**3. XML and JSON files**

1. \*CSE(AIML).xml\*:

xml

<Departments>

<Department>

<ID>1A</ID>

<name>Computer Science</Name>

<Head>Dr.ram</Head>

</Department>

<Department>

<ID>2A</ID>

<Name>Electrical Engineering</Name>

<Head>Dr.omkar</Head>

</Department>

</Departments>

2. \*Year.xml\*:

xml

<Years>

<Year>

<ID>1st</ID>

<Name>Humanities and sciences</Name>

</Year>

<Year>

<ID>2nd</ID>

<Name>2nd years</Name>

</Year>

<Year>

<ID>3rd</ID>

<Name>3rd years</Name>

</Year>

<Year>

<ID>4th</ID>

<Name>final years</Name>

</Year>

</Years>

3. \*Student.xml\*:

xml

<Students>

<Student>

<ID>6601</ID>

<Name>nikitha</Name>

<DepartmentID>1A</DepartmentID>

<YearID>3rd</YearID>

<GPA>8.0</GPA>

</Student>

<Student>

<ID>6602</ID>

<Name>virat</Name>

<DepartmentID>2A</DepartmentID>

<YearID>3rd</YearID>

<GPA>8.5</GPA>

</Student>

</Students>

### JSON Files:

1. \*Department.json\*:

json

{

"Departments": [

{

"ID": “1A”,

"Name": "Computer Science",

"Head": "Dr.ram"

},

{

"ID": “2A”,

"Name": "Electrical Engineering",

"Head": "Dr. omkar"

}

]

}

2. \*Year.json\*:

json

{

"Years": [

{

"ID": 1,

"Name": "Humanities and sciences"

},

{

"ID": 2,

"Name": "2nd years"

},

{

"ID": 3,

"Name": "3rd years"

},

{

"ID": 4,

"Name": "final years"

}

]

}

3. \*Student.json\*:

json

{

"Students": [

{

"ID":”6601”,

"Name": "nikitha",

"DepartmentID": “1A”,

"YearID": 1,

"GPA": 8.0

},

{

"ID":6602,

"Name": "virat",

"DepartmentID": “2A”,

"YearID": 2,

"GPA": 8.5

}

]

}

**4.Create a file with department as root , year as sub root and student as an element**

<Departments>

<Department>

<ID>1A</ID>

<Name>Computer Science</Name>

<Head>Dr.ram</Head>

<Year>

<ID>1st</ID>

<Name>Humanities and sciences</Name>

<Students>

<Student>

<ID>6601</ID>

<Name>nikitha</Name>

<GPA>8.0</GPA>

</Student>

</Students>

</Year>

<Year>

<ID>2nd</ID>

<Name>2nd years</Name>

<Students>

<!-- No students in this year for Computer Science -->

</Students>

</Year>

<Year>

<ID>3rd</ID>

<Name>3rd years</Name>

<Students>

<Student>

<ID>6601</ID>

<Name>nikitha</Name>

<GPA>8.0</GPA>

</Student>

</Students>

</Year>

<Year>

<ID>4th</ID>

<Name>final years</Name>

<Students>

<!-- No students in this year for Computer Science -->

</Students>

</Year>

</Department>

<Department>

<ID>2A</ID>

<Name>Electrical Engineering</Name>

<Head>Dr.omkar</Head>

<Year>

<ID>1st</ID>

<Name>Humanities and sciences</Name>

<Students>

<!-- No students in this year for Electrical Engineering -->

</Students>

</Year>

<Year>

<ID>2nd</ID>

<Name>2nd years</Name>

<Students>

<!-- No students in this year for Electrical Engineering -->

</Students>

</Year>

<Year>

<ID>3rd</ID>

<Name>3rd years</Name>

<Students>

<Student>

<ID>6602</ID>

<Name>virat</Name>

<GPA>8.5</GPA>

</Student>

</Students>

</Year>

<Year>

<ID>4th</ID>

<Name>final years</Name>

<Students>

<!-- No students in this year for Electrical Engineering -->

</Students>

</Year>

</Department>

</Departments>

**5.Simple login page**

html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Login Page</title>

<style>

body {

font-family: Arial, sans-serif;

background-color: #f4f4f4;

margin: 0;

padding: 0;

