City (For Example Multan):**

**Web Engineering**

Name: Hamood Ahmed Lab#4 STD ID: 19694

**Task 2**



**Web Engineering**

**Web Engineering (LAB) 5**

**Name: Hamood Ahmed**

**ID: 022-20-119694**

**Program: BSCS**

**Subject: Web Engineering (LAB)**

**Course Instructor: Sir Syed Atir Iftikhar**

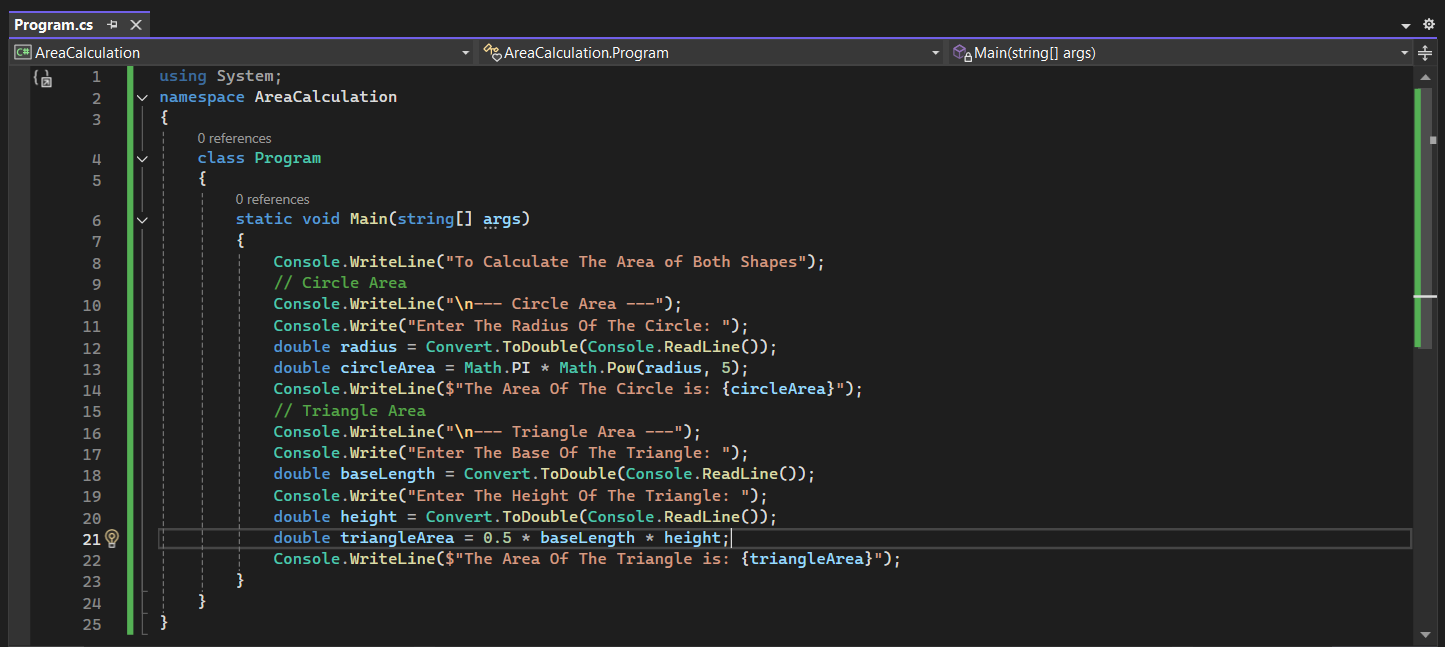
### Lab Task

1. Write a C# console program that reads input & calculates the area of circle & Triangle?

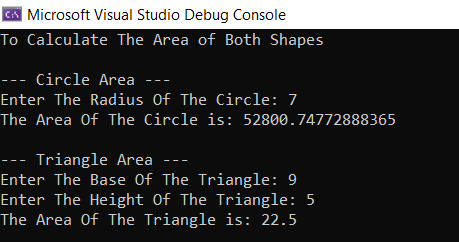
Name: Hamood Ahmed Lab#5 STD ID: 19694

**Task 1**

**Code:**

****

**Output:**

****

**Web Engineering**

Name: Hamood Ahmed Lab#5 STD ID: 19694

**Task 2**

1. Create a calculator which will ask user to input 2 number and an operation (Add,Subtract, Divide and multiply) that user want to perform.

**Code:**

using System;

namespace Calculator

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine(" Simple Calculator Performing Several Operations ");

Console.Write("Enter the First Number: ");

double number1 = Convert.ToDouble(Console.ReadLine());

Console.Write("Enter the Second Number: ");

double number2 = Convert.ToDouble(Console.ReadLine());

double result = 0;

bool validOperation = false;

// While Loop

while (!validOperation)

{

Console.Write("Enter the operations (Add, Subtract, Multiply, Divide): ");

string operation = Console.ReadLine().ToLower();

// Calculation based on the operation

if (operation == "add")

{

result = number1 + number2;

validOperation = true;

}

else if (operation == "subtract")

{

result = number1 - number2;

validOperation = true;

}

else if (operation == "multiply")

{

result = number1 \* number2;

validOperation = true;

}

else if (operation == "divide")

{

if (number2 != 0)

{

result = number1 / number2;

validOperation = true;

}

else

{

Console.WriteLine("Error: Division by zero is not allowed.");

}

}

else

{

Console.WriteLine("Invalid operation. Please enter Add, Subtract, Multiply, or Divide.");

}

}

// Display the result

Console.WriteLine($"The result of the operation is: {result}");

}

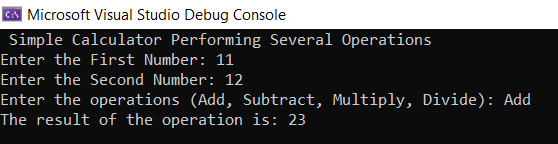
}

}

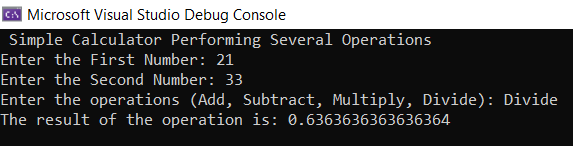
**Web Engineering**

**Output**

**Snap 1(Add Operation Performed):**

****

**Snap 2(Divide Operation Performed Similarly Other Operations are performed in this way):**

****

1. Take two numbers from user and find out all even and odd numbers between then using while loop?

Name: Hamood Ahmed Lab#5 STD ID: 19694

**Task 3**

**Code:**

using System;

namespace EvenOddNumbers

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Even and Odd Numbers Between The Two Numbers");

Console.Write("Enter the First Number: ");

int start = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter the Second Number: ");

int end = Convert.ToInt32(Console.ReadLine());

if (start > end)

{

int temp = start;

start = end;

end = temp;

}

Console.WriteLine("\nThe Even number is:", start, end);

int current = start;

// Even Numbers

while (current <= end)

{

if (current % 2 == 0)

{

Console.Write(current + " ");

}

current++;

}

Console.WriteLine("\nThe Odd Number is:", start, end);

current = start;

// Odd Numbers

while (current <= end)

{

if (current % 2 != 0)

{

Console.Write(current + " ");

}

current++;

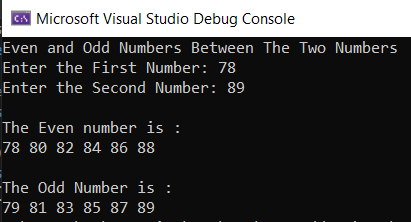
}

}

}

}

**Output:**



**Web Engineering**

1. Write a program to print a string in reverse order.For your convenience the output is given below(Hint: Use string length property)

Output:

Input A string: Welcome

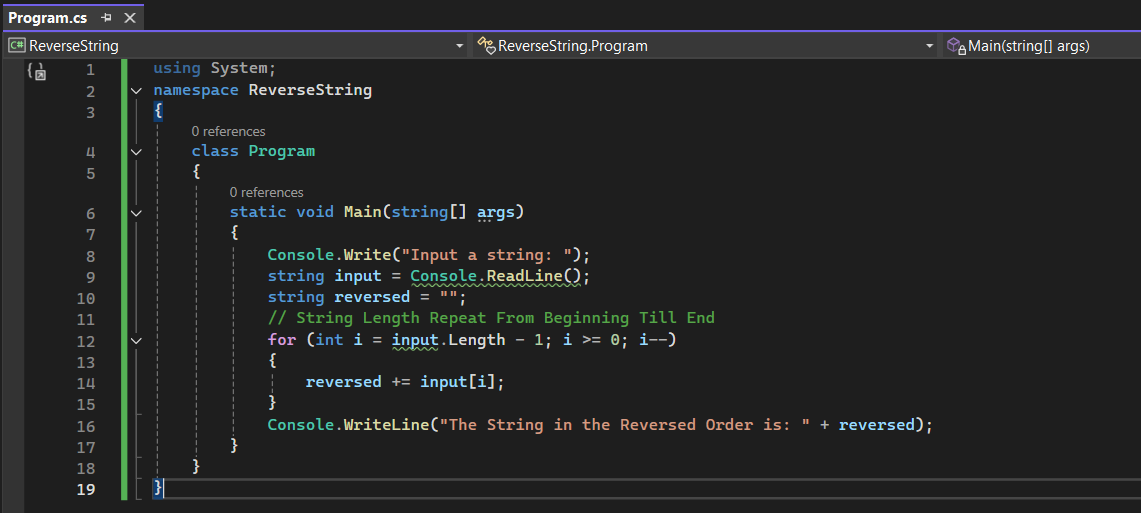
The string in the reversed string is : emocleW

Name: Hamood Ahmed Lab#5 STD ID: 19694

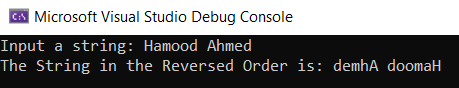
**Task 4**

**Code:**

**Web Engineering**

****

**Output:**

****

1. Home Task
2. Write a program which will take input from user and search it in an array if input number is present in an array then prints “Found” otherwise “Not Found”?

Name: Hamood Ahmed Lab#5 STD ID: 19694

**Task 1**

**Code:**

using System;

namespace ArraySearch

{

class Program

{

static void Main(string[] args)

{

// Searching of The Array

int[] numbers = { 10, 15, 23, 25, 26, 27, 28, 33, 57, 69, 99, 103 };

// Input From The User

Console.Write("Enter a number to search: ");

int inputNumber = Convert.ToInt32(Console.ReadLine());

bool isFound = false;

// Searching Of The Input Number From The Array

foreach (int number in numbers)

{

if (number == inputNumber)

{

isFound = true;

break;

}

}

// To Print the Result

if (isFound)

{

Console.WriteLine("Found The Number");

}

else

{

Console.WriteLine("Not Found Number Not Available");

}

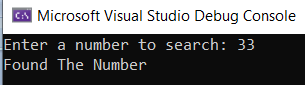
}

}

}

**Web Engineering**

**Output:**

****

1. Write a Program to make the pattern using C# on output screen1 2 3 4 5

2 1 2 3 4

3 2 1 2 3

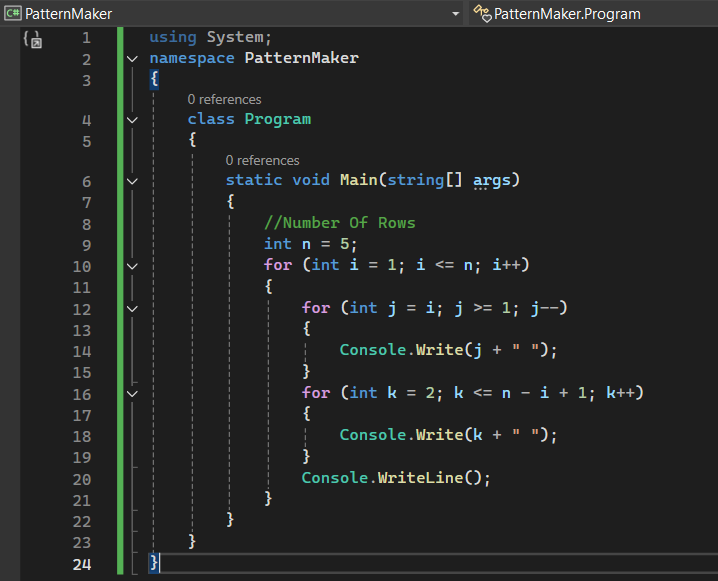
4 3 2 1 2

5 4 3 2 1

Name: Hamood Ahmed Lab#5 STD ID: 19694

**Task 2**

**Code:**

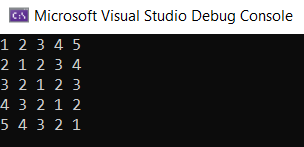
****

**Web Engineering**

**Output:**

Name: Hamood Ahmed Lab#5 STD ID: 19694

**Task 2**

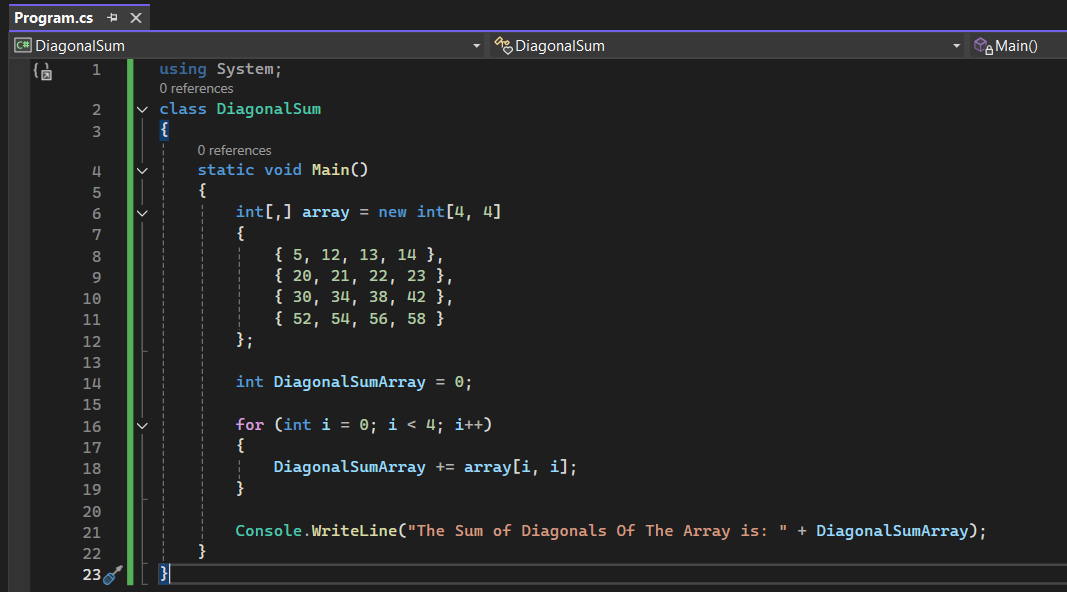
****

1. Write a program to make 2d array of 4x4 and also calculate the sum of diagonals of that array.

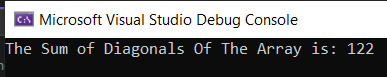
Name: Hamood Ahmed Lab#5 STD ID: 19694

**Task 3**

**Code:**

****

**Output:**

****

1. Write a program in which you should create two classes (Circle & Triangle), in both classes an AreaCalculation function must be created.Next,in the main program create the constructor of both classes and use area function for the calculation of both shapes.

Name: Hamood Ahmed Lab#5 STD ID: 19694

**Task 4**

**Web Engineering**

**Code:**

using System;

namespace ShapeCalculation

{

// Circle Class

class Circle

{

private double radius;

// Constructor To Set Radius

public Circle(double radius)

{

this.radius = radius;

}

// Method To Calculate Area Of The Circle

public double AreaCalculation()

{

return Math.PI \* Math.Pow(radius, 2);

}

}

// Triangle Class

class Triangle

{

private double baseLength;

private double height;

// Constructor To Set Base and Height

public Triangle(double baseLength, double height)

{

this.baseLength = baseLength;

this.height = height;

}

// Method To Calculate Area Of The Triangle

public double AreaCalculation()

{

return 0.5 \* baseLength \* height;

}

}

class Program

{

static void Main(string[] args)

{

// Create an Instance of Circle with a Specified Radius

Console.Write("Enter The Radius Of The Circle: ");

double radius = Convert.ToDouble(Console.ReadLine());

Circle circle = new Circle(radius);

double circleArea = circle.AreaCalculation();

Console.WriteLine("The Area Of Circle is: " + circleArea);

// Create an instance of Triangle with a Specified Base and Height

Console.Write("Enter The Base Of The Triangle: ");

double baseLength = Convert.ToDouble(Console.ReadLine());

Console.Write("Enter The Height Of The Triangle: ");

double height = Convert.ToDouble(Console.ReadLine());

Triangle triangle = new Triangle(baseLength, height);

double triangleArea = triangle.AreaCalculation();

Console.WriteLine("The Area of Triangle is: " + triangleArea);

}

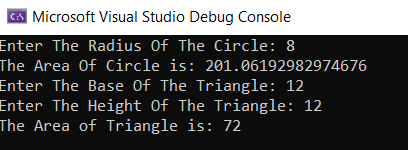
}

}

Name: Hamood Ahmed Lab#5 STD ID: 19694

**Task 4**

**Output:**

****

**Web Engineering**

**Web Engineering (LAB) 6**

**Name: Hamood Ahmed**

**ID: 022-20-119694**

**Program: BSCS**

**Subject: Web Engineering (LAB)**

**Course Instructor: Sir Syed Atir Iftikhar**

**Lab Task**

Consider the case study; One of an institution wants to develop an Attendance Management System, which offer different short course to the students. In previous one to two years, the student growth is tremendously increased. Currently they are using the manual system/excel based sheets for marking the attendance of their students. Now they are facing a lot issues in maintaining the records and reporting to the center manager is at risk from the stuff as they unable to handle the file based system. They wants to hire you as a web developer to design, develop and successfully implement the system in their esteemed organization. Keep in mind that in every software development cycle, the requirement gathering is one of the most important step. Assume that the Centre Manager has done a meeting with you, he/she has given you the requirement, and you have already created the requirement document. As you have gathered the requirement, before designing the front end and back end development a necessary part of database designing occurs. Now, consider the given below requirement points; build a normalized database and make a database on SQL Server using SSMS. Create a script of database, which you have created in SSMS, create the ERD from Database Diagrams folder and submit it. Centre Manager states that:

1. They offer different courses in the institute (namely, Interactive Design and Multimedia, Slick Dot Net Development, Digital Media & Entrepreneurship)
2. Different teachers are hired by the institute to teach different courses in fixed time slots {Fixed Time means Like 2.5 hrs class twice a week from 7:00pm-9:00pm on Friday and Saturday or 5hrs class in a week from 10am-3:00pm}.

3- A class can be comprise of many students (you can limit it to 50 students per class/time slot).

1. Once a student registered in our portal, then whenever he/she wants to take another course from our institute.

He/She do not have to register again; He/She will pay the fee and are enrolled in the course as per suitable time slot.

5- Now the major change they wants to do in marking of a student attendance is that; teacher itself will mark it.

6- Teacher will login to the system, select the course of ongoing class, put the date of day and marked the students present one by one or mark absent.

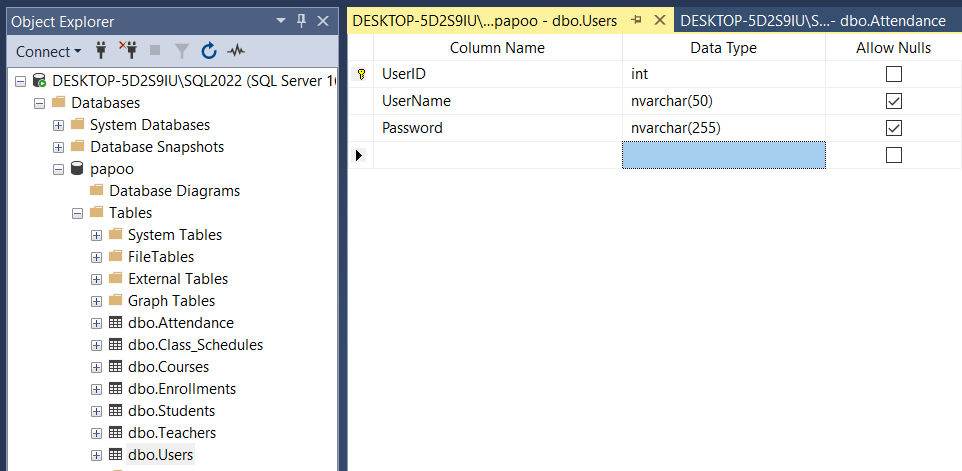
7- There will be many formats of the report, which will be provided later, after the completion of entry forms.

Name: Hamood Ahmed Lab#6 STD ID: 19694

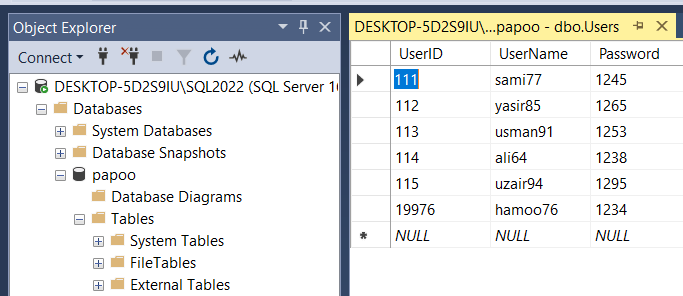
**Task**

**Attendance Management System A System is made with 7 Tables.**

**Table 1: “Users” for User Login With Its UserID,UserName and Password**

****

**Table 1: “Users” Data.**

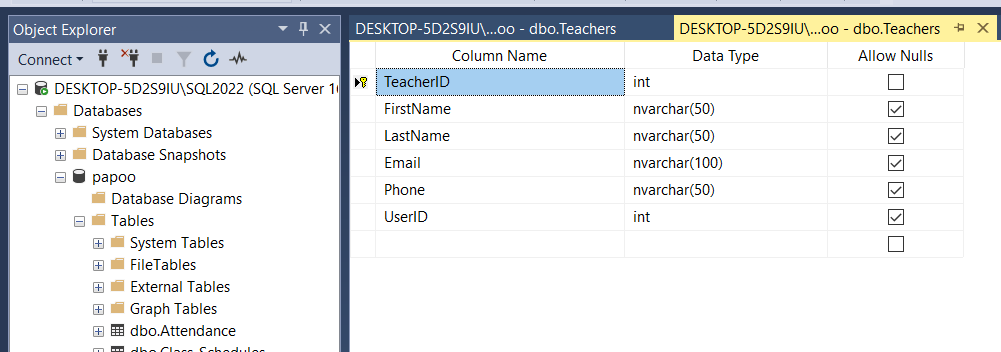
****

**Web Engineering**

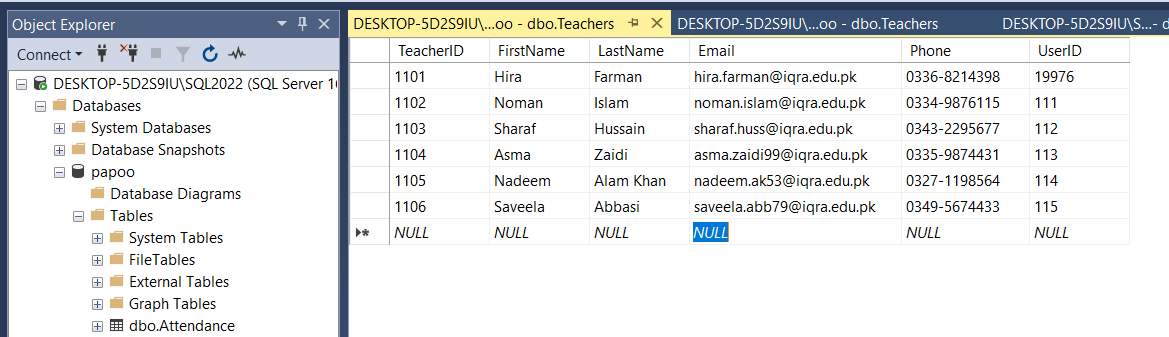
Name: Hamood Ahmed Lab#6 STD ID: 19694

**Task**

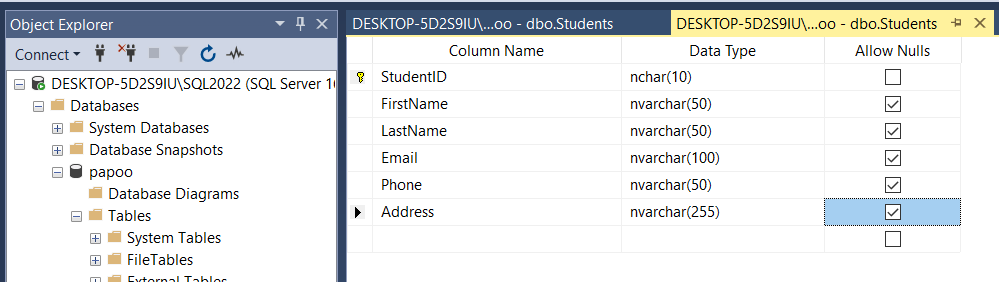
**Table 2: “Teachers” for Teachers and Their Details are given.**

****

**Table 2: “Teachers” Data.**

****

**Table 3: “Students” for Students and Their Details are mentioned.**

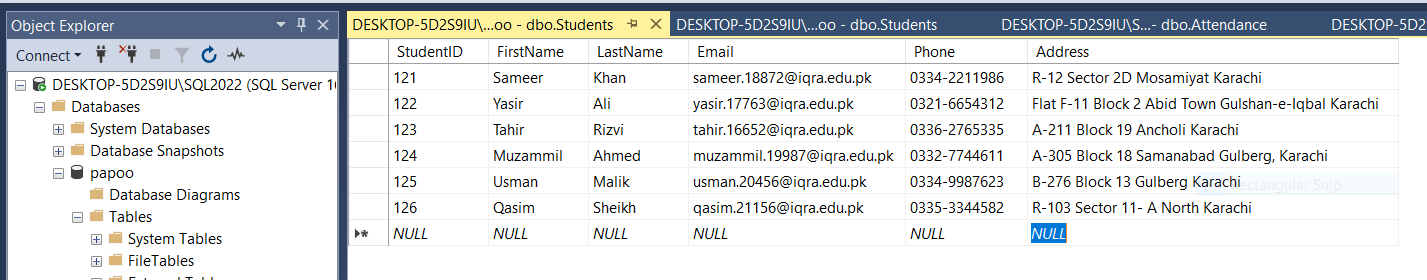
****

**Web Engineering**

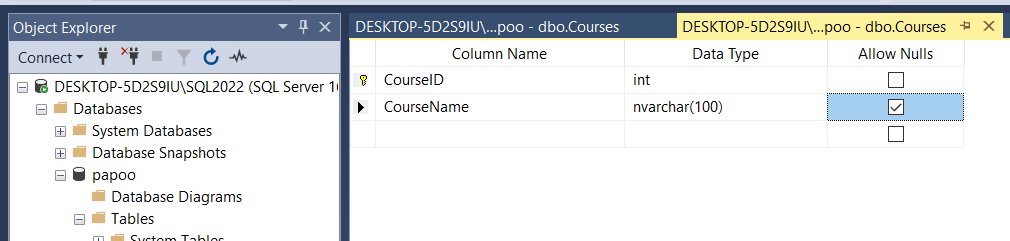
Name: Hamood Ahmed Lab#6 STD ID: 19694

**Task**

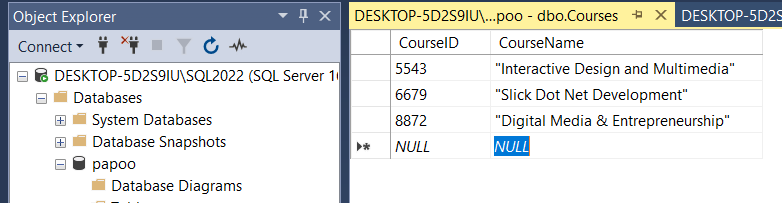
**Table 3: “Students” Data**

****

**Table 4: “Courses” for Courses Details.**

****

**Table 4: “Courses” Data.**

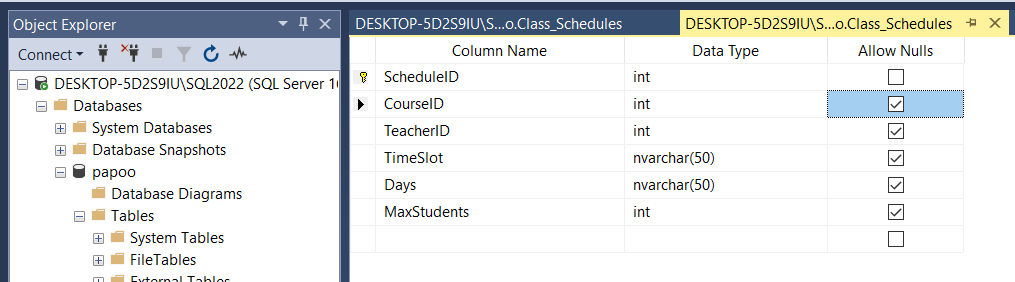
****

**Web Engineering**

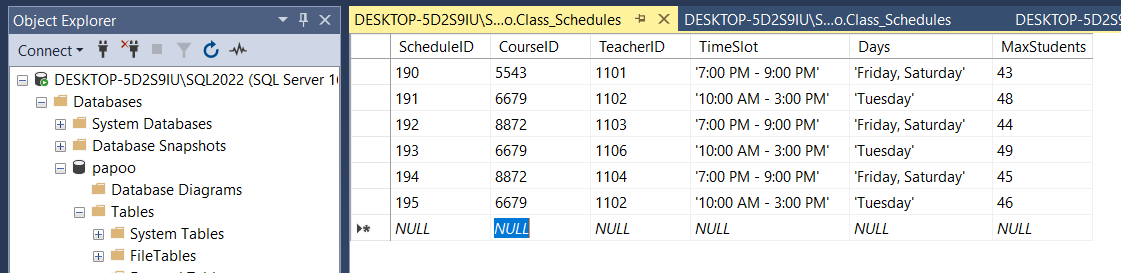
Name: Hamood Ahmed Lab#6 STD ID: 19694

**Task**

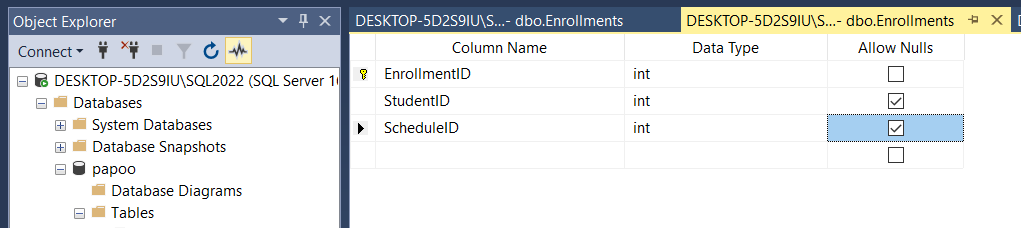
**Table 5: “Class\_Schedules” for Classes,Timings and Teacher Details.**

****

**Table 5: “Class\_Schedules” Data.**

****

**Table 6: “Enrollments” for Student Enrollment.**

****

**Web Engineering**

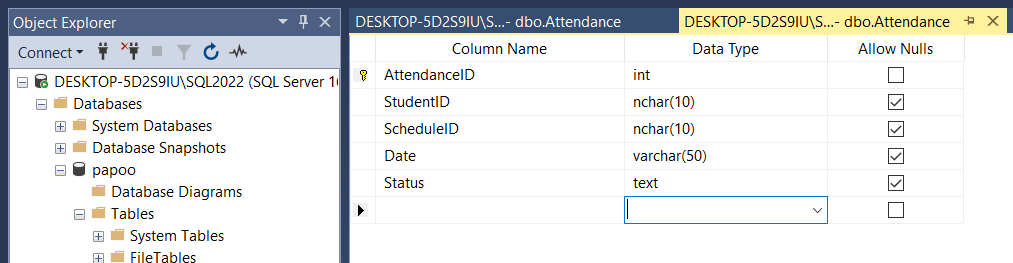
Name: Hamood Ahmed Lab#6 STD ID: 19694

**Task**

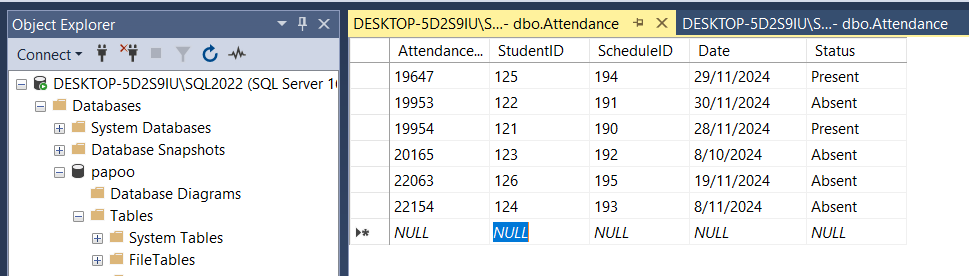
**Table 6: “Enrollments” Data.**

****

**Table 7: “Attendance” for Student Weekly Attendance.**

****

**Table 7: “Attendance” Data.**

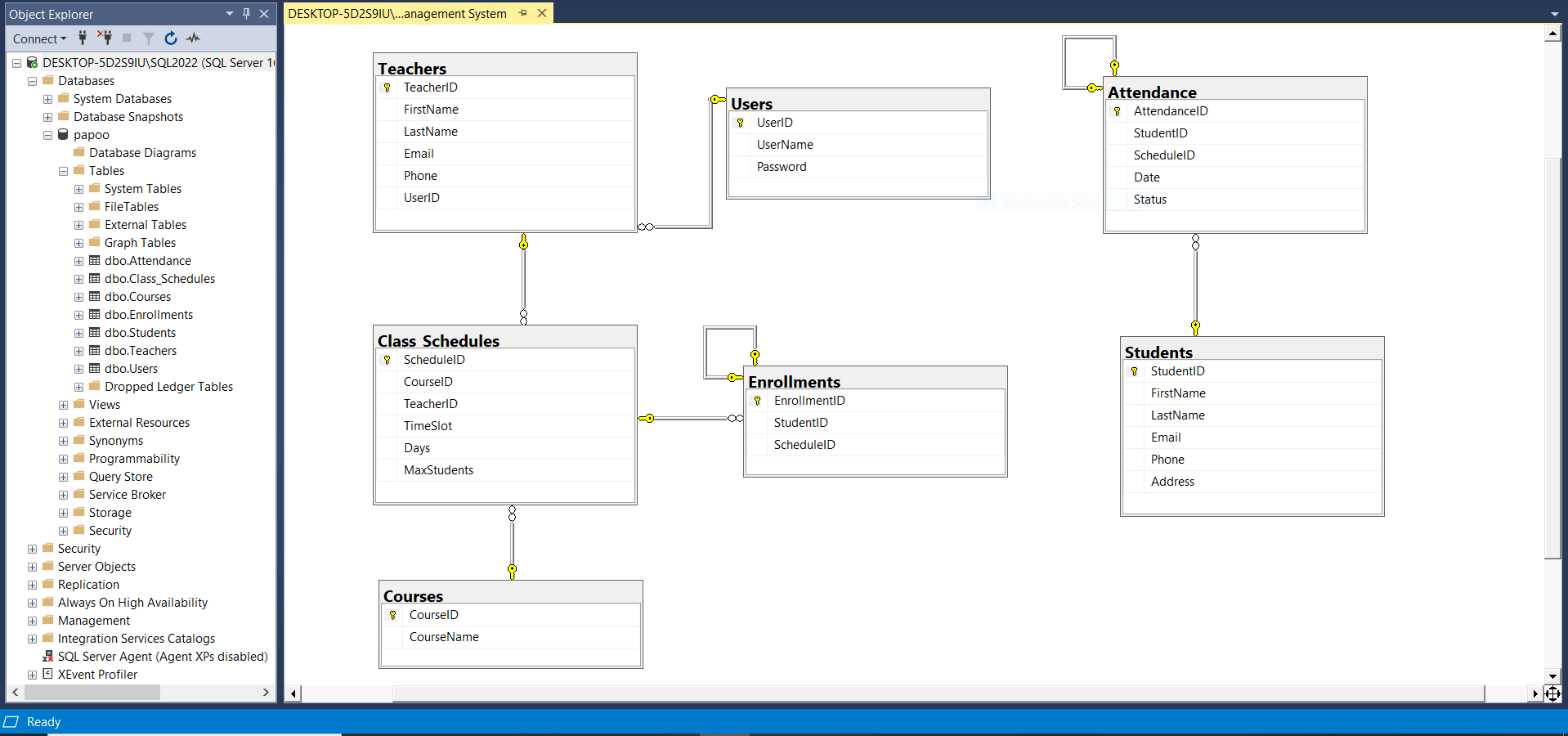
****

**Web Engineering**

Name: Hamood Ahmed Lab#6 STD ID: 19694

**Task**

**ERD Diagram of “Attendance Management System”**

****

**ERD Diagram Explanation:**

From the Above 7 Tables and Its Data-Base that is created and from the ERD Diagram it is shown that the Attendance Management System is working in this way with many relationships of **One-to-One and One-to-Many** are made with Several **Foreign Key(FK’s) and Primary Keys** are attached.

* **Users → Teachers**: One-to-one relationship.
* **Courses → Class\_Schedules**: One-to-many relationship.
* **Teachers → Class\_Schedules**: One-to-many relationship.
* **Students → Enrollments**: One-to-many relationship.
* **Class\_Schedules → Enrollments**: One-to-many relationship.

Name: Hamood Ahmed Lab#6 STD ID: 19694

**Task**

* **Students → Attendance**: One-to-many relationship.
* **Class\_Schedules → Attendance**: One-to-many relationship.

1. **Users Table**: UserID (PK -> Primary Key, int, not null).
2. **Teachers Table:** TeacherID (PK, int, not null) and UserID (FK -> Foreign Key, int, null).
3. **Students Table:** StudentID (PK, nchar(10), not null).
4. **Enrollments Table:** EnrollmentID (PK, FK, int, not null) and ScheduleID (FK, int, null).
5. **Courses Table:** CourseID (PK, int, not null).
6. **Class\_Schedules Table:** ScheduleID (PK, int, not null), CourseID (FK, int, null) and TeacherID (FK, int, null).
7. **Attendance Table:** AttendanceID (PK,FK, int, not null) and StudentID (FK, nchar(10), null).

**Web Engineering**

**Web Engineering (LAB) 7**

**Name: Hamood Ahmed**

**ID: 022-20-119694**

**Program: BSCS**

**Subject: Web Engineering (LAB)**

**Course Instructor: Sir Syed Atir Iftikhar**

11. Lab Task/Home Task

1. Do the changes in the App.js component and implement the routing.

Name: Hamood Ahmed Lab#7 STD ID: 19694

**Task 1**

**App.js File**

**Code:**

**// App.js**

import React from 'react';

import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';

// Components Are Imported

import Home from './components/Home';

import About from './components/About';

import Contact from './components/Contact';

function App() {

  return (

    <Router>

      <div>

        <nav>

          <ul>

            <li>

              <a href="/">Home</a>

**Web Engineering**

            </li>

            <li>

              <a href="/about">About</a>

            </li>

            <li>

              <a href="/contact">Contact</a>

            </li>

          </ul>

        </nav>

        <Routes>

          <Route path="/" element={<Home />} />

          <Route path="/about" element={<About />} />

          <Route path="/contact" element={<Contact />} />

        </Routes>

      </div>

    </Router>

  );

}

export default App;

Name: Hamood Ahmed Lab#7 STD ID: 19694

**Task 1**

**Home.js Code**

import React from 'react';

function Home() {

  return (

    <div>

**Web Engineering**

      <h2>This is Home Page</h2>

      <p>

        A Home Page is the main web page that a visitor will view when they

        navigate to a website via a search engine, and it may also function

        as a landing page to attract visitors.

      </p>

      <imgsrc="/Home.jpg" alt="Home" />

    </div>

  );

}

export default Home;

Name: Hamood Ahmed Lab#7 STD ID: 19694

**Task 1**

**About.js Code**

import React from 'react';

function About() {

  return (

    <div>

      <h2>This is About Page</h2>

      <p>

        An About Us Page is a section on a website that provides information about a

        company, organization, or individual. It's an opportunity to tell your brand's

        story, share your vision or mission, introduce team members, and outline your

        history and achievements.

      </p>

      <imgsrc="/About.jpg" alt="About" />

    </div>

  );

}

**Web Engineering**

export default About;

Name: Hamood Ahmed Lab#7 STD ID: 19694

**Task 1**

**Contact.js Code**

import React from 'react';

function Contact() {

  return (

    <div>

      <h2>This is Contact Page</h2>

      <p>

        A contact page is a common web page on a website for visitors to contact

        the organization or individual providing the website. The page contains

        one or more of the following items: an e-mail address.

      </p>

      <imgsrc="/Contact.JPEG" alt="Contact" />

    </div>

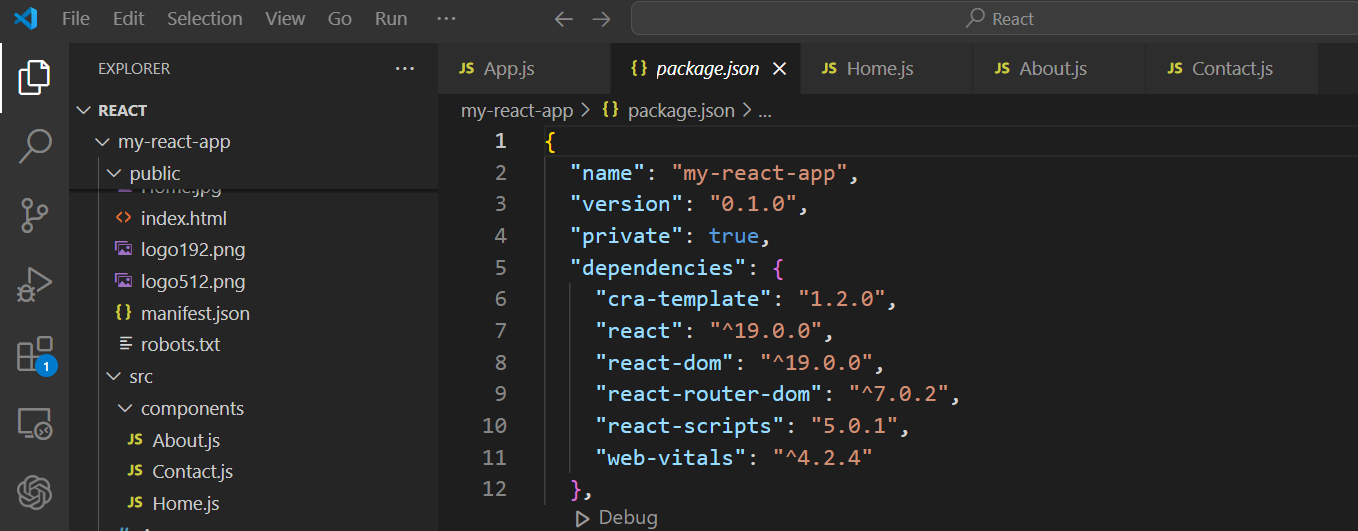
  );

}

export default Contact;

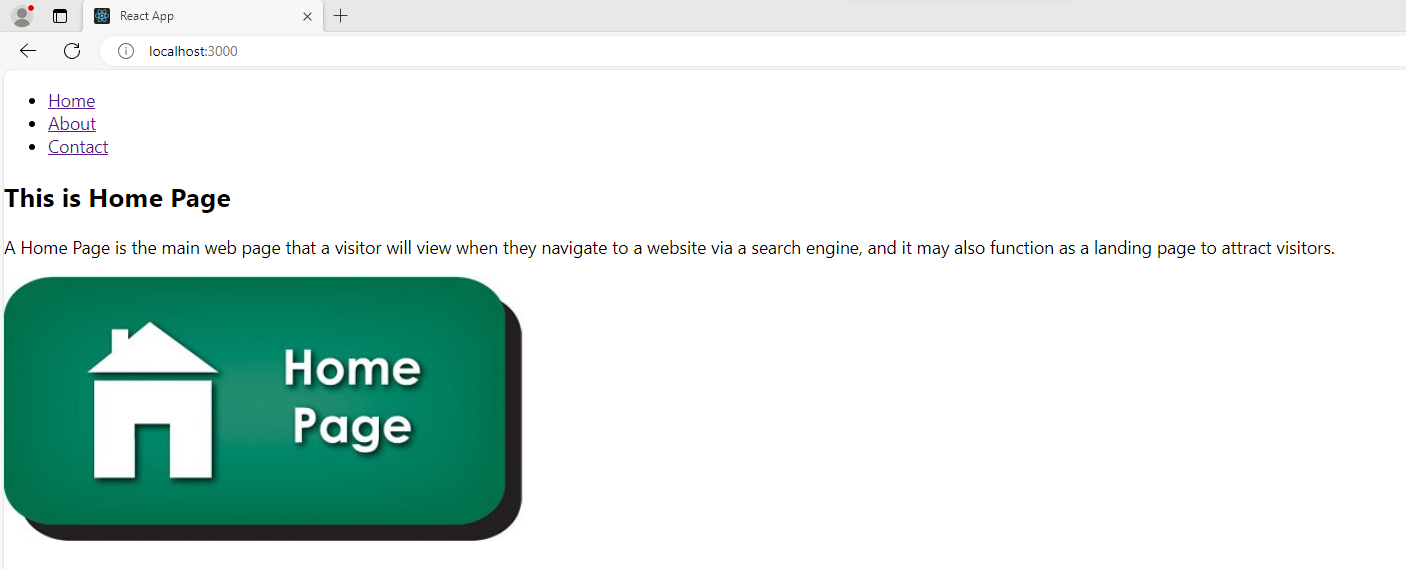
**Routing is Implemented and React-Router-Dom is Installed**

**Web Engineering**



**Output(React-Router-Page)**

**HomePage**



**About Page**

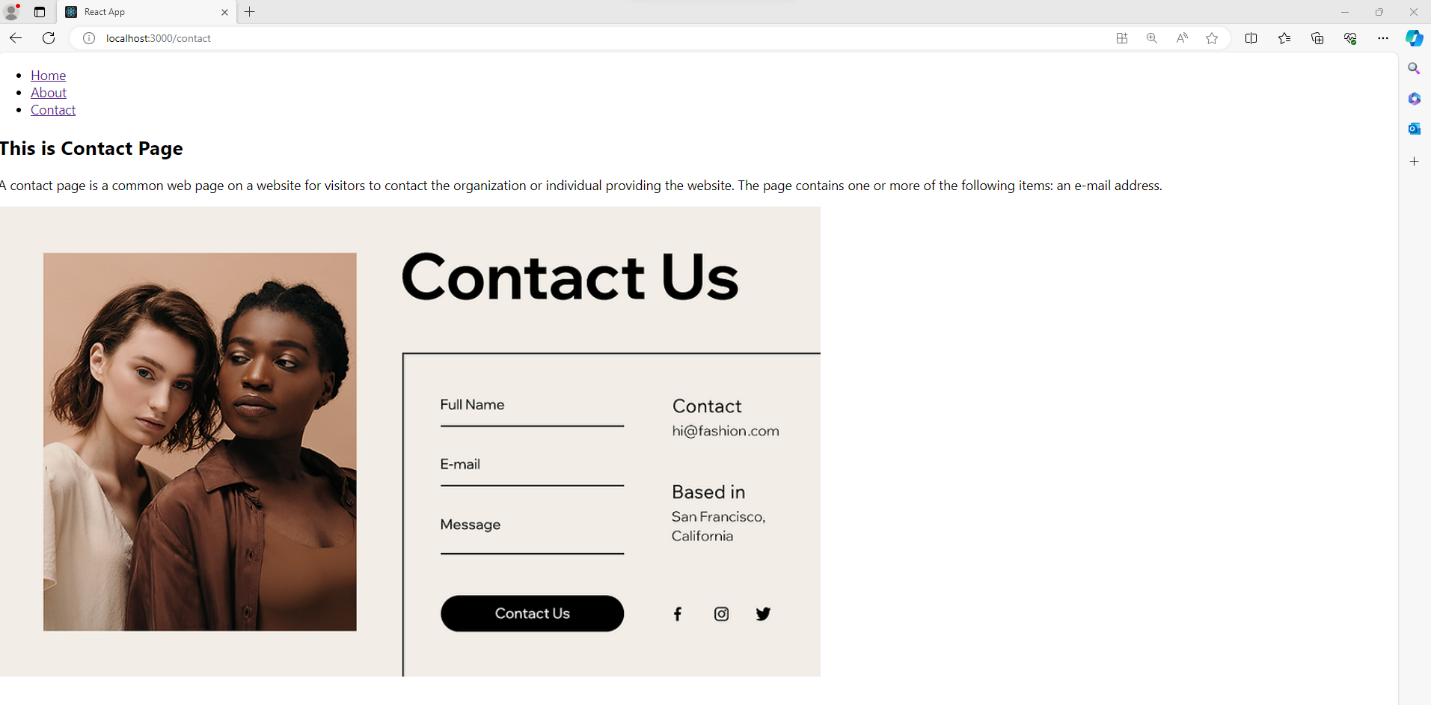
**Web Engineering**

Name: Hamood Ahmed Lab#7 STD ID: 19694

**Task 1**



**Contact Page**



**Web Engineering**

Name: Hamood Ahmed Lab#7 STD ID: 19694

**Task 2**

2. Use the boostrap nav bar and do the code to make it similar as shown below



**App.js**

Code:

import React from 'react';

import 'bootstrap/dist/css/bootstrap.min.css';

import './index.css';

function App() {

  return (

    <div className="container mt-5">

      <h1 className="text-center heading">React JS Frontend</h1>

      <nav className="mt-4 d-flex justify-content-start">

        <button className="btnbtn-outline-primary mx-2">Home</button>

        <button className="btnbtn-outline-primary mx-2">Time Slot</button>

        <button className="btnbtn-outline-primary mx-2">Course</button>

      </nav>

    </div>

  );

}

export default App;

**Web Engineering**

Name: Hamood Ahmed Lab#7 STD ID: 19694

**Task 2**

**For Page Styling**

**Index.css File**

Code:

body {

  background-color: #e6e7eb;

  font-family: Verdana, Geneva, Tahoma, sans-serif;

  margin: 0;

  padding: 0;

}

.heading {

  font-size: 2.5rem;

  font-weight: bold;

  color: rgb(99, 97, 97);

}

nav {

  padding: 10px;

}

.btn {

  font-size: 1.5rem;

  color: #007bff !important;

  border-color: #007bff !important;

}

.btn:hover {

  background-color: #e9f5ff;

  color: #0056b3 !important;

**Web Engineering**

}

**Index.JS File For ReactDOM Rendering**

Code:

import 'bootstrap/dist/css/bootstrap.min.css';

import './index.css';

import React from 'react';

import ReactDOM from 'react-dom/client';

import App from './App';

const rootElement = document.getElementById('root');

//ReactDOM for Rendering the Application

const root = ReactDOM.createRoot(rootElement);

root.render(

  <React.StrictMode>

    <App />

  </React.StrictMode>

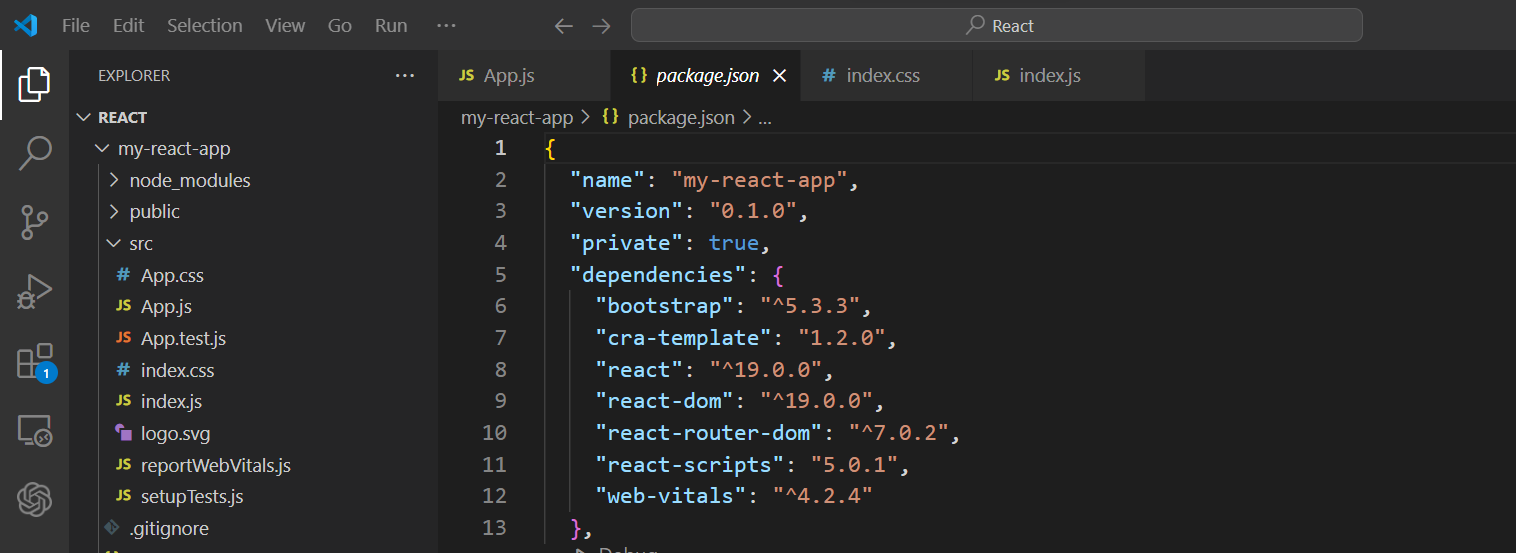
);

Name: Hamood Ahmed Lab#7 STD ID: 19694

**Task 2**

**Bootstrap is Implemented and Installed**

**Web Engineering**



Name: Hamood Ahmed Lab#7 STD ID: 19694

**Task 2**

**Output (Bootstrap Navbar)**



**Navbar Hover**



**Web Engineering**

Name: Hamood Ahmed Lab#7 STD ID: 19694

**Task 2**

If Permission Not Set See how we can set the permission

https://www.youtube.com/watch?v=J\_596H-sWsk

How to Install React JS In Visual Studio Code

https://www.youtube.com/watch?v=HIdPpm-0ZNQ

**Web Engineering**

**Web Engineering (LAB) 8**

**Name: Hamood Ahmed**

**ID: 022-20-119694**

**Program: BSCS**

**Subject: Web Engineering (LAB)**

**Course Instructor: Sir Syed Atir Iftikhar**

**Lab Tasks:**

**1. Add a new navigation item called "Profile"**

**2. Create a profile page component with a user information form**

**3. Implement form validation using React state**

**4. Style the user form using Tailwind CSS**

**5. Style the button using Tailwind CSS**

Name: Hamood Ahmed Lab#8 STD ID: 19694

**Task 1**

1. **Add a new navigation item called "Profile"**

**Navbar.js**

**Code:**

import React from 'react';

import { Link } from 'react-router-dom';

const Navbar = () => {

  return (

    <nav className="bg-blue-500 p-4 text-white flex justify-between items-center">

      <h1 className="text-2xl font-bold">Web Page</h1>

      <p className="mb-3 text-gray-500 dark:text-gray-400">The Home Page is the Most Important Page on any

Website and receives the most views of any Page.</p>

      <ul className="flex space-x-6">

        <li>

          <Link to="/" className="hover:underline">

            Home

          </Link>

        </li>

        <li>

          <Link to="/profile" className="hover:underline">

            Profile

          </Link>

        </li>

      </ul>

    </nav>

  );

};

export default Navbar;

**Web Engineering**

Name: Hamood Ahmed Lab#8 STD ID: 19694

**Task 2**

1. **Create a profile page component with a user information form**

**Profile.js**

**Code:**

import React, { useState } from 'react';

const Profile = () => {

  return (

    <div className="min-h-screen bg-gray-100 flex flex-col items-center py-10">

      <h2 className="text-3xl font-semibold text-blue-600 mb-6">User Profile</h2>

      <UserForm />

    </div>

  );

};

const UserForm = () => {

  const [formData, setFormData] = useState({

    name: '',

    email: '',

    phone: '',

  });

  const [errors, setErrors] = useState({});

  const handleChange = (e) => {

    setFormData({

      ...formData,

      [e.target.name]: e.target.value,

    });

  };

  const validate = () => {

    const newErrors = {};

    if (!formData.name) newErrors.name = "Name is required";

    if (!formData.email) newErrors.email = "Email is required";

    else if (!/^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(formData.email))

      newErrors.email = "Invalid email format";

    if (!formData.phone) newErrors.phone = "Phone number is required";

    else if (!/^\d{10}$/.test(formData.phone))

      newErrors.phone = "Phone number must be 10 digits";

    setErrors(newErrors);

    return Object.keys(newErrors).length === 0;

  };

  const handleSubmit = (e) => {

    e.preventDefault();

    if (validate()) {

      alert("Form submitted successfully!");

    }

  };

  return (

    <form className="bg-white p-8 rounded-lg shadow-lg w-full max-w-md" onSubmit={handleSubmit}>

      <div className="mb-6">

        <label className="block text-sm font-bold text-gray-700" htmlFor="name">

          Name

        </label>

        <input

          type="text"

          id="name"

          name="name"

          value={formData.name}

          onChange={handleChange}

          className={`w-full p-2 border rounded-lg mt-1 ${errors.name ? 'border-red-500' : 'border-gray-300'}`}

        />

        {errors.name && <p className="text-red-500 text-sm mt-1">{errors.name}</p>}

      </div>

      <div className="mb-6">

        <label className="block text-sm font-bold text-gray-700" htmlFor="email">

          Email

        </label>

        <input

          type="email"

          id="email"

          name="email"

          value={formData.email}

          onChange={handleChange}

          className={`w-full p-2 border rounded-lg mt-1 ${errors.email ? 'border-red-500' : 'border-gray-300'}`}

        />

        {errors.email && <p className="text-red-500 text-sm mt-1">{errors.email}</p>}

      </div>

      <div className="mb-6">

        <label className="block text-sm font-bold text-gray-700" htmlFor="phone">

          Phone Number

        </label>

        <input

          type="text"

          id="phone"

          name="phone"

          value={formData.phone}

          onChange={handleChange}

          className={`w-full p-2 border rounded-lg mt-1 ${errors.phone ? 'border-red-500' : 'border-gray-300'}`}

        />

        {errors.phone && <p className="text-red-500 text-sm mt-1">{errors.phone}</p>}

      </div>

      <button

        type="submit"

        className="w-full bg-teal-500 text-white py-2 px-4 rounded-lg hover:bg-teal-600 font-semibold"

      >

        Save

      </button>

<img

  src="./Profile.png"

  alt="Profile"

  className="w-1/4 max-w-sm mx-auto rounded-lg shadow-lg"

/>

    </form>

  );

};

export default Profile;

**Web Engineering**

Name: Hamood Ahmed Lab#8 STD ID: 19694

**Task 3**

1. **Implement form validation using React state**

**Profile.js File Code Part (Form Validation Using React State)**

const UserForm = () => {

  const [formData, setFormData] = useState({

    name: '',

    email: '',

    phone: '',

  });

  const [errors, setErrors] = useState({});

  const handleChange = (e) => {

    setFormData({

      ...formData,

      [e.target.name]: e.target.value,

    });

  };

  const validate = () => {

    const newErrors = {};

    if (!formData.name) newErrors.name = "Name is required";

    if (!formData.email) newErrors.email = "Email is required";

    else if (!/^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(formData.email))

      newErrors.email = "Invalid email format";

    if (!formData.phone) newErrors.phone = "Phone number is required";

    else if (!/^\d{10}$/.test(formData.phone))

      newErrors.phone = "Phone number must be 10 digits";

    setErrors(newErrors);

    return Object.keys(newErrors).length === 0;

  };

  const handleSubmit = (e) => {

    e.preventDefault();

    if (validate()) {

      alert("Form submitted successfully!");

    }

  };

  return (

    <form className="bg-white p-8 rounded-lg shadow-lg w-full max-w-md" onSubmit={handleSubmit}>

      <div className="mb-6">

        <label className="block text-sm font-bold text-gray-700" htmlFor="name">

          Name

        </label>

        <input

          type="text"

          id="name"

          name="name"

          value={formData.name}

          onChange={handleChange}

          className={`w-full p-2 border rounded-lg mt-1 ${errors.name ? 'border-red-500' : 'border-gray-300'}`}

        />

        {errors.name && <p className="text-red-500 text-sm mt-1">{errors.name}</p>}

      </div>

      <div className="mb-6">

        <label className="block text-sm font-bold text-gray-700" htmlFor="email">

          Email

        </label>

        <input

          type="email"

          id="email"

          name="email"

          value={formData.email}

          onChange={handleChange}

          className={`w-full p-2 border rounded-lg mt-1 ${errors.email ? 'border-red-500' : 'border-gray-300'}`}

        />

        {errors.email && <p className="text-red-500 text-sm mt-1">{errors.email}</p>}

      </div>

      <div className="mb-6">

        <label className="block text-sm font-bold text-gray-700" htmlFor="phone">

          Phone Number

        </label>

        <input

          type="text"

          id="phone"

          name="phone"

          value={formData.phone}

          onChange={handleChange}

          className={`w-full p-2 border rounded-lg mt-1 ${errors.phone ? 'border-red-500' : 'border-gray-300'}`}

        />

        {errors.phone && <p className="text-red-500 text-sm mt-1">{errors.phone}</p>}

      </div>

      <button

        type="submit"

        className="w-full bg-teal-500 text-white py-2 px-4 rounded-lg hover:bg-teal-600 font-semibold"

      >

        Save

      </button>

<img

  src="./Profile.png"

  alt="Profile"

  className="w-1/4 max-w-sm mx-auto rounded-lg shadow-lg"

/>

    </form>

**Web Engineering**

Name: Hamood Ahmed Lab#8 STD ID: 19694

**Task 4**

1. **Style the user form using Tailwind CSS**

**Profile.js File Code Part(User Form Styling Using Tailwind CSS)**

 return (

    <form className="bg-white p-8 rounded-lg shadow-lg w-full max-w-md" onSubmit={handleSubmit}>

      <div className="mb-6">

        <label className="block text-sm font-bold text-gray-700" htmlFor="name">

          Name

        </label>

        <input

          type="text"

          id="name"

          name="name"

          value={formData.name}

          onChange={handleChange}

          className={`w-full p-2 border rounded-lg mt-1 ${errors.name ? 'border-red-500' : 'border-gray-300'}`}

        />

        {errors.name && <p className="text-red-500 text-sm mt-1">{errors.name}</p>}

      </div>

      <div className="mb-6">

        <label className="block text-sm font-bold text-gray-700" htmlFor="email">

          Email

        </label>

        <input

          type="email"

          id="email"

          name="email"

          value={formData.email}

          onChange={handleChange}

          className={`w-full p-2 border rounded-lg mt-1 ${errors.email ? 'border-red-500' : 'border-gray-300'}`}

        />

        {errors.email && <p className="text-red-500 text-sm mt-1">{errors.email}</p>}

      </div>

      <div className="mb-6">

        <label className="block text-sm font-bold text-gray-700" htmlFor="phone">

          Phone Number

        </label>

        <input

          type="text"

          id="phone"

          name="phone"

          value={formData.phone}

          onChange={handleChange}

          className={`w-full p-2 border rounded-lg mt-1 ${errors.phone ? 'border-red-500' : 'border-gray-300'}`}

        />

        {errors.phone && <p className="text-red-500 text-sm mt-1">{errors.phone}</p>}

      </div>

      <button

        type="submit"

        className="w-full bg-teal-500 text-white py-2 px-4 rounded-lg hover:bg-teal-600 font-semibold"

      >

        Save

      </button>

<img

  src="./Profile.png"

  alt="Profile"

  className="w-1/4 max-w-sm mx-auto rounded-lg shadow-lg"

/>

    </form>

  );

**Web Engineering**

Name: Hamood Ahmed Lab#8 STD ID: 19694

**Task 5**

1. **Style the button using Tailwind CSS**

**Profile.js File Code Part(Button Styling Using Tailwind CSS)**

<button

        type="submit"

        className="w-full bg-teal-500 text-white py-2 px-4 rounded-lg hover:bg-teal-600 font-semibold"

      >

        Save

      </button>

**App.js File**

**Code:**

import React from 'react';

import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';

import Navbar from './Navbar';

import Profile from './Profile';

function App() {

  return (

    <Router>

      <div>

        <Navbar />

        <Routes>

          <Route

            path="/"

            element={

              <div className="text-center p-6">

                <h1 className="text-3xl font-bold mb-4">Welcome To The Home Page!</h1>

                <img

                  src="./Home.png"

                  alt="Home"

                  className="w-full max-w-4xl mx-auto rounded-lg shadow-lg"

                />

              </div>

            }

          />

          <Route path="/profile" element={<Profile />} />

        </Routes>

      </div>

    </Router>

  );

}

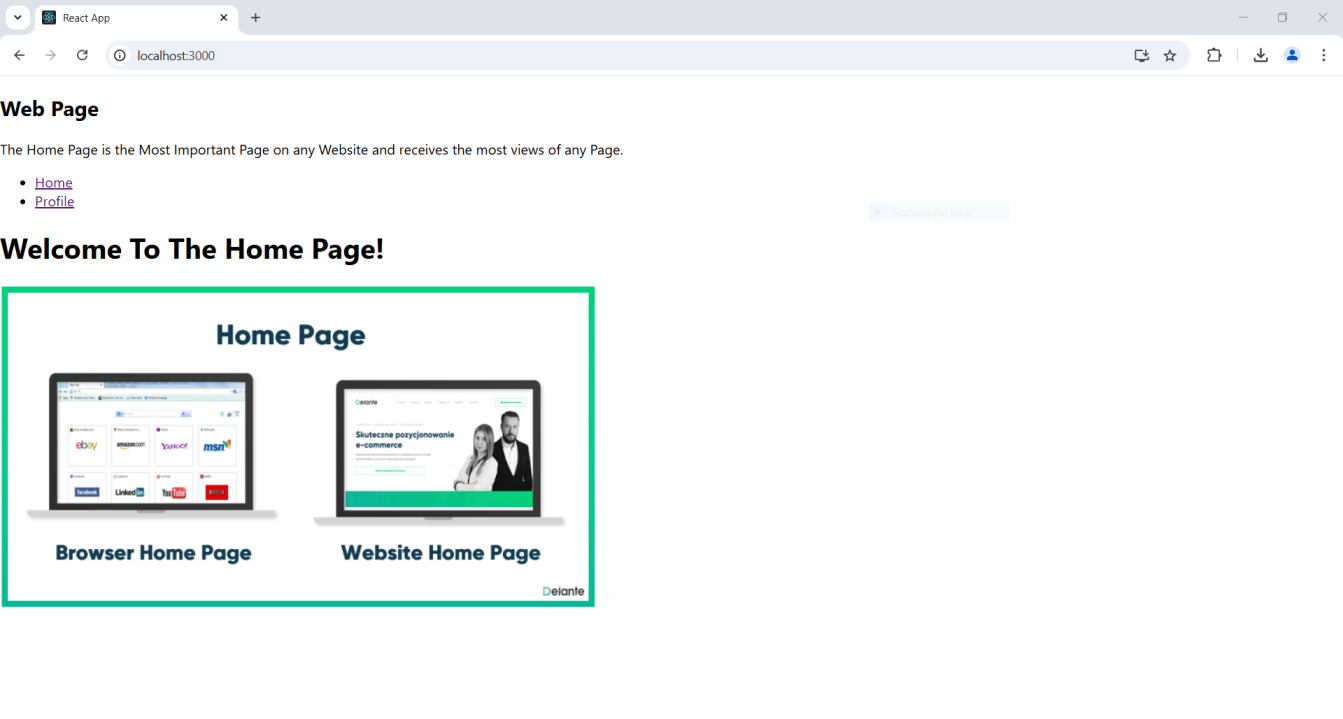
export default App;

**Web Engineering**

Name: Hamood Ahmed Lab#8 STD ID: 19694

**Task 5**

**Output (Home Page)**

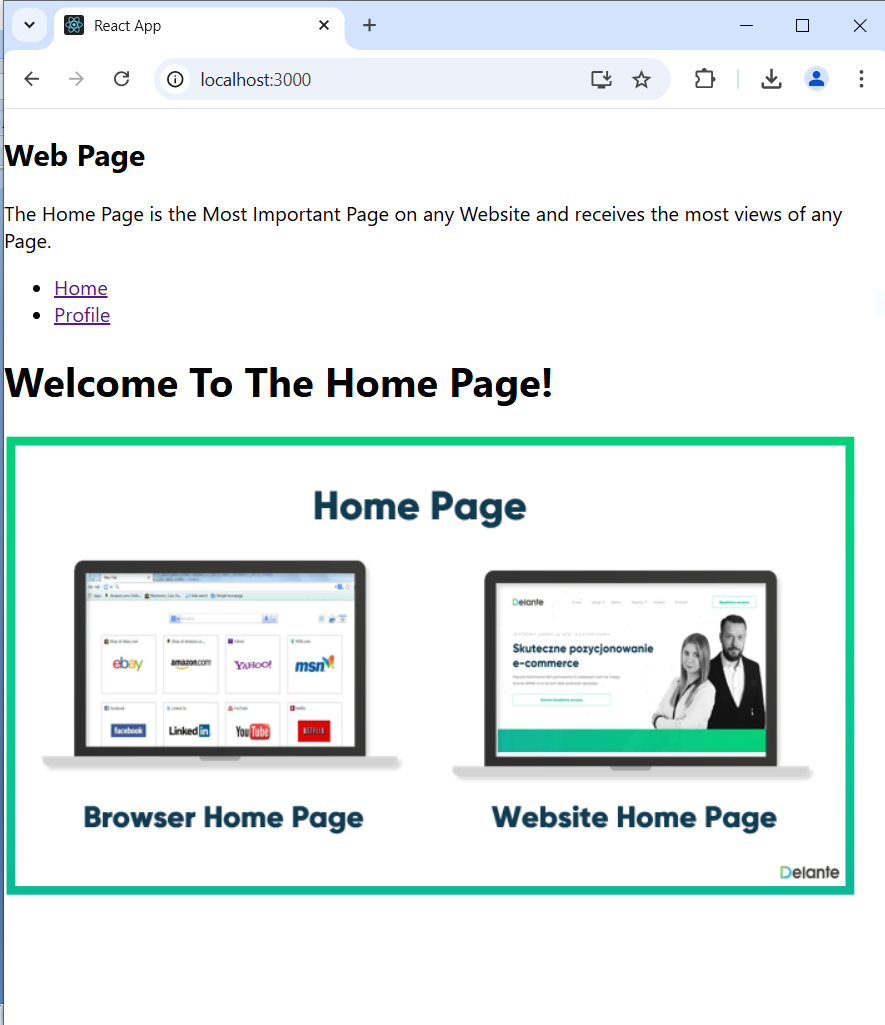
****

**Web Engineering**

Name: Hamood Ahmed Lab#8 STD ID: 19694

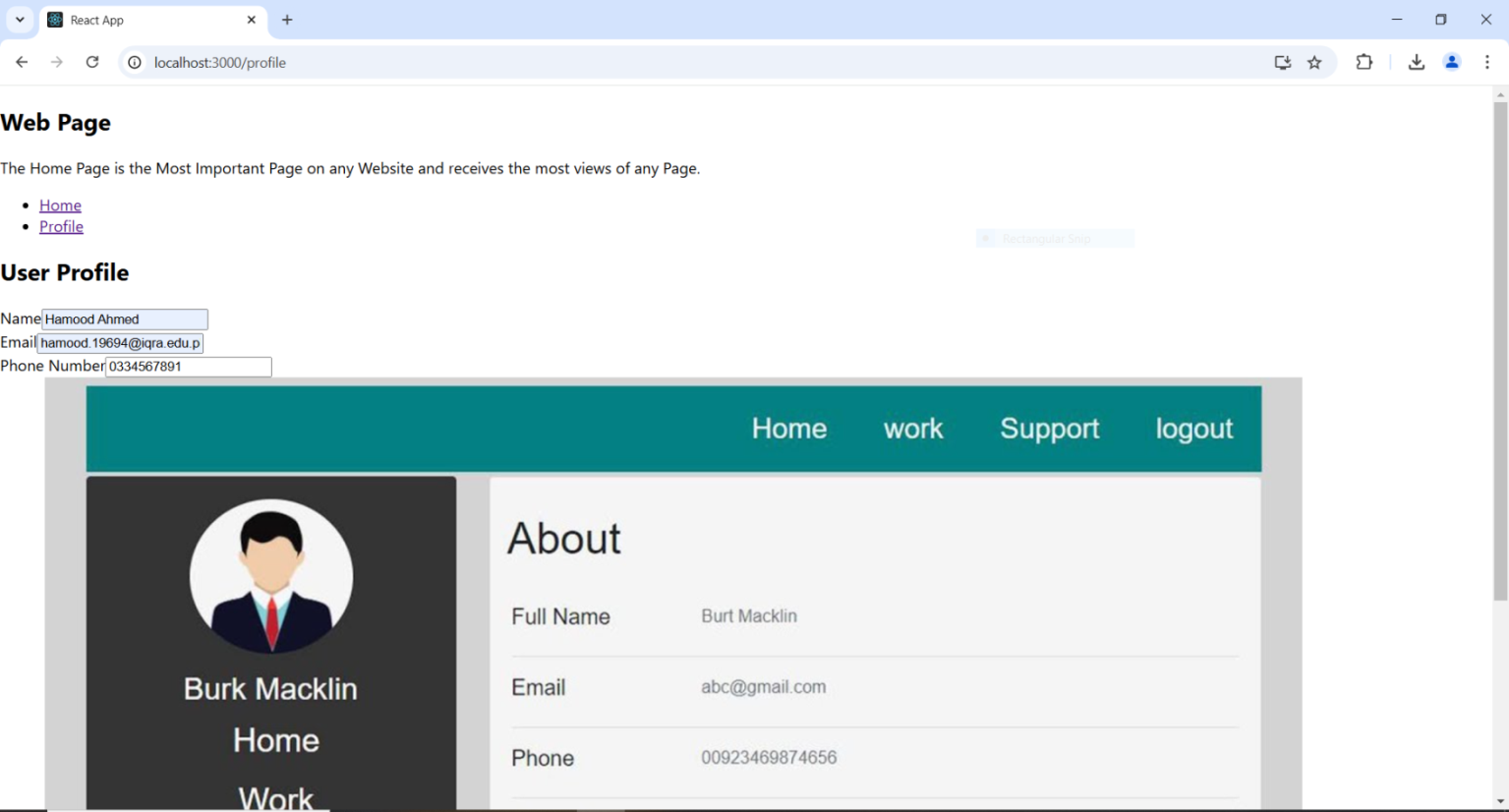
**Task 5**

**Output (Home Page Fully Responsive)**

****

**Output Profile Page 🡪 User Profile and Form**

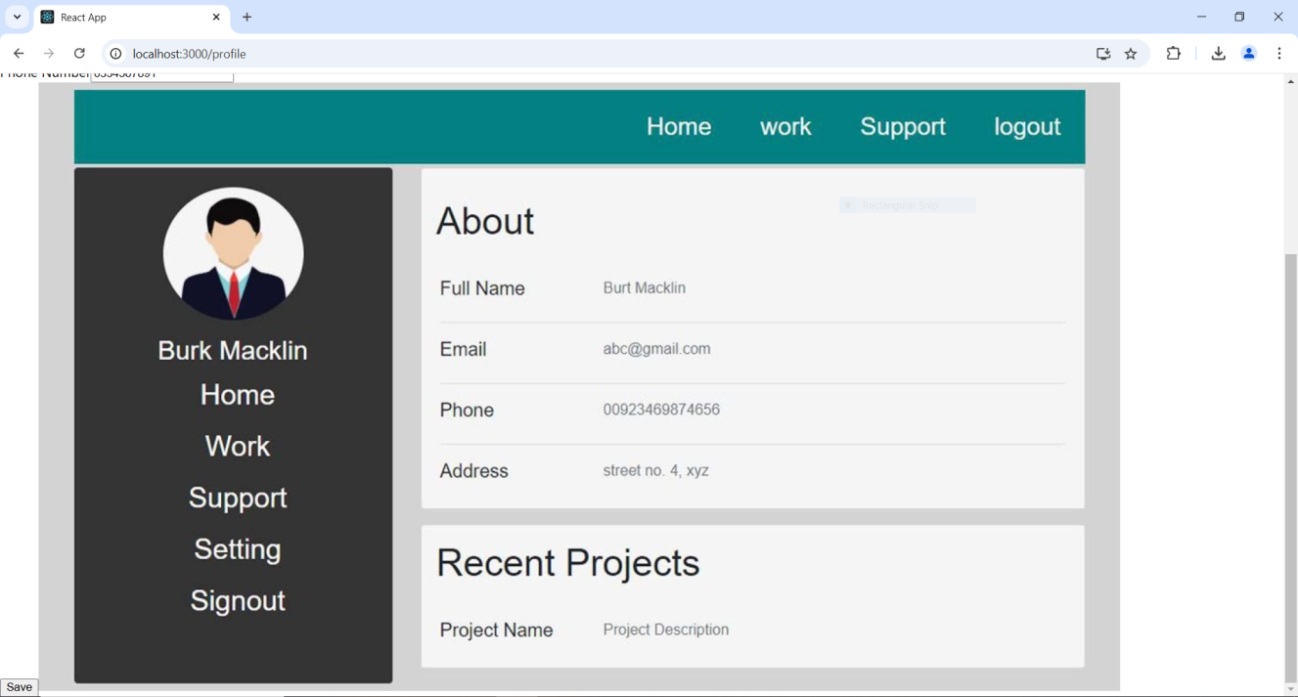
**Web Engineering**

****

Name: Hamood Ahmed Lab#8 STD ID: 19694

**Task 5**

**Output(Form Filling and Form Validation)**

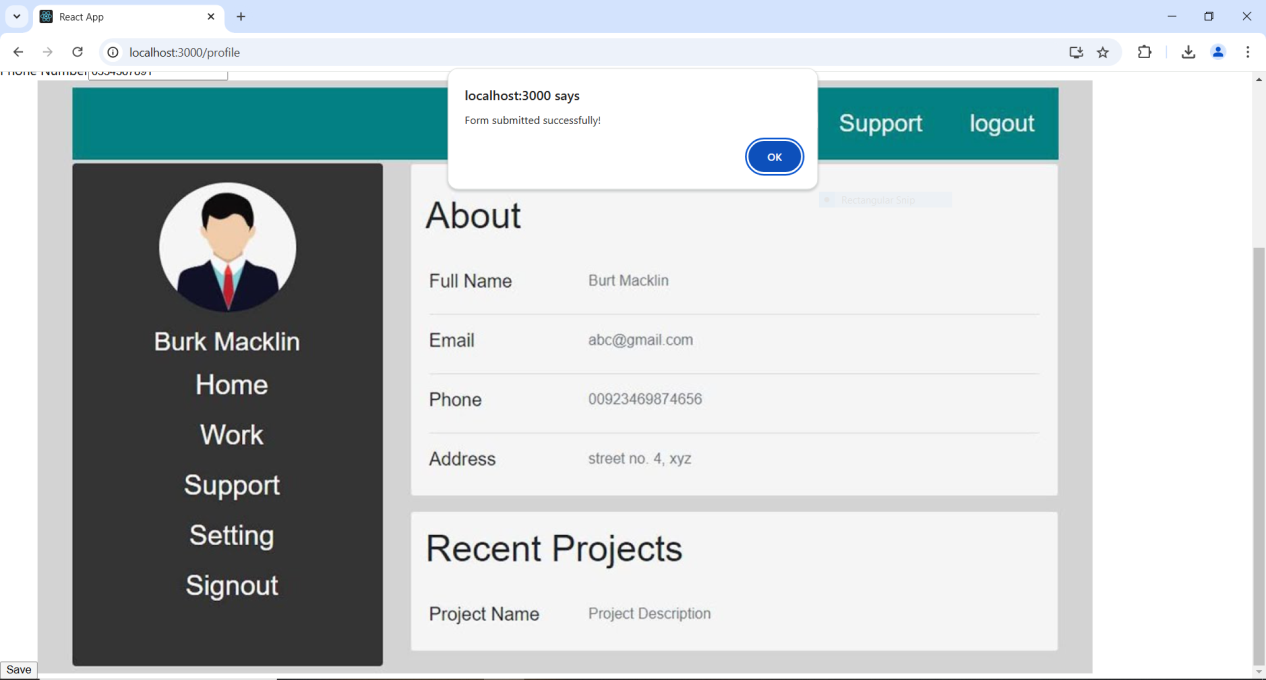
****

**Web Engineering**

Name: Hamood Ahmed Lab#8 STD ID: 19694

**Task 5**

**Output(Form Submitted Successfully of Profile Page “Fully Responsive” when Clicking on “Save” Button)**

****

**Tailwind.config.js(Using Tailwind CSS For Styling in React)**

**Code**

module.exports = {

  content: [

    "./src/\*\*/\*.{js,jsx,ts,tsx}",

    "./public/index.html",

  ],

  theme: {

    extend: {},

  },

  plugins: [],

};

**Postcss.config.js(For Tailwind CSS Initialization)**

module.exports = {

    plugins: {

      tailwindcss: {},

      autoprefixer: {},

    },

  };

**input.css(For Tailwind CSS Declaration)**

@tailwind base;

@tailwind components;

@tailwind utilities;

**Package.json(react-router-dom,tailwindcss and redux all packages Installed for React)**

{

  "name": "myapp",

  "version": "0.1.0",

  "private": true,

  "dependencies": {

    "@reduxjs/toolkit": "^2.5.0",

    "autoprefixer": "^10.4.20",

    "cra-template": "1.2.0",

    "postcss": "^8.4.49",

    "react": "^19.0.0",

    "react-dom": "^19.0.0",

    "react-redux": "^9.2.0",

    "react-router-dom": "^6.28.0",

    "react-scripts": "5.0.1"

  },

  "scripts": {

    "start": "react-scripts start",

    "build": "react-scripts build",

    "test": "react-scripts test",

    "eject": "react-scripts eject"

  },

  "eslintConfig": {

    "extends": [

      "react-app",

      "react-app/jest"

    ]

  },

  "browserslist": {

    "production": [

      ">0.2%",

      "not dead",

      "not op\_mini all"

    ],

    "development": [

      "last 1 chrome version",

      "last 1 firefox version",

      "last 1 safari version"

    ]

  },

  "devDependencies": {

    "tailwindcss": "^3.4.17"

  }

}

**Web Engineering**

**Web Engineering (LAB) 9**

**Name: Hamood Ahmed**

**ID: 022-20-119694**

**Program: BSCS**

**Subject: Web Engineering (LAB)**

**Course Instructor: Sir Syed Atir Iftikhar**

**Lab Tasks:**

1. Create a Simple Redux Counter App:

Implement a counter app where students can increase, decrease, or reset the count.

Add buttons for each action and use Redux to manage the state.

2. Explore Redux DevTools:

Enable Redux DevTools in their project and monitor state changes while

performing actions in their app.

3. Modify the ToDo App:

Add functionality to edit an existing task.

Enhance the app by categorizing tasks (e.g., “Personal” and “Work”).

Name: Hamood Ahmed Lab# 9 STD ID: 19694

**Task All**

**Tasks**

**App.js (Code):**

import React from 'react';

import Counter from './components/Counter';

import ToDo from './components/ToDo';

const App = () => {

  return (

    <div className="app">

      <Counter />

      <ToDo />

    </div>

  );

};

export default App;

**Counter.js(Code For Counter App)**

import React from 'react';

import { useDispatch, useSelector } from 'react-redux';

import { increment, decrement, reset } from '../redux/counterSlice';

import './Counter.css';

const Counter = () => {

  const count = useSelector((state) =>state.counter);

  const dispatch = useDispatch();

  return (

    <div className="counter">

      <h2>Counter App</h2>

      <h3>Count: {count}</h3>

      <div className="buttons">

        <button onClick={() => dispatch(increment())}>Increment</button>

        <button onClick={() => dispatch(decrement())}>Decrement</button>

        <button onClick={() => dispatch(reset())}>Reset</button>

**Web Engineering**

      </div>

    </div>

  );

};

export default Counter;

**Index.js (Code)**

import React from 'react';

import ReactDOM from 'react-dom/client';

import { Provider } from 'react-redux';

import { store } from './redux/store';

import App from './App';

import './index.css';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

  <Provider store={store}>

    <App />

  </Provider>

);

**Index.css(Code)**

body {

  font-family: Arial, sans-serif;

  margin: 0;

  padding: 0;

  background-color: #f3c664;

}

.app {

  padding: 20px;

}

**Store.js(Code)**

import { configureStore } from '@reduxjs/toolkit';

import counterReducer from './counterSlice';

import todoReducer from './todoSlice';

export const store = configureStore({

  reducer: {

    counter: counterReducer,

    todos: todoReducer,

  },

});

**Counter.css (Code)**

.counter {

    margin: 20px 0;

    padding: 20px;

    border: 1px solid #0e0404;

    border-radius: 8px;

    background: #68cfca;

  }

  .buttons button {

    margin: 5px;

    padding: 10px 20px;

    background: #a76208;

    color: #fff;

    border: none;

    border-radius: 5px;

    cursor: pointer;

  }

  .buttonsbutton:hover {

    background: #a83fc2;

  }

**ToDo.css (Code)**

.todo {

    padding: 20px;

    border: 1px solid #20054d;

    border-radius: 8px;

    background: #7fa0c7;

  }

  .input-group {

    display: flex;

    gap: 10px;

    margin-bottom: 10px;

  }

  .input-group input {

    flex: 1;

    padding: 10px;

    border: 1px solid #0d1542;

    border-radius: 5px;

  }

  .input-group select, .input-group button {

    padding: 10px;

    border: none;

    border-radius: 5px;

  }

  ul {

    list-style: none;

    padding: 0;

  }

  li {

    display: flex;

    justify-content: space-between;

    padding: 8px;

    border-bottom: 1px solid #4d3c0d;

  }

  li button {

    margin-left: 5px;

    padding: 5px 10px;

    background: #9b6707;

    color: #ffffff;

    border: none;

    border-radius: 5px;

    cursor: pointer;

  }

  li button:hover {

    background: #e93428;

  }

**ToDo.js (Code)**

import React, { useState } from 'react';

import { useDispatch, useSelector } from 'react-redux';

import { addTask, deleteTask, editTask } from '../redux/todoSlice';

import './ToDo.css';

const ToDo = () => {

  const [text, setText] = useState('');

  const [category, setCategory] = useState('Personal');

  const [editId, setEditId] = useState(null);

  const todos = useSelector((state) =>state.todos);

  const dispatch = useDispatch();

  const handleAddOrEdit = () => {

    if (editId) {

      dispatch(editTask({ id: editId, text, category }));

      setEditId(null);

    } else {

      dispatch(addTask({ text, category }));

    }

    setText('');

  };

  return (

    <div className="todo">

      <h2>ToDo App</h2>

      <div className="input-group">

        <input

          type="text"

          value={text}

          onChange={(e) =>setText(e.target.value)}

          placeholder="Enter a task"

        />

        <select value={category} onChange={(e) =>setCategory(e.target.value)}>

          <option value="Personal">Personal</option>

          <option value="Work">Work</option>

        </select>

        <button onClick={handleAddOrEdit}>{editId ? 'Edit Task' : 'Add Task'}</button>

      </div>

      <ul>

        {todos.map((task) => (

          <li key={task.id}>

            <span>{task.text} ({task.category})</span>

            <button onClick={() => dispatch(deleteTask(task.id))}>Delete</button>

            <button

              onClick={() => {

                setText(task.text);

                setCategory(task.category);

                setEditId(task.id);

              }}

            >

              Edit

            </button>

          </li>

        ))}

      </ul>

    </div>

  );

};

export default ToDo;

**counterSlice.js(Code)**

import { createSlice } from '@reduxjs/toolkit';

const counterSlice = createSlice({

  name: 'counter',

  initialState: 0,

  reducers: {

    increment: (state) => state + 1,

    decrement: (state) => state - 1,

    reset: () => 0,

  },

});

export const { increment, decrement, reset } = counterSlice.actions;

export default counterSlice.reducer;

**todoSlice.js(Code)**

import { createSlice } from '@reduxjs/toolkit';

const todoSlice = createSlice({

  name: 'todos',

  initialState: [],

  reducers: {

    addTask: (state, action) => {

      state.push({ id: Date.now(), text: action.payload.text, category: action.payload.category });

    },

    deleteTask: (state, action) =>state.filter((task) => task.id !== action.payload),

    editTask: (state, action) => {

      const { id, text, category } = action.payload;

      const task = state.find((task) => task.id === id);

      if (task) {

        task.text = text;

        task.category = category;

      }

    },

  },

});

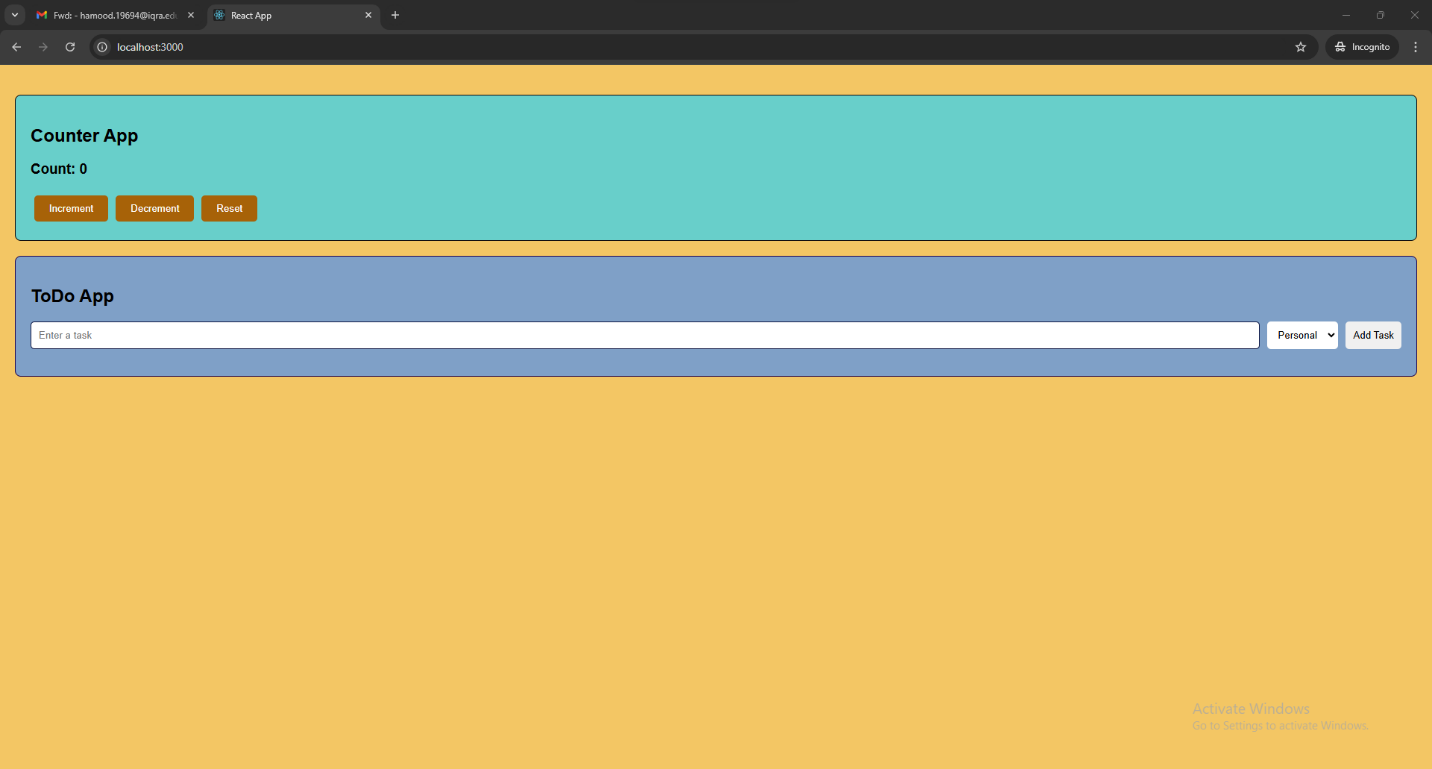
Name: Hamood Ahmed Lab# 9 STD ID: 19694

**Task All**

export const { addTask, deleteTask, editTask } = todoSlice.actions;

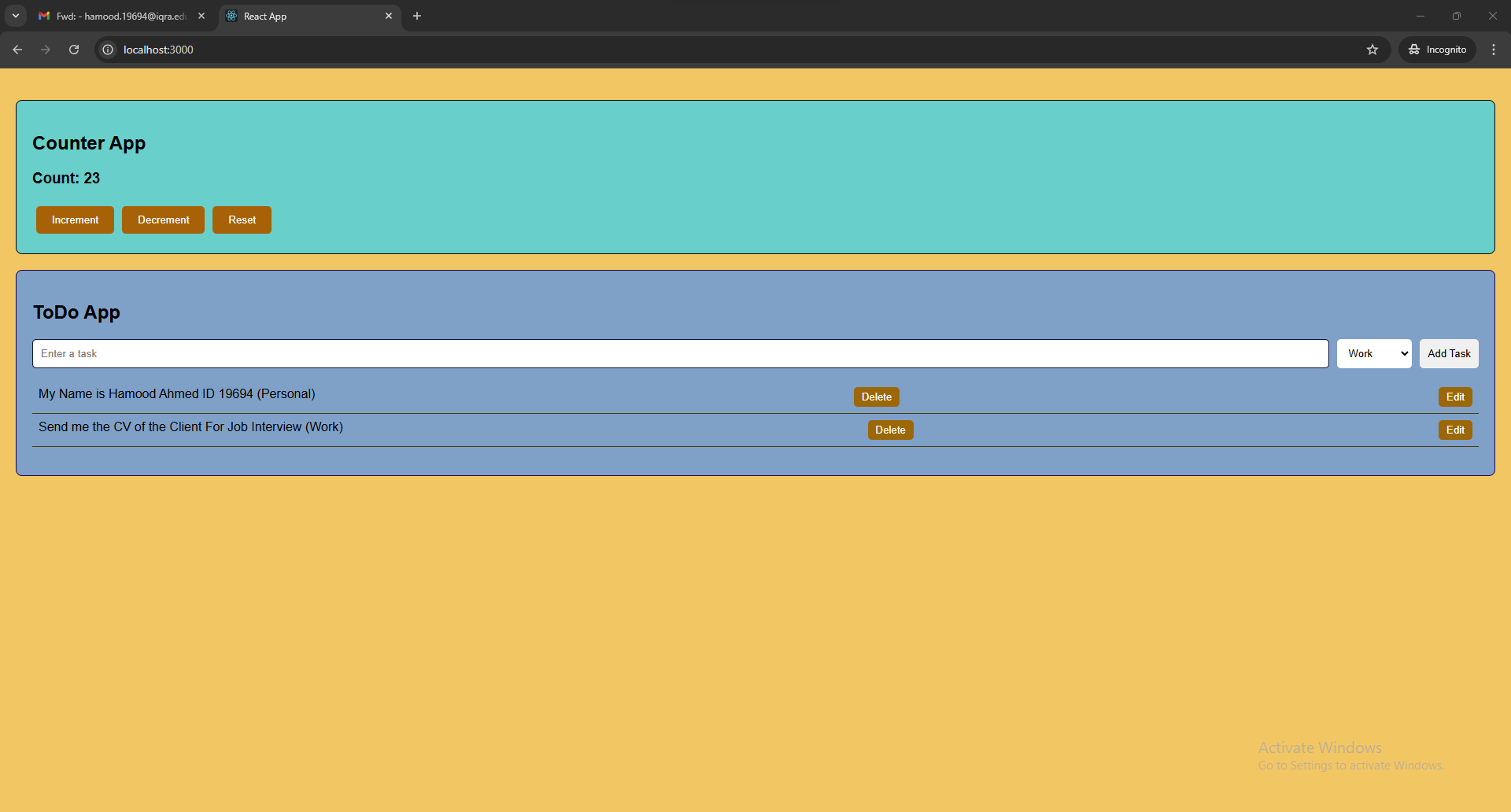
export default todoSlice.reducer;

**Snap 1(Output Counter App and ToDo App Functionality for Writing Personal and Work Task):**

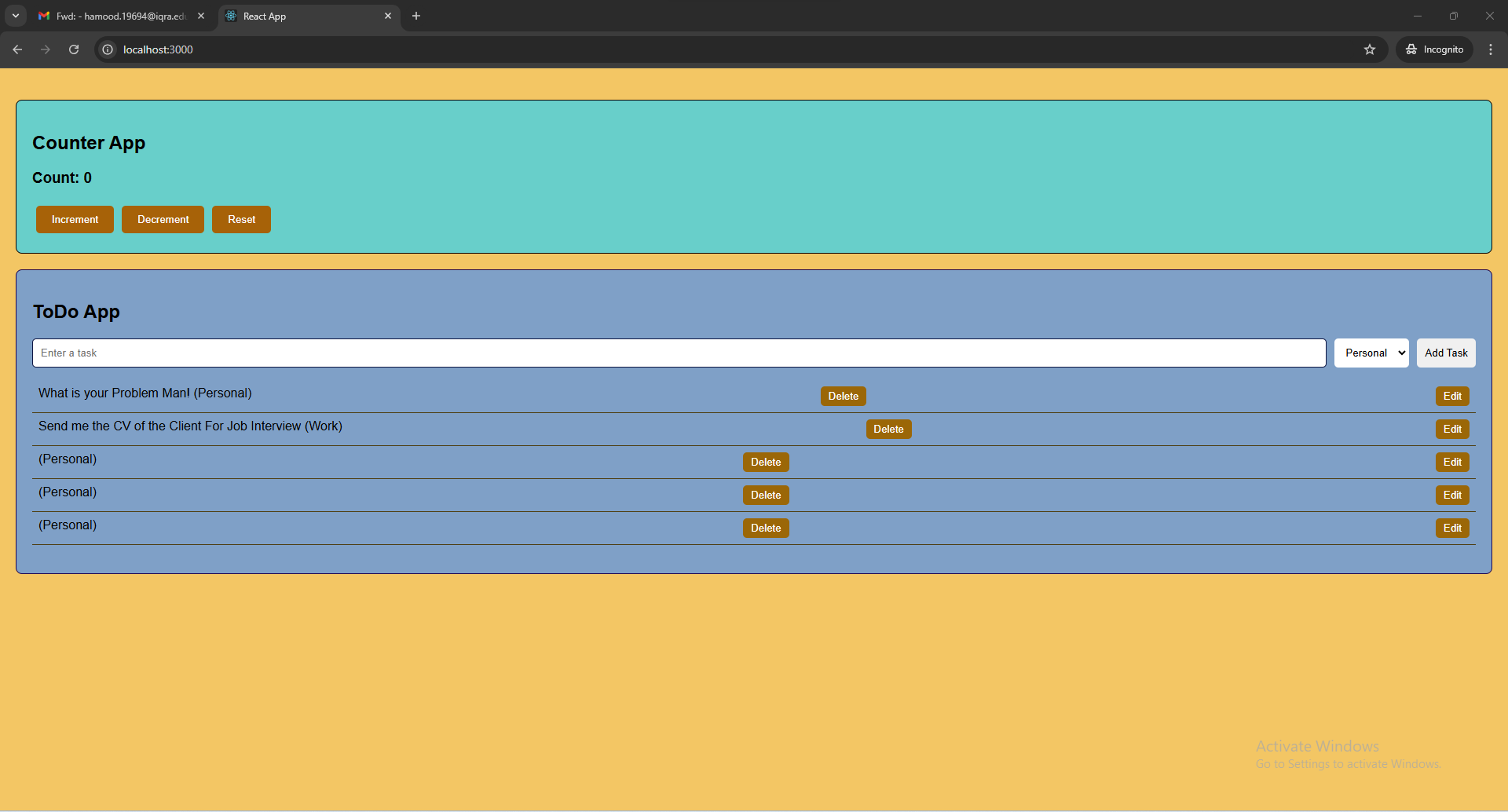


**Snap 2(Counter App Increment and Decrement + ToDo App Tasks Personal and Work):**

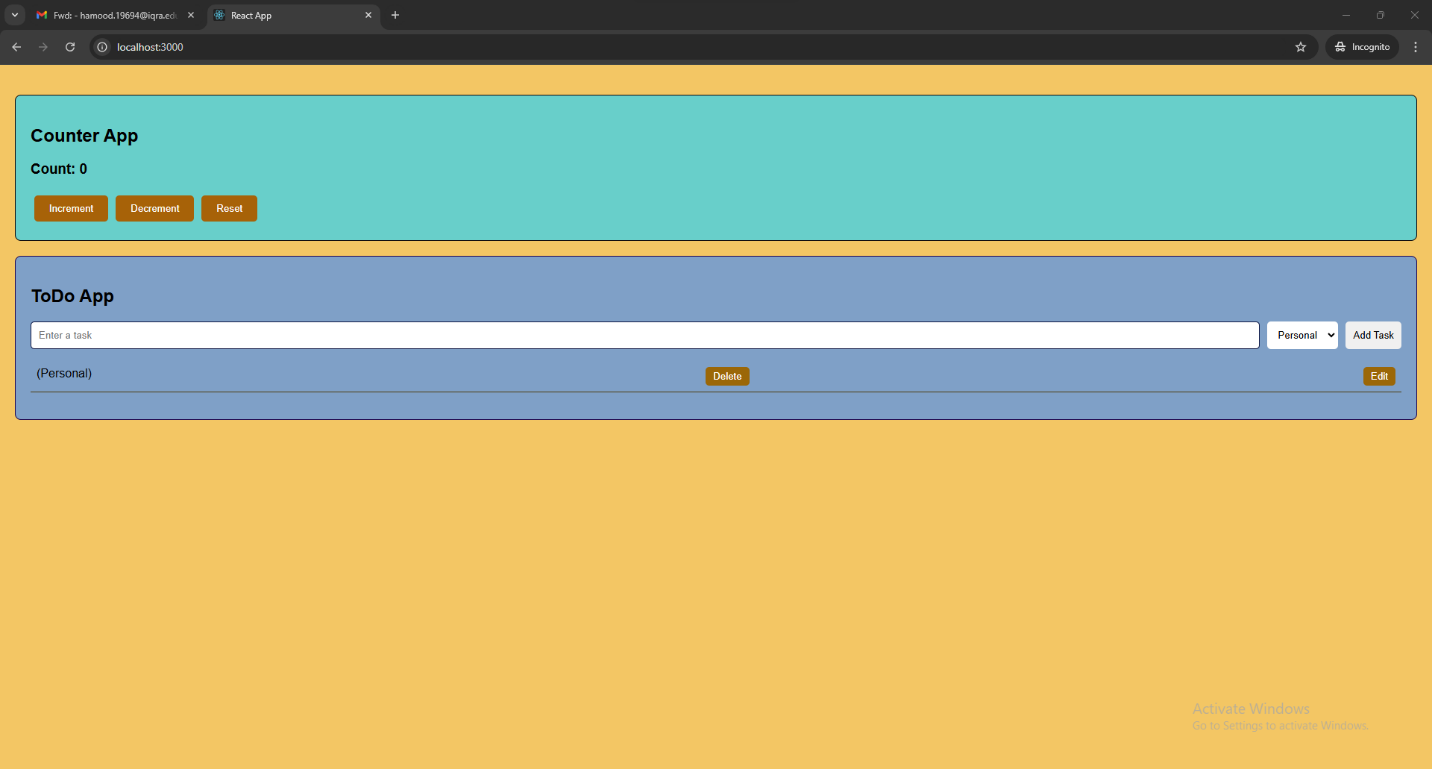
**Web Engineering**



**Snap 3(ToDo App Tasks Edit and Delete Also):**



**Snap 4(ToDo App Tasks Deleted and Whole Web-Page is Working Correctly):**



**Web Engineering**