# **PRACTICAL – 1**

**AIM: Design a User Interface for a) Welcome screen b) Multiplication and Addition of any two numbers**

* **Code of Welcome.html:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

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

<title>Welcome</title>

<style>

\* {

box-sizing: border-box;

padding: 0;

margin: 0;

font-family: "Times New Roman", Times, serif;

}

html {

width: 100%;

height: 100%;

background: linear-gradient(#e66465, #9198e5, rgb(1, 59, 250));

background-repeat: no-repeat;

}

#box {

display: flex;

width: 40%;

height: 60%;

background-color: white;

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

border: none;

z-index: 999;

border-radius: 18px;

background-color: rgba(255, 255, 255, 0.479);

box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);

backdrop-filter: blur(10px);

}

#box a {

display: flex;

width: 100%;

flex-wrap: wrap;

font-weight: 700;

text-align: center;

text-transform: uppercase;

align-items: center;

justify-content: center;

font-size: 48px;

border: none;

text-decoration: none;

color: #2030e0;

border-radius: 18px;

box-shadow: rgba(0, 0, 0, 0.548) 20px 10px 30px 20px;

}

#box a:hover{

color: #074feb;

text-decoration: underline;

cursor: pointer;

}

</style>

</head>

<body>

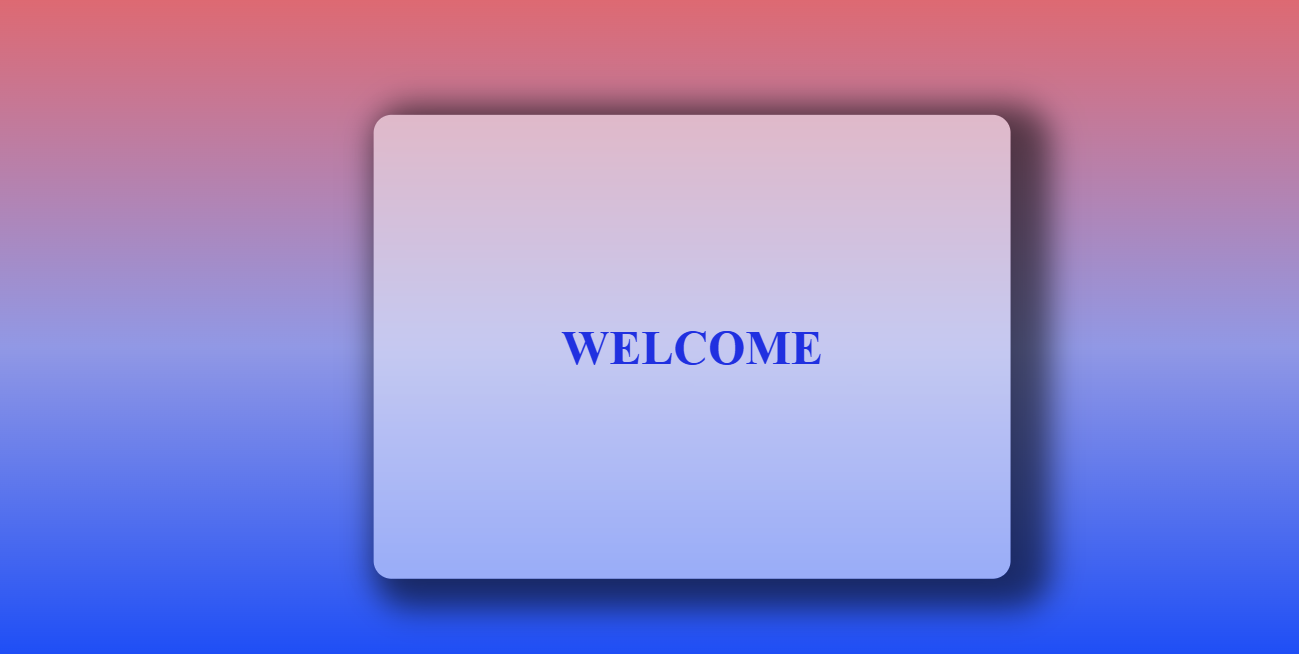
<div id="box">

<a href="cal.html">Welcome</a>

</div>

</body>

</html>

* **Output:**
* **Code of cal.html:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1" />

<title>Simple Calculator</title>

<style>

body {

background: linear-gradient(#e66465, #9198e5, rgb(1, 59, 250));

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

}

.calculator {

background: #ffffffdd;

padding: 2rem 3rem;

border-radius: 15px;

box-shadow: rgba(0, 0, 0, 0.548) 20px 10px 30px 20px;

text-align: center;

width: 300px;

}

input[type="number"] {

width: 120px;

padding: 0.5rem;

margin: 0.5rem 0.25rem;

font-size: 1.1rem;

border-radius: 8px;

border: 1.5px solid #a777e3;

outline-color: #6e8efb;

}

button {

padding: 0.55rem 1.5rem;

margin: 0.5rem 0.5rem 1rem 0.5rem;

font-size: 1.3rem;

border: none;

border-radius: 12px;

background-color: #6e8efb;

color: white;

cursor: pointer;

transition: background-color 0.3s ease;

}

button:hover {

background-color: #a777e3;

}

#result {

display: block;

font-size: 1.5rem;

font-weight: 600;

color: #4a2c9f;

min-height: 2em;

margin-top: 0.5rem;

}

</style>

</head>

<body>

<div class="calculator">

<input type="number" name="num1" id="num1" placeholder="Enter a number" />

<input type="number" name="num2" id="num2" placeholder="Enter a number" />

<div>

<button id="plus" title="Add">+</button>

<button id="mul" title="Multiply">\*</button>

</div>

<label id="result" aria-live="polite"></label>

</div>

<script type="text/javascript">

let btnPlus = document.getElementById("plus");

let btnMul = document.getElementById("mul");

let input1 = document.getElementById("num1");

let input2 = document.getElementById("num2");

let resultLabel = document.getElementById("result");

function parseInputs() {

let val1 = parseFloat(input1.value);

let val2 = parseFloat(input2.value);

return { val1, val2 };

}

btnPlus.addEventListener("click", function (event) {

const { val1, val2 } = parseInputs();

if (isNaN(val1) || isNaN(val2)) {

resultLabel.textContent =

"Please enter valid numbers in both fields.";

return;

}

const sum = val1 + val2;

resultLabel.textContent = `${val1} + ${val2} = ${sum}`;

});

btnMul.addEventListener("click", function (event) {

const { val1, val2 } = parseInputs();

if (isNaN(val1) || isNaN(val2)) {

resultLabel.textContent =

"Please enter valid numbers in both fields.";

return;

}

const product = val1 \* val2;

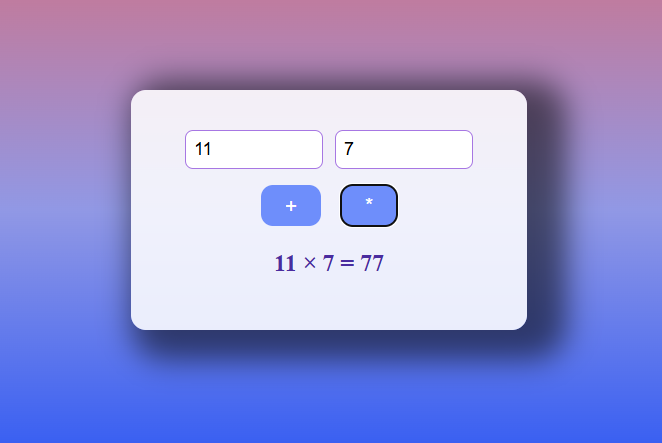
resultLabel.textContent = `${val1} × ${val2} = ${product}`;

});

</script>

</body>

</html>

* **Output:**

# **PRACTICAL – 2**

**AIM: 2.1. Design an user interface for assigning a grade to students based on the subjects marks**

**2.2. Design an User interface for printing the numbers in a) Ascending order b) Descending order c) Subtraction**

* **Code:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

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

<title>Grade</title>

<style>

\* {

box-sizing: border-box;

padding: 0;

margin: 0;

font-family: "Times New Roman", Times, serif;

}

html {

width: 100%;

height: 100%;

background: linear-gradient(#e66465, #9198e5, rgb(1, 59, 250));

background-repeat: no-repeat;

}

#box {

display: flex;

justify-content: center;

align-items: center;

align-content: center;

flex-direction: column;

width: 40%;

height: 60%;

background-color: white;

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

border: none;

z-index: 999;

border-radius: 18px;

background-color: rgba(255, 255, 255, 0.479);

box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);

backdrop-filter: blur(10px);

}

input[type="text"] {

width: 200px;

text-align: center;

height: 25px;

padding: 0.5rem;

margin: 0.5rem 0.25rem;

font-size: 1.1rem;

border-radius: 8px;

border: 1.5px solid #a777e3;

outline-color: #6e8efb;

}

button {

padding: 0.55rem 1.5rem;

margin: 0.5rem 0.5rem 1rem 0.5rem;

font-size: 1.3rem;

border: none;

border-radius: 12px;

background-color: #6e8efb;

color: white;

cursor: pointer;

transition: background-color 0.3s ease;

}

button:hover {

background-color: #a777e3;

}

#result {

display: block;

font-size: 1.5rem;

font-weight: 600;

color: #4a2c9f;

min-height: 2em;

margin-top: 0.5rem;

}

.header\_text {

color: black;

font-size: 18px;

font-weight: 600;

margin: 10px 0px;

}

</style>

</head>

<body>

<div id="box">

<label for="marks" class="header\_text"> Enter Marks to see your grade</label>

<input type="text" id="marks" name="marks" placeholder="Enter Grade" />

<label id="result"></label>

<label for="num" class="header\_text"> Enter numbers in comma seprated to see Ascending order and Descending order</label>

<input type="text" id="num" name="num" placeholder="Enter number" />

<label id="result2"></label>

<label id="result3"></label>

<label for="num2" class="header\_text"> Enter Two numbers to perform subtraction </label>

<div>

<input type="text" id="num2" name="num2" placeholder="Enter first operand" />

<input type="text" id="num3" name="num3" placeholder="Enter second operand" />

</div>

<label id="result4"></label>

<button id="subtraction">Subtraction</button>

</div>

<script>

let inp\_grade = document.getElementById("marks");

let result = document.getElementById("result");

let inp\_num = document.getElementById("num");

let result2 = document.getElementById("result2");

let result3 = document.getElementById("result3");

let inp\_num2 = document.getElementById("num2");

let inp\_num3 = document.getElementById("num3");

let result4 = document.getElementById("result4");

let subtraction = document.getElementById("subtraction");

subtraction.addEventListener("click", function () {

let num1 = Number(inp\_num2.value);

let num2 = Number(inp\_num3.value);

if (isNaN(num1) || isNaN(num2)) { result4.textContent = "Please enter valid numbers."; }

else { result4.textContent = "Subtraction Result: " + (num1 - num2); }

});

inp\_num.addEventListener("input", function () {

let numbers = inp\_num.value.split(",").map(Number);

if (numbers.some(isNaN)) {

result2.textContent = "Please enter valid numbers.";

result3.textContent = "";

} else {

let asc = [...numbers].sort((a, b) => a - b);

let desc = [...numbers].sort((a, b) => b - a);

result2.textContent = "Ascending Order: " + asc.join(", ");

result3.textContent = "Descending Order: " + desc.join(", ");

}

});

inp\_grade.addEventListener("input", function () {

let grade = Number(inp\_grade.value);

if (grade >= 80 && grade <= 100) {

result.textContent = "Grade is A";

} else if (grade >= 70 && grade < 80) {

result.textContent = "Grade is B";

} else if (grade >= 60 && grade < 70) {

result.textContent = "Grade is C";

} else if (grade >= 50 && grade < 60) {

result.textContent = "Grade is D";

} else if (grade >= 0 && grade < 50) {

result.textContent = "Grade is F";

} else {

result.textContent = "";

}

});

</script>

</body>

</html>

* A screenshot of a computer

  AI-generated content may be incorrect.**Output:**

# **PRACTICAL – 3**

**AIM: 3.1 Design an user interface for calculator**

**3.2 Design a user interface for registration of a student for admissions.**

* **Code of 3.1:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Calculator</title>

</head>

<style>

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

html, body {

width: 100%;

height: 100%;

display: flex;

justify-content: center;

align-items: center;

background-color: #f7f7f7;

font-family: Arial, sans-serif;

background: linear-gradient(whitesmoke, rgb(196, 219, 226), lightblue, rgb(140, 186, 201));

background-repeat: no-repeat;

}

table {

width: 300px;

height: 400px;

background-color: #333;

border-radius: 10px;

overflow: hidden;

box-shadow: 0 4px 15px rgba(0, 0, 0, 0.2);

}

#result {

background-color: #222;

font-size: 32px;

font-weight: 800;

color: white;

text-align: right;

padding-right: 10px;

border-bottom: 1px solid #444;

}

#result td {

width: 100%;

}

#textbox {

display: block;

width: 100%;

}

button {

width: 50px;

height: 50px;

border-radius: 50%;

font-size: 18px;

border: none;

color: white;

cursor: pointer;

transition: background-color 0.3s;

}

.btns td {

width: 25%;

}

#zero {

width: 110px;

border-radius: 25px;

}

#clear {

background-color: #ff9500;

}

#abs, #mod, #divi, #mul, #subs, #adds {

background-color: #ff9500;

}

#exe {

background-color: #ff9500;

}

button:hover {

background-color: #ffbf47;

}

.nums {

background-color: #505050;

}

</style>

<body>

<table>

<tr id="result" name="result">

<td colspan="4"> <label for="result" id="textbox">0</label></td>

</tr>

<tr class="btns">

<td align="center"><button type="button"

id="clear">AC</button></td>

<td align="center"><button type="button"

id="abs">+/-</button></td>

<td align="center"><button type="button" id="mod"

class="nums">%</button></td>

<td align="center"><button type="button" id="divi"

class="nums">&divide;</button></td>

</tr>

<tr class="btns">

<td align="center"><button class="nums"

type="button">7</button></td>

<td align="center"><button class="nums"

type="button">8</button></td>

<td align="center"><button class="nums"

type="button">9</button></td>

<td align="center"><button type="button" id="mul"

class="nums">x</button></td>

</tr>

<tr class="btns">

<td align="center"><button class="nums"

type="button">4</button></td>

<td align="center"><button class="nums"

type="button">5</button></td>

<td align="center"><button class="nums"

type="button">6</button></td>

<td align="center"><button type="button" id="subs"

class="nums">-</button></td>

</tr>

<tr class="btns">

<td align="center"><button class="nums"

type="button">1</button></td>

<td align="center"><button class="nums"

type="button">2</button></td>

<td align="center"><button class="nums"

type="button">3</button></td>

<td align="center"><button type="button" id="adds"

class="nums">+</button></td>

</tr>

<tr class="btns">

<td colspan="2" align="center"><button type="button" id="zero"

class="nums">0</button></td>

<td align="center"><button type="button"

class="nums">.</button></td>

<td align="center"><button type="button"

id="exe">=</button></td>

</tr>

</table>

</body>

<script>

const buttons = document.querySelectorAll('button.nums');

const result = document.getElementById('textbox');

let firstValue = null;

let operator = null;

let isOperatorPressed = false;

buttons.forEach(function(button) {

button.addEventListener('click', function() {

if (result.innerText === '0' || isOperatorPressed) {

result.innerText = this.innerText;

isOperatorPressed = false;

} else {

result.innerText += this.innerText;

}

});

});

document.getElementById('clear').addEventListener('click', () => {

result.innerText = '0';

firstValue = null;

operator = null;

isOperatorPressed = false;

});

document.getElementById('abs').addEventListener('click', () => {

let currentValue = result.innerText;

if (currentValue.charAt(0) === '-') {

result.innerText = currentValue.slice(1);

} else {

result.innerText = '-' + currentValue;

}

});

document.querySelectorAll('#adds, #subs, #mul, #divi, #mod').forEach(function(button) {

button.addEventListener('click', function() {

if (result.innerText !== '') {

if (firstValue !== null && operator !== null) {

const secondValue = parseFloat(result.innerText);

let computation;

switch (operator) {

case '+':

computation = firstValue + secondValue;

break;

case '-':

computation = firstValue - secondValue;

break;

case '\*':

computation = firstValue \* secondValue;

break;

case '/':

computation = firstValue / secondValue;

break;

case '%':

computation = firstValue % secondValue;

break;

}

result.innerText = computation.toString();

firstValue = computation;

} else {

firstValue = parseFloat(result.innerText);

}

operator = this.innerText === 'x' ? '\*' : (this.innerText === '÷' ? '/' : this.innerText);

isOperatorPressed = true;

}

});

});

document.getElementById('exe').addEventListener('click', () => {

if (firstValue !== null && operator !== null) {

const secondValue = parseFloat(result.innerText);

let computation;

switch (operator) {

case '+':

computation = firstValue + secondValue;

break;

case '-':

computation = firstValue - secondValue;

break;

case '\*':

computation = firstValue \* secondValue;

break;

case '/':

computation = firstValue / secondValue;

break;

case '%':

computation = firstValue % secondValue;

break;

default:

computation = secondValue;

}

result.innerText = computation.toString();

firstValue = computation;

operator = null;

isOperatorPressed = false;

}

});

</script>

</html>

* **A calculator with orange and black buttons

  AI-generated content may be incorrect.Output:**
* **Code of 3.2:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Registration Form</title>

<style>

html {

width: 100%;

height: 100%;

background: linear-gradient( rgba(173, 216, 230, 0.800), rgba(173, 216, 230, 0.700), rgba(173, 216, 230, 0.600), rgba(173, 216, 230, 0.500));

background-repeat: no-repeat;

}

form {

display: flex;

justify-content: center;

align-items: center;

height: 70%;

width: 35%;

font-family: Arial, sans-serif;

background-color: whitesmoke;

box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);

backdrop-filter: blur(10px);

border-radius: 10px;

padding: 20px;

box-sizing: border-box;

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

}

input[type="text"], input[type="email"], input[type="tel"]{

width: 200px;

height: 25px;

text-align: center;

padding: 0.5rem;

margin: 0.5rem 0.25rem;

font-size: 1.1rem;

border-radius: 8px;

border: 1.5px solid #a777e3;

outline-color: #6e8efb;

}

select{

width: 217px;

margin: 0.5rem 0.25rem;

padding: 0.5rem;

font-size: 1.1rem;

border-radius: 8px;

border: 1.5px solid #a777e3;

outline-color: #6e8efb;

}

button {

padding: 0.55rem 1.5rem;

margin: 0.5rem 0.5rem 1rem 0.5rem;

font-size: 1.3rem;

border: none;

border-radius: 12px;

background-color: #6e8efb;

color: white;

cursor: pointer;

transition: background-color 0.3s ease;

}

button:hover {

background-color: #a777e3;

}

</style>

</head>

<body>

<form action="#" enctype="multipart/form-data">

<table align="center" width="400px" height="400px">

<caption> <h2>Admission Form</h2> </caption>

<tr>

<td><label for="name">Full Name:</label></td>

<td><input type="text" name="name" id="name" placeholder="ex John Marston"></td>

</tr>

<tr>

<td><label for="marks">Marks(Diploma/12th):</label></td>

<td><input type="text" name="marks" id="marks" placeholder="CGPA/percentage"></td>

</tr>

<tr>

<td><label for="email">Email:</label></td>

<td><input type="email" name="email" id="email" placeholder="example@gmail.com"></td>

</tr>

<tr>

<td><label for="gender">Gender:</label></td>

<td>

<input type="radio" name="gender" id="male" value="male">

<label for="male">Male</label>

<input type="radio" name="gender" id="female" value="female">

<label for="female">Female</label>

</td>

</tr>

<tr>

<td><label for="city">City:</label></td>

<td>

<select name="city" id="city">

<option value="new\_york">New York</option>

<option value="los\_angeles">Los Angeles</option>

<option value="chicago">Chicago</option>

<option value="houston">Houston</option>

<option value="phoenix">Tahiti</option>

</select>

</td>

</tr>

<tr>

<td><label for="phone">Personal Phone Number:</label></td>

<td><input type="tel" name="phone" id="phone" placeholder="+91 123456789"></td>

</tr>

<tr>

<td><label for="phone">Parent's Phone Number:</label></td>

<td><input type="tel" name="phone" id="phone" placeholder="+91 123456789"></td>

</tr>

<tr>

<td colspan="2" align="center">

<button type="submit">Submit</button>

<button type="reset">Reset</button>

</td>

</tr>

</table>

</form>

</body>

</html>

* A screenshot of a registration form

  AI-generated content may be incorrect.**Output:**

# **PRACTICAL – 4**

**AIM: 4.1 Design an user interface for semester registration**

**4.2 Design an user interface for displaying and changing of picture on the form**

**4.3 Design an user interface for To count the number of digits in a given number**

* **Code of 4.1:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Semester Form</title>

<style>

html {

width: 100%;

height: 100%;

background: linear-gradient( rgba(173, 216, 230, 0.800), rgba(173, 216, 230, 0.700), rgba(173, 216, 230, 0.600), rgba(173, 216, 230, 0.500));

background-repeat: no-repeat;

}

form {

display: flex;

justify-content: center;

align-items: center;

height: 70%;

width: 35%;

font-family: Arial, sans-serif;

background-color: whitesmoke;

box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);

backdrop-filter: blur(10px);

border-radius: 10px;

padding: 20px;

box-sizing: border-box;

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

}

input[type="text"], input[type="email"], input[type="tel"]{

width: 200px;

height: 25px;

text-align: center;

padding: 0.5rem;

margin: 0.5rem 0.25rem;

font-size: 1.1rem;

border-radius: 8px;

border: 1.5px solid #a777e3;

outline-color: #6e8efb;

}

select{

width: 217px;

margin: 0.5rem 0.25rem;

padding: 0.5rem;

font-size: 1.1rem;

border-radius: 8px;

border: 1.5px solid #a777e3;

outline-color: #6e8efb;

}

button {

padding: 0.55rem 1.5rem;

margin: 0.5rem 0.5rem 1rem 0.5rem;

font-size: 1.3rem;

border: none;

border-radius: 12px;

background-color: #6e8efb;

color: white;

cursor: pointer;

transition: background-color 0.3s ease;

}

button:hover { background-color: #a777e3; }

</style>

</head>

<body>

<form action="#" enctype="multipart/form-data">

<table align="center" width="400px" height="400px">

<caption> <h2>Semester Form</h2> </caption>

<tr>

<td><label for="name">Full Name:</label></td>

<td><input type="text" name="name" id="name" placeholder="ex John Marston"></td>

</tr>

<tr>

<td><label for="marks">Enrollment No</label></td>

<td><input type="text" name="eno" id="eno" placeholder="Enrollment no"></td>

</tr>

<tr>

<td><label for="email">Email:</label></td>

<td><input type="email" name="email" id="email" placeholder="example@gmail.com"></td>

</tr>

<tr>

<td><label for="selective\_sub\_1">Elective subject 1</label></td>

<td>

<input type="radio" name="sub1" id="DS" value="DS">

<label for="DS">Data Structure</label>

<input type="radio" name="sub1" id="JAVA" value="JAVA">

<label for="JAVA">JAVA</label>

</td>

</tr>

<tr>

<td><label for="sub2">Elective subject 2</label></td>

<td>

<select name="sub2" id="sub2">

<option value="C">C</option>

<option value="Android">Android</option>

<option value="PHP">PHP</option>

<option value="python">Python</option>

</select>

</td>

</tr>

<tr>

<td><label for="phone">Phone Number:</label></td>

<td><input type="tel" name="phone" id="phone" placeholder="+91 123456789"></td>

</tr>

<tr>

<td colspan="2" align="center">

<button type="submit">Submit</button>

<button type="reset">Reset</button>

</td>

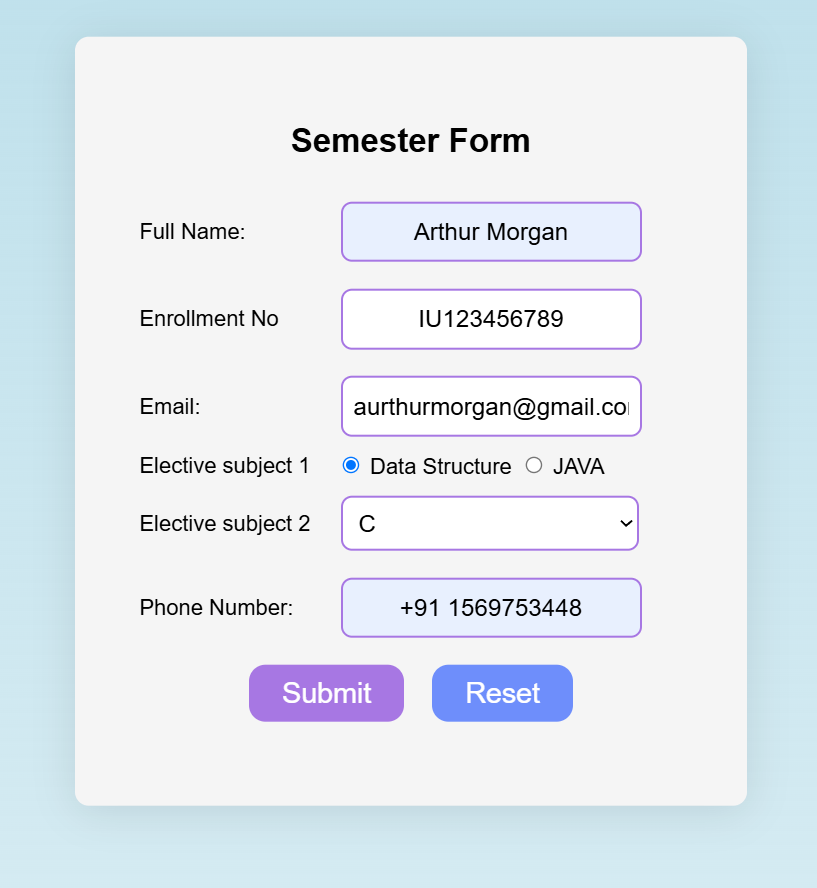
</tr>

</table>

</form>

</body>

</html>

* **Output:**
* **Code of 4.2:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title> Images </title>

<style>

html {

width: 100%;

height: 100%;

background: linear-gradient( rgba(173, 216, 230, 0.800), rgba(173, 216, 230, 0.700), rgba(173, 216, 230, 0.600), rgba(173, 216, 230, 0.500));

background-repeat: no-repeat;

}

form {

height: 80%;

width: 90%;

padding: 10px;

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

font-family: Arial, sans-serif;

background-color: whitesmoke;

box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);

backdrop-filter: blur(10px);

border-radius: 10px;

box-sizing: border-box;

}

button {

padding: 0.45rem 1.1rem;

font-size: 1rem;

border: none;

border-radius: 12px;

background-color: #6e8efb;

color: white;

cursor: pointer;

transition: background-color 0.3s ease;

}

button:hover { background-color: #a777e3; }

#header\_text {

margin-bottom: 20px;

font-size: 2rem;

color: #333;

text-align: center;

font-weight: 900;

text-transform: uppercase;

}

.grid-container {

display: grid;

grid-template-columns: repeat(4, 1fr);

gap: 10px;

padding: 10px;

}

.box {

padding: 20px;

background-color: #f0f0f0;

border: 1px solid #ccc;

border-radius: 8px;

text-align: center;

font-size: 1.5rem;

color: #333;

}

</style>

</head>

<body>

<form action="#" enctype="multipart/form-data">

<p id="header\_text"> Images </p>

<div class="grid-container">

<div class="box">

<button class="img-btn" id="btn1">Select Image</button>

<input type="file" id="file1" style="display:none" accept="image/\*">

</div>

<div class="box">

<button class="img-btn" id="btn2">Select Image</button>

<input type="file" id="file2" style="display:none" accept="image/\*">

</div>

<div class="box">

<button class="img-btn" id="btn3">Select Image</button>

<input type="file" id="file3" style="display:none" accept="image/\*">

</div>

<div class="box">

<button class="img-btn" id="btn4">Select Image</button>

<input type="file" id="file4" style="display:none" accept="image/\*">

</div>

<div class="box">

<button class="img-btn" id="btn5">Select Image</button>

<input type="file" id="file5" style="display:none" accept="image/\*">

</div>

<div class="box">

<button class="img-btn" id="btn6">Select Image</button>

<input type="file" id="file6" style="display:none" accept="image/\*">

</div>

<div class="box">

<button class="img-btn" id="btn7">Select Image</button>

<input type="file" id="file7" style="display:none" accept="image/\*">

</div>

<div class="box">

<button class="img-btn" id="btn8">Select Image</button>

<input type="file" id="file8" style="display:none" accept="image/\*">

</div>

</div>

</form>

<script>

for (let i = 1; i <= 8; i++) {

const btn = document.getElementById(`btn${i}`);

const fileInput = document.getElementById(`file${i}`);

const box = btn.parentElement;

btn.addEventListener('click', (e) => {

e.preventDefault();

fileInput.click();

});

fileInput.addEventListener('change', () => {

const file = fileInput.files[0];

if (file && file.type.startsWith("image/")) {

const reader = new FileReader();

reader.onload = function(e) {

const existingImg = box.querySelector("img");

if (existingImg) existingImg.remove();

const img = document.createElement("img");

img.src = e.target.result;

img.style.maxWidth = "100%";

img.style.maxHeight = "150px";

img.style.marginTop = "10px";

box.appendChild(img);

btn.textContent = "Change Image";

};

reader.readAsDataURL(file);

}

});

}

</script>

</body>

</html>

* A screenshot of a computer

  AI-generated content may be incorrect.**Output:**
* **Code of 4.3:**

<!DOCTYPE html>

<html lang="en">

<head>

<title>Count Digits</title>

<style>

html {

width: 100%;

height: 100%;

background: linear-gradient(rgba(173, 216, 230, 0.800), rgba(173, 216, 230, 0.700), rgba(173, 216, 230, 0.600), rgba(173, 216, 230, 0.500));

}

form {

height: 40%;

width: 40%;

padding: 10px;

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

display: flex;

flex-direction: column;

align-items: center;

background-color: whitesmoke;

box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);

backdrop-filter: blur(10px);

border-radius: 10px;

}

input[type="text"] {

width: 200px;

height: 25px;

text-align: center;

padding: 0.5rem;

margin: 0.5rem 0.25rem;

font-size: 1.1rem;

border-radius: 8px;

border: 1.5px solid #a777e3;

outline-color: #6e8efb;

}

#header\_text {

margin-bottom: 20px;

font-size: 2rem;

color: #333;

text-align: center;

font-weight: 900;

text-transform: uppercase;

}

#result {

margin-top: 20px;

font-size: 1.3rem;

color: #333;

}

</style>

</head>

<body>

<form action="#" onsubmit="return false;">

<p id="header\_text">Enter a Number</p>

<input type="text" id="num" name="num" placeholder="Enter number" />

<label id="result"></label>

</form>

<script>

const numInput = document.getElementById('num');

const resultLabel = document.getElementById('result');

numInput.addEventListener('input', function () {

const value = numInput.value;

const digitMatches = value.match(/\d/g);

if (value === '') { resultLabel.textContent = ''; }

else {

const count = digitMatches ? digitMatches.length : 0;

resultLabel.textContent = `The number of digits is: ${count}`;

}

});

</script>

</body>

</html>

* **A screenshot of a computer

  AI-generated content may be incorrect.Output:**

# **PRACTICAL – 5**

**AIM: 5.1 Design an user interface for simple sort program**

**5.2 Design an User interface User interest form, Making suggestion form**

**5.3 Design an user interface to check whether the year is leap year or not**

**5.4 Design an user interface for menu based Program**

* **Code of 5.1:**

<!DOCTYPE html>

<html lang="en">

<head>

<title>Count number</title>

<style>

html {

width: 100%;

height: 100%;

background: linear-gradient(rgba(173, 216, 230, 0.8), rgba(173, 216, 230, 0.7), rgba(173, 216, 230, 0.6), rgba(173, 216, 230, 0.5));

background-repeat: no-repeat;

}

form {

height: 50%;

width: 45%;

display: flex;

flex-direction: column;

justify-content: center;

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

align-items: center;

padding: 20px;

font-family: Arial, sans-serif;

background-color: whitesmoke;

box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);

backdrop-filter: blur(10px);

border-radius: 10px;

box-sizing: border-box;

}

input[type="text"] {

width: 200px;

height: 25px;

text-align: center;

padding: 0.5rem;

margin: 0.5rem 0.25rem;

font-size: 1.1rem;

border-radius: 8px;

border: 1.5px solid #a777e3;

outline-color: #6e8efb;

}

.header\_text {

font-size: 1.2rem;

font-weight: bold;

margin-bottom: 10px;

text-align: center;

}

</style>

</head>

<body>

<form action="#" enctype="multipart/form-data">

<label for="num" class="header\_text">

Enter numbers in comma seprated to see Ascending order and Descending order</label>

<input type="text" id="num" name="num" placeholder="Enter number" />

<label id="result2"></label>

<label id="result3"></label>

</form>

<script>

const num = document.getElementById("num");

const result2 = document.getElementById("result2");

const result3 = document.getElementById("result3");

num.addEventListener("input", () => {

const values = num.value.split(",").map(Number);

const ascending = [...values].sort((a, b) => a - b);

const descending = [...values].sort((a, b) => b - a);

result2.textContent = "Ascending: " + ascending.join(", ");

result3.textContent = "Descending: " + descending.join(", ");

});

</script>

</body>

</html>

* A screenshot of a computer

  AI-generated content may be incorrect.**Output:**
* **Code of 5.2: <!DOCTYPE html>**

<html lang="en">

<head>

<meta charset="UTF-8" />

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

<title>Contact Us Form</title>

<style>

html {

width: 100%;

height: 100%;

background: linear-gradient(rgba(173, 216, 230, 0.8), rgba(173, 216, 230, 0.7), rgba(173, 216, 230, 0.6), rgba(173, 216, 230, 0.5));

background-repeat: no-repeat;

}

form {

display: flex;

justify-content: center;

align-items: center;

height: 70%;

width: 35%;

font-family: Arial, sans-serif;

background-color: whitesmoke;

box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);

backdrop-filter: blur(10px);

border-radius: 10px;

padding: 20px;

box-sizing: border-box;

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

}

input[type="text"], input[type="email"], input[type="tel"], textarea {

width: 200px;

height: 25px;

text-align: center;

padding: 0.5rem;

margin: 0.5rem 0.25rem;

font-size: 1.1rem;

border-radius: 8px;

border: 1.5px solid #a777e3;

outline-color: #6e8efb;

}

select {

width: 217px;

margin: 0.5rem 0.25rem;

padding: 0.5rem;

font-size: 1.1rem;

border-radius: 8px;

border: 1.5px solid #a777e3;

outline-color: #6e8efb;

}

button {

padding: 0.55rem 1.5rem;

margin: 0.5rem 0.5rem 1rem 0.5rem;

font-size: 1.3rem;

border: none;

border-radius: 12px;

background-color: #6e8efb;

color: white;

cursor: pointer;

transition: background-color 0.3s ease;

}

button:hover { background-color: #a777e3; }

</style>

</head>

<body>

<form action="#" enctype="multipart/form-data">

<table align="center" width="400px" height="400px">

<caption>

<h2>Contact US</h2>

</caption>

<tr>

<td><label for="name">Full Name:</label></td>

<td>input type="text" name="name" id="name" placeholder="ex Arthur Morgan"/><td>

</tr>

<tr>

<td><label for="email">Email:</label></td>

<td>

<input type="email" name="email" id="email" placeholder="example@gmail.com"/>

</td>

</tr>

<tr>

<td><label for="phone">Phone Number:</label></td>

<td>

<input type="tel" name="phone" id="phone" placeholder="+91 123456789"/>

</td>

</tr>

<tr>

<td><label for="Interests">Interests:</label></td>

</tr>

<tr>

<td colspan="2" style="width:100%; ">

<textarea wrap="soft" name="Interests" id="Interests" placeholder="Your Interests" style="width:100%;"></textarea>

</td>

</tr>

<tr>

<td colspan="2" align="center">

<button type="submit">Submit</button>

<button type="reset">Reset</button>

</td>

</tr>

</table>

</form>

</body>

</html>

* A screenshot of a contact us form

  AI-generated content may be incorrect.**Output:**
* **Code of 5.3:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

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

<title>Leap Year Checker</title>

<style>

html {

width: 100%;

height: 100%;

background: linear-gradient(rgba(173, 216, 230, 0.8),rgba(173, 216, 230, 0.7),rgba(173, 216, 230, 0.6),rgba(173, 216, 230, 0.5));

background-repeat: no-repeat;

}

form {

height: 40%;

width: 35%;

display: flex;

flex-direction: column;

justify-content: center;

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

align-items: center;

padding: 20px;

font-family: Arial, sans-serif;

background-color: whitesmoke;

box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);

backdrop-filter: blur(10px);

border-radius: 10px;

box-sizing: border-box;

}

input[type="text"] {

width: 200px;

height: 25px;

text-align: center;

padding: 0.5rem;

margin: 0.5rem 0.25rem;

font-size: 1.1rem;

border-radius: 8px;

border: 1.5px solid #a777e3;

outline-color: #6e8efb;

}

.header\_text {

font-size: 1.2rem;

font-weight: bold;

margin-bottom: 10px;

text-align: center;

}

</style>

</head>

<body>

<form action="#" enctype="multipart/form-data">

<label for="num" class="header\_text">

Enter year to Check Leap Year</label>

<input type="text" id="num" name="num" placeholder="Enter year" />

<label id="result2"></label>

</form>

<script>

const num = document.getElementById("num");

const result2 = document.getElementById("result2");

num.addEventListener("input", () => {

const year = parseInt(num.value);

if (isNaN(year)) {

result2.textContent = "Please enter a valid year.";

return;

}

if ((year % 4 === 0 && year % 100 !== 0) || (year % 400 === 0)) {

result2.textContent = year + " is a leap year.";

} else {

result2.textContent = year + " is not a leap year.";

}

});

</script>

</body>

</html>

* **A screenshot of a computer

  AI-generated content may be incorrect.Output:**
* **Code of 5.4:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

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

<title>Menu Based Program</title>

<style>

html {

width: 100%;

height: 100%;

background: linear-gradient(rgba(173, 216, 230, 0.8), rgba(173, 216, 230, 0.7), rgba(173, 216, 230, 0.6), rgba(173, 216, 230, 0.5));

background-repeat: no-repeat;

}

body { font-family: Arial, sans-serif; margin: 40px; }

form {

height: 50%;

width: 45%;

display: flex;

flex-direction: column;

justify-content: center;

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

align-items: center;

padding: 20px;

font-family: Arial, sans-serif;

background-color: whitesmoke;

box-shadow: 0 4px 30px rgba(0, 0, 0, 0.1);

backdrop-filter: blur(10px);

border-radius: 10px;

box-sizing: border-box;

}

.header\_text {

font-size: 1.2rem;

font-weight: bold;

margin-bottom: 10px;

text-align: center;

}

.menu { margin-bottom: 20px; display: flex; flex-direction: column; gap: 10px; }

button {

padding: 0.45rem 1.1rem;

font-size: 1rem;

border: none;

border-radius: 12px;

background-color: #6e8efb;

color: white;

cursor: pointer;

transition: background-color 0.3s ease;

}

button:hover { background-color: #a777e3; }

#output { margin-top: 30px; font-weight: bold; }

</style>

</head>

<body>

<form action="#" enctype="multipart/form-data">

<!-- Design an user interface for menu based Program -->

<h2>Menu</h2>

<div class="menu">

<button onclick="showOption(1)">Show Date</button>

<button onclick="showOption(2)">Show Time</button>

<button onclick="showOption(3)">About</button>

<button onclick="showOption(0)">Exit</button>

</div>

<div id="output"></div>

</form>

<script>

function showOption(option) {

const output = document.getElementById('output');

switch(option) {

case 1:

output.textContent = "Today's date is: " + new Date().toLocaleDateString();

break;

case 2:

output.textContent = "Current time is: " + new Date().toLocaleTimeString();

break;

case 3:

output.textContent = "Basic web for Time show";

break;

case 0:

output.textContent = "Thank you!!";

break;

default:

output.textContent = "";

}

}

</script>

</body>

</html>

* **A screenshot of a menu

  AI-generated content may be incorrect.A screenshot of a menu

  AI-generated content may be incorrect.A screenshot of a menu

  AI-generated content may be incorrect.A screenshot of a menu

  AI-generated content may be incorrect.Output:**

# **PRACTICAL – 6**

**AIM: 6 Project: SplayList – A group recommendation Music webapp**

* **Purpose**

The purpose of Splaylist is to facilitate seamless music collaboration among groups by automatically generating a shared playlist that reflects the combined taste of all group members. By integrating with Spotify, Splaylist allows each user to authenticate, fetch their top tracks, and collectively analyze group preferences to create a unique playlist that everyone will enjoy. The project aims to leverage machine learning and data visualization to provide insights about group music compatibility and trends, making group music sessions more engaging and tailored.

* **Scope**

Splaylist is a web-based application that enables users to log in with their Spotify accounts, add or remove others from a collaborative group, and generate playlists that represent the musical taste of the group. The system collects and analyzes users’ top tracks using the Spotify API and applies clustering techniques to identify group-level preferences. Besides playlist generation, Splaylist offers visual analytics (charts, statistics, PDF reports) to help users understand the diversity and overlap in their group’s listening habits. The project is scoped for educational purposes as a mini project and is intended for users with existing Spotify accounts who want to automate and enhance group playlist creation without manual song selection.

* **Technologies Used**
* **Programming Language:**

Python 3.x – The primary language for application logic, data processing, APIs, and machine learning.

* **Web Framework:**

Flask – A lightweight Python web framework used to build the web-based user interface and manage HTTP requests.

* **Third-Party APIs:**

Spotify Web API (accessed via Spotipy) – Used for authenticating users, fetching user data, and managing playlists.

* **Data Science and Machine Learning Libraries:**
  + Pandas – For data cleaning and analysis.
  + NumPy – For numerical computing and data manipulation.
  + scikit-learn – For clustering music data and building recommendation logic.
* **Data Visualization & Reporting:**
  + Matplotlib & Seaborn – For generating analytical charts and graphs.
  + ReportLab – For automating the creation of PDF reports.
* **Other Libraries:**
  + Requests – For HTTP requests (used internally by APIs).
  + python-dotenv – For securely managing environment variables (e.g., API keys).
* **Operating Environment:**
  + Web browser (for user interface)
  + Runs on standard desktop operating systems (Windows/Linux/Mac)
* **Installation & Dependency Management:**
  + requirements.txt – Specifies all Python dependencies for quick setup.
* **Functional Requirements**

**FR1: Authentication Requirements**

* **FR1.1**: The system SHALL authenticate users through Spotify OAuth2 protocol
* **FR1.2**: The system SHALL maintain secure session management with token refresh capabilities
* **FR1.3**: The system SHALL provide logout functionality with complete session and cache cleanup
* **FR1.4**: The system SHALL handle authentication errors gracefully with user-friendly messages
* **FR1.5**: The system SHALL force fresh login dialogs when adding different users

**FR2: Group Management Requirements**

* **FR2.1**: The system SHALL allow users to add multiple group members to a collaborative session
* **FR2.2**: The system SHALL provide individual user removal functionality from active sessions
* **FR2.3**: The system SHALL display current group members with management controls
* **FR2.4**: The system SHALL prevent duplicate user additions with appropriate warnings
* **FR2.5**: The system SHALL provide "Clear All Users" functionality to reset sessions

**FR3: Data Collection and Processing Requirements**

* **FR3.1**: The system SHALL collect user's top 50 tracks data from Spotify API
* **FR3.2**: The system SHALL process and clean music data for analysis
* **FR3.3**: The system SHALL generate pseudo-genres for tracks without genre information
* **FR3.4**: The system SHALL save processed data in CSV format for future analysis
* **FR3.5**: The system SHALL handle missing or incomplete data gracefully

**FR4: Playlist Generation Requirements**

* **FR4.1**: The system SHALL create unified playlists combining all group members' preferences
* **FR4.2**: The system SHALL remove duplicate tracks while preserving order
* **FR4.3**: The system SHALL save generated playlists directly to authenticated user's Spotify account
* **FR4.4**: The system SHALL handle playlist creation errors with fallback mechanisms
* **FR4.5**: The system SHALL provide Spotify links to created playlists

**FR5: Analytics and Visualization Requirements**

* **FR5.1**: The system SHALL generate genre distribution charts
* **FR5.2**: The system SHALL create popularity analysis visualizations
* **FR5.3**: The system SHALL provide user diversity comparison charts
* **FR5.4**: The system SHALL perform statistical analysis (descriptive and inferential)
* **FR5.5**: The system SHALL generate comprehensive PDF reports with embedded charts
* **FR5.6**: The system SHALL allow users to download analysis results

**FR6: Machine Learning Requirements**

* **FR6.1**: The system SHALL apply K-Means clustering to group similar music preferences
* **FR6.2**: The system SHALL automatically determine optimal number of clusters
* **FR6.3**: The system SHALL provide cluster-based recommendations
* **FR6.4**: The system SHALL save and load trained models for consistency
* **Non – Functional Requirements**

**NFR1: Performance Requirements**

* **NFR1.1**: Playlist generation SHALL complete within 30 seconds for groups up to 10 users
* **NFR1.2**: Chart generation SHALL complete within 15 seconds for datasets up to 500 tracks
* **NFR1.3**: User authentication SHALL complete within 10 seconds
* **NFR1.4**: PDF report generation SHALL complete within 60 seconds including all visualizations
* **NFR1.5**: The system SHALL handle concurrent user sessions without performance degradation

**NFR2: Usability Requirements**

* **NFR2.1**: The web interface SHALL be intuitive and require minimal training
* **NFR2.2**: The system SHALL provide clear navigation with consistent design patterns
* **NFR2.3**: Response messages SHALL be informative and guide users through processes
* **NFR2.4**: The interface SHALL be responsive and work across different screen sizes
* **NFR2.5**: Error messages SHALL be user-friendly and provide actionable guidance

**NFR3: Reliability Requirements**

* **NFR3.1**: The system SHALL handle Spotify API failures gracefully without crashing
* **NFR3.2**: Authentication token refresh SHALL occur automatically without user intervention
* **NFR3.3**: Data processing SHALL continue even with partial data availability
* **NFR3.4**: Error recovery SHALL maintain session state whenever possible
* **NFR3.5**: The system SHALL have 95% uptime during active development/demonstration

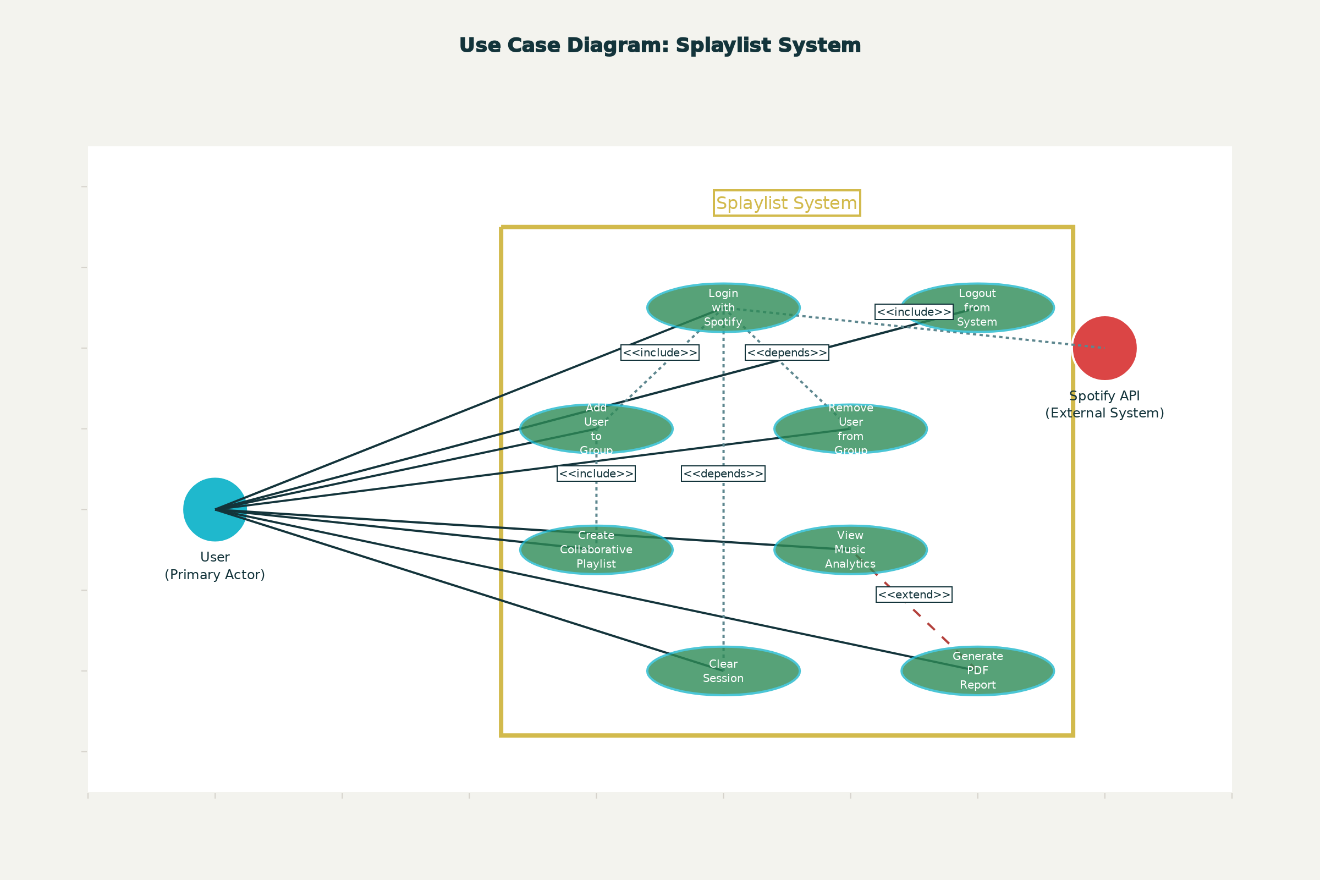
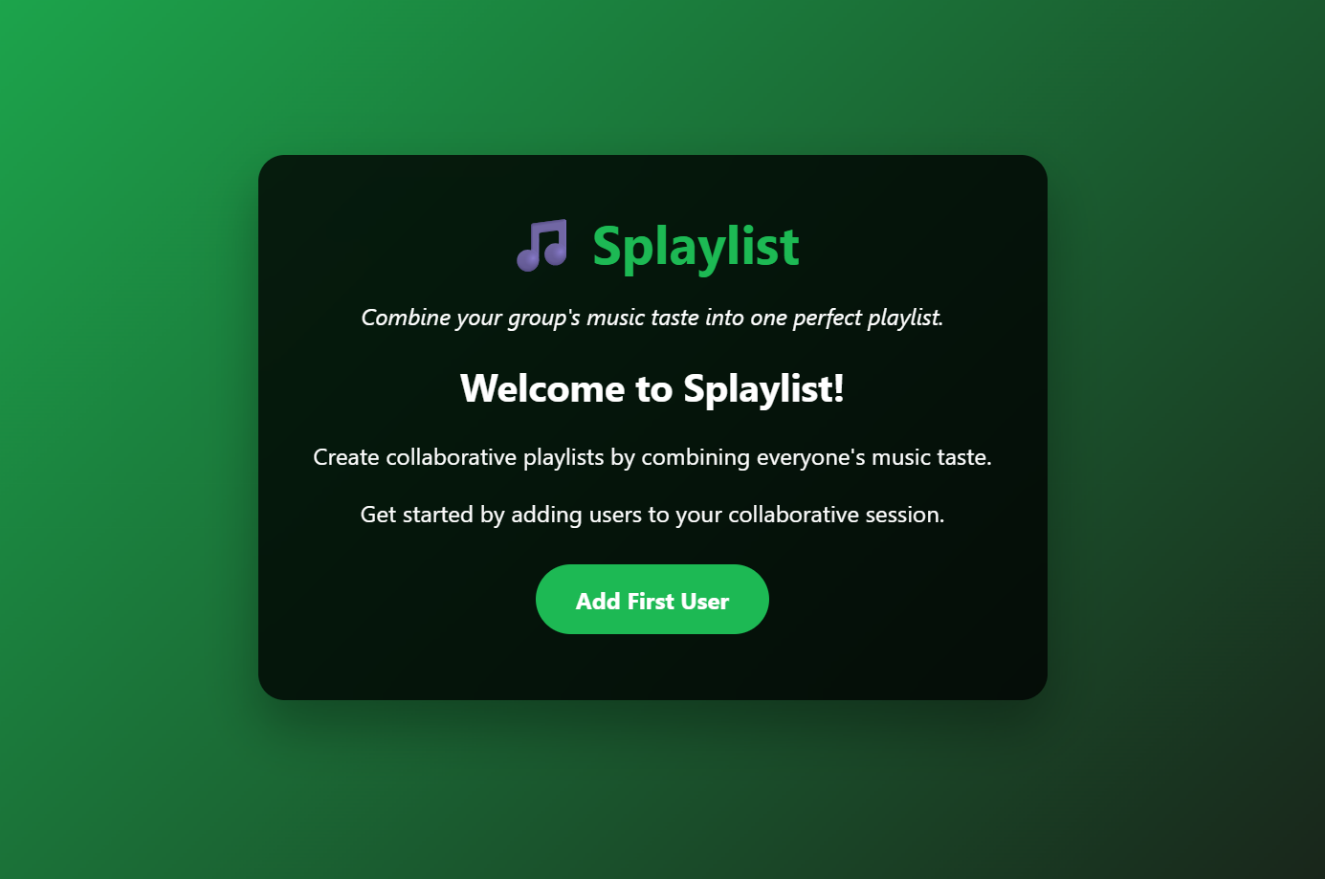
**NFR4: Security Requirements**

* **NFR4.1**: User authentication SHALL use OAuth2 standard security protocols
* **NFR4.2**: API keys and sensitive data SHALL be stored securely using environment variables
* **NFR4.3**: Session data SHALL be properly cleared upon logout or session expiration
* **NFR4.4**: The system SHALL not store user passwords or permanent personal data
* **NFR4.5**: All HTTP communications SHALL be conducted over secure connections

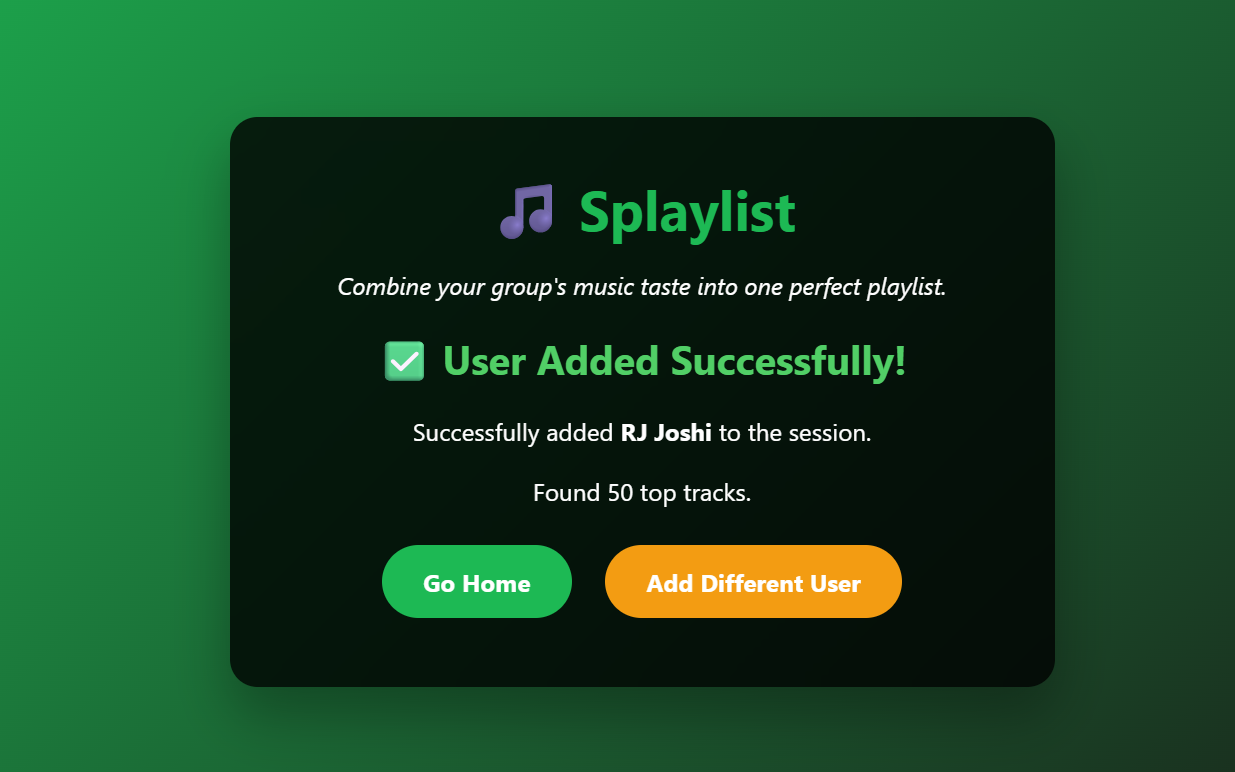
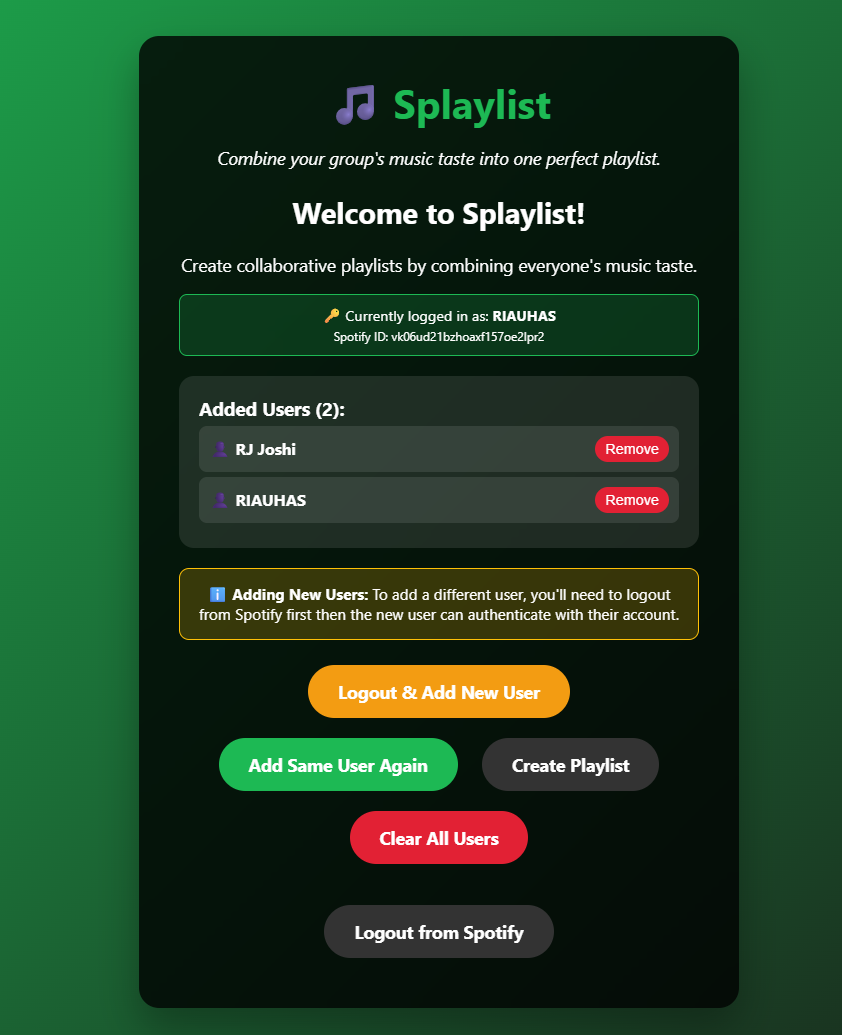
**NFR5: Compatibility Requirements**

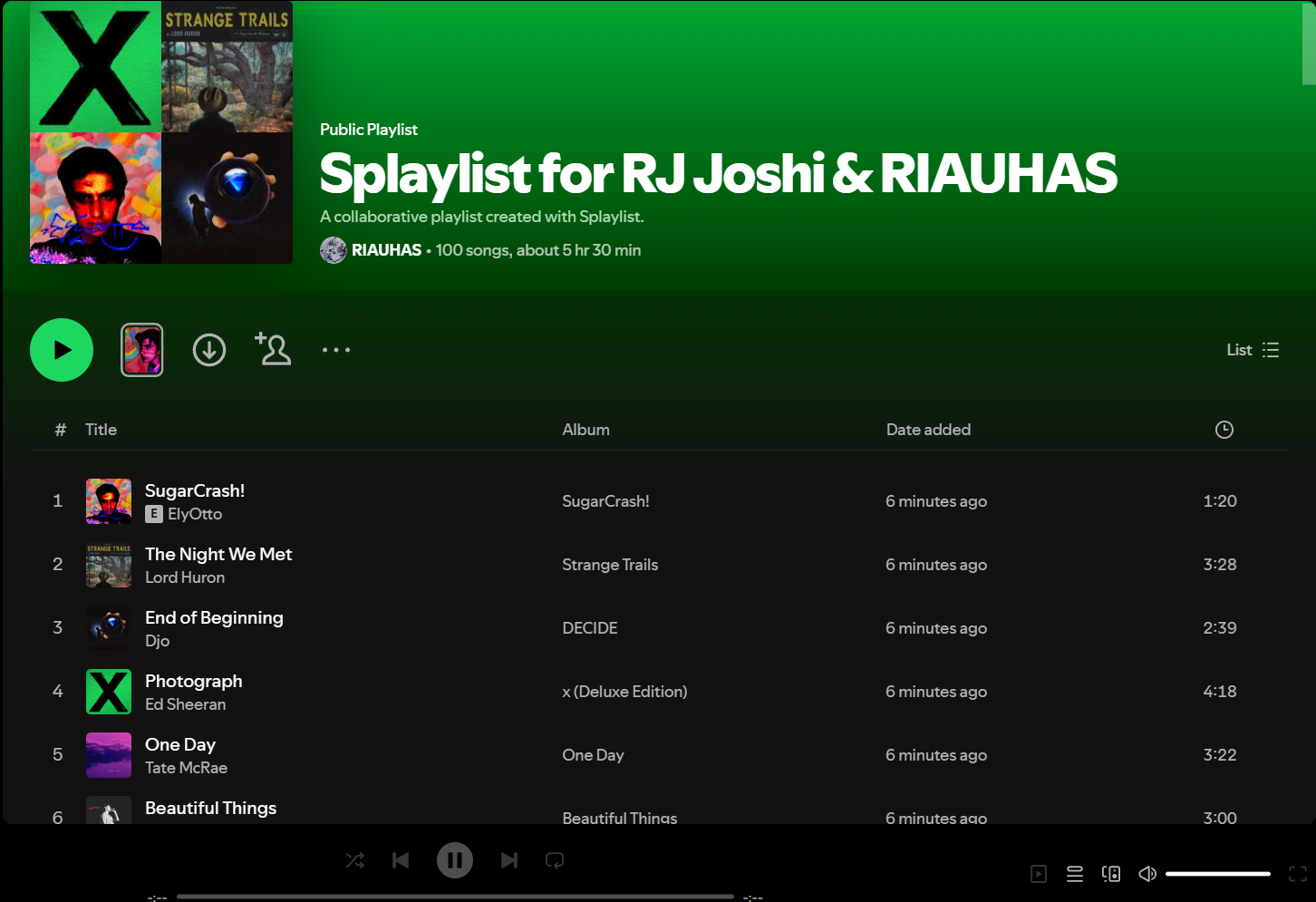
* **NFR5.1**: The system SHALL be compatible with modern web browsers (Chrome, Firefox, Safari, Edge)
* **NFR5.2**: The application SHALL run on Python 3.8+ environments
* **NFR5.3**: Generated files SHALL be compatible with standard PDF viewers and CSV applications
* **NFR5.4**: The system SHALL work across Windows, macOS, and Linux operating systems
* **NFR5.5**: The interface SHALL be mobile-responsive for basic functionality

**NFR6: Scalability Requirements**

* **NFR6.1**: The system SHALL support groups of up to 10 users efficiently
* **NFR6.2**: The system SHALL handle datasets of up to 1000 tracks per analysis
* **NFR6.3**: Memory usage SHALL not exceed 512MB during normal operation
* **NFR6.4**: The system SHALL process multiple visualization requests concurrently
* **System Design of the SplayList**
* Use Case Diagram
* Home Page
* A screenshot of a black screen

  AI-generated content may be incorrect.A screenshot of a black screen

  AI-generated content may be incorrect.Auth Dialog (For Ex. Adding two different users)
* User added Page
* User added Page
* A screenshot of a music playlist

  AI-generated content may be incorrect.Play-list created
* Play list in Spotify