}

.login-container {

width: 300px;

margin: 100px auto;

background-color: #fff;

border-radius: 5px;

padding: 20px;

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

}

.login-container h2 {

text-align: center;

margin-bottom: 20px;

}

.form-group {

margin-bottom: 20px;

}

.form-group label {

display: block;

margin-bottom: 5px;

}

.form-group input[type="text"],

.form-group input[type="password"] {

width: 100%;

padding: 10px;

border: 1px solid #ccc;

border-radius: 3px;

}

.form-group button {

width: 100%;

padding: 10px;

background-color: #007bff;

border: none;

color: #fff;

border-radius: 3px;

cursor: pointer;

}

.form-group button:hover {

background-color: #0056b3;

}

</style>

</head>

<body>

<div class="login-container">

<h2>Login</h2>

<form id="login-form" action="your\_backend\_login\_endpoint" method="POST">

<div class="form-group">

<label for="username">Username</label>

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

</div>

<div class="form-group">

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

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

</div>

<div class="form-group">

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

</div>

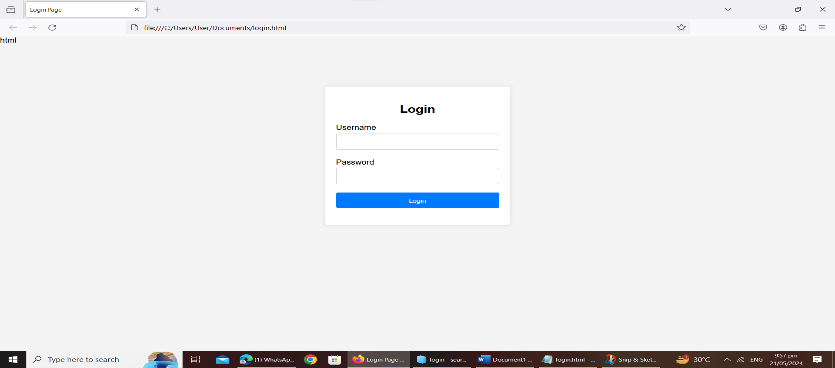
</form>

</div>

</body>

</html>

This HTML code creates a login page with fields for entering a username and password, along with a "Login" button. You can adjust the styling and functionality as needed for your specific requirements. Remember to replace "your\_backend\_login\_endpoint" with the actual endpoint where you handle the login request on your backend server.



**6.create user creation screen by using elements like list,radio button, drop down, check box**

html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>User Creation Screen</title>

<style>

body {

font-family: Arial, sans-serif;

background-color: #f4f4f4;

margin: 0;

padding: 20px;

}

.container {

max-width: 600px;

margin: 50px auto;

background-color: #fff;

padding: 20px;

border-radius: 5px;

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

}

.form-group {

margin-bottom: 20px;

}

.form-group label {

display: block;

margin-bottom: 5px;

font-weight: bold;

}

.form-group input[type="text"],

.form-group input[type="email"],

.form-group select {

width: 100%;

padding: 10px;

border: 1px solid #ccc;

border-radius: 3px;

}

.form-group input[type="radio"],

.form-group input[type="checkbox"] {

margin-right: 5px;

}

.form-group button {

padding: 10px 20px;

background-color: #007bff;

border: none;

color: #fff;

border-radius: 3px;

cursor: pointer;

}

.form-group button:hover {

background-color: #0056b3;

}

</style>

</head>

<body>

<div class="container">

<h2>User Creation Form</h2>

<form id="user-form" action="#" method="POST">

<div class="form-group">

<label for="username">Username</label>

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

</div>

<div class="form-group">

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

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

</div>

<div class="form-group">

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

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

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

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

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

</div>

<div class="form-group">

<label for="country">Country</label>

<select id="country" name="country" required>

<option value="">Select Country</option>

<option value="usa">USA</option>

<option value="uk">UK</option>

<option value="canada">Canada</option>

<option value="australia">Australia</option>

</select>

</div>

<div class="form-group">

<label for="interests">Interests</label>

<input type="checkbox" id="music" name="interests" value="music">

<label for="music">Music</label>

<input type="checkbox" id="sports" name="interests" value="sports">

<label for="sports">Sports</label>

<input type="checkbox" id="movies" name="interests" value="movies">

<label for="movies">Movies</label>

</div>

<div class="form-group">

<button type="submit">Create User</button>

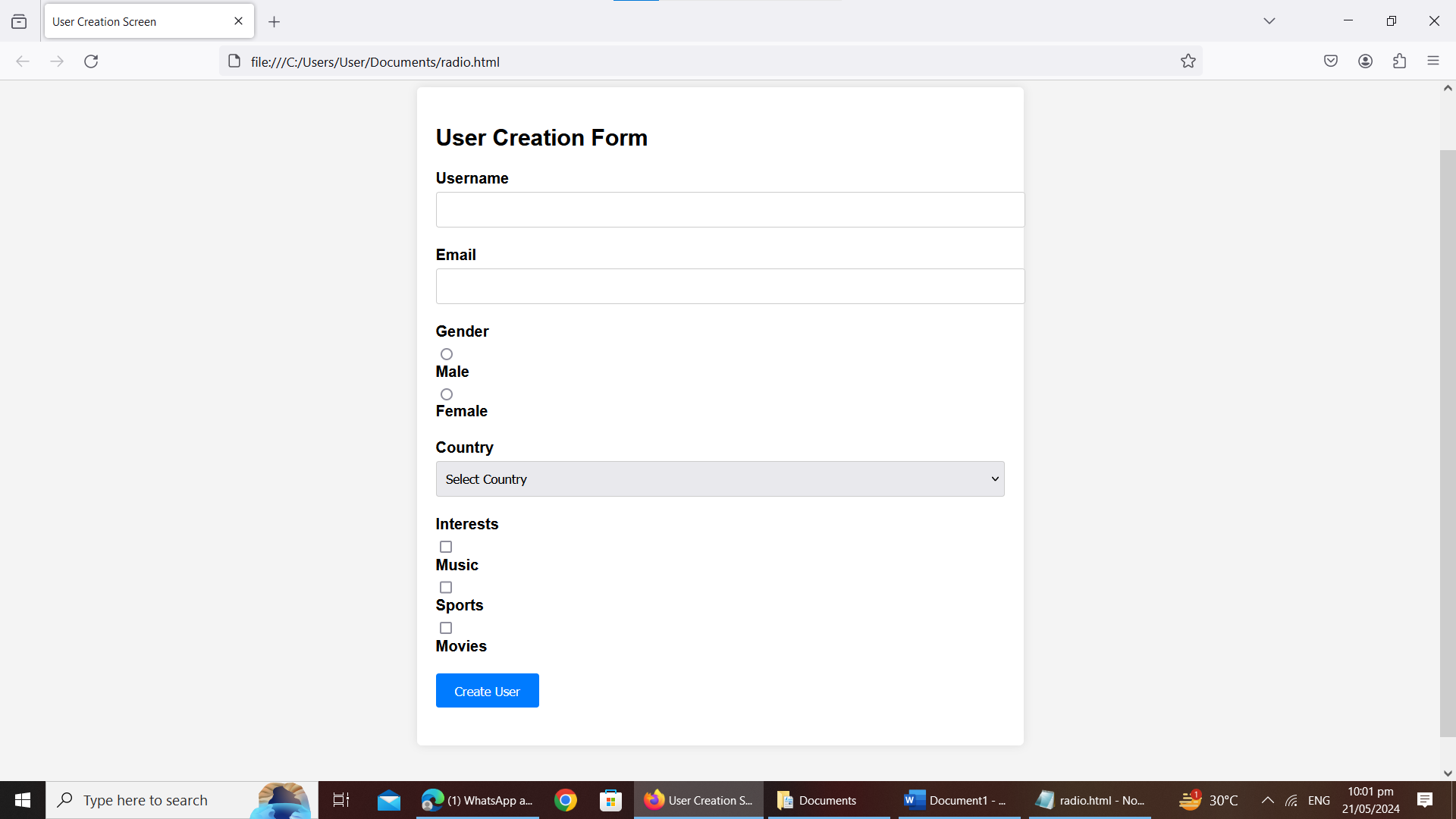
</div>

</form>

</div>

</body>

</html>



**7. list all users ,update user and delete user**

html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>User Management System</title>

<style>

body {

font-family: Arial, sans-serif;

background-color: #f4f4f4;

margin: 0;

padding: 20px;

}

.container {

max-width: 800px;

margin: 20px auto;

background-color: #fff;

border-radius: 5px;

padding: 20px;

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

}

.user-list {

list-style-type: none;

padding: 0;

}

.user-list li {

margin-bottom: 10px;

border-bottom: 1px solid #ccc;

padding-bottom: 10px;

}

.user-list li:last-child {

border-bottom: none;

}

.btn {

padding: 8px 20px;

background-color: #007bff;

border: none;

color: #fff;

border-radius: 3px;

cursor: pointer;

}

.btn:hover {

background-color: #0056b3;

}

.confirm-popup {

display: none;

position: fixed;

top: 50%;

left: 50%;

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

background-color: #fff;

padding: 20px;

border-radius: 5px;

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

}

</style>

</head>

<body>

<div class="container">

<h2>User Management System</h2>

<button class="btn" id="add-user">Add User</button>

<ul class="user-list" id="user-list">

<!-- User list will be populated dynamically -->

</ul>

</div>

<div class="confirm-popup" id="confirm-popup">

<p>Are you sure you want to delete this user?</p>

<button class="btn" id="confirm-delete">Yes</button>

<button class="btn" id="cancel-delete">Cancel</button>

</div>

<script>

// Dummy user data (replace with actual data from backend)

const users = [

{ id: 1, name: "John Doe", email: "john@example.com" },

{ id: 2, name: "Jane Smith", email: "jane@example.com" },

{ id: 3, name: "Alice Johnson", email: "alice@example.com" }

];

// Function to render user list

function renderUserList() {

const userList = document.getElementById("user-list");

userList.innerHTML = "";

users.forEach(user => {

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

li.innerHTML = `

<strong>${user.name}</strong> - ${user.email}

<button class="btn" onclick="editUser(${user.id})">Edit</button>

<button class="btn" onclick="showConfirmPopup(${user.id})">Delete</button>

`;

userList.appendChild(li);

});

}

// Function to edit user

function editUser(userId) {

// Placeholder for editing user functionality

alert("Editing user with ID: " + userId);

}

// Function to show confirmation popup for deleting user

function showConfirmPopup(userId) {

const popup = document.getElementById("confirm-popup");

popup.style.display = "block";

const confirmBtn = document.getElementById("confirm-delete");

const cancelBtn = document.getElementById("cancel-delete");

confirmBtn.onclick = function() {

deleteUser(userId);

popup.style.display = "none";

};

cancelBtn.onclick = function() {

popup.style.display = "none";

};

}

// Function to delete user

function deleteUser(userId) {

// Placeholder for deleting user functionality

alert("Deleting user with ID: " + userId);

// After deletion, you would typically update the user list

// Here, for demonstration, let's just remove the user from the array

const index = users.findIndex(user => user.id === userId);

if (index !== -1) {

users.splice(index, 1);

renderUserList(); // Update the user list on the UI

}

}

// Function to add event listener for adding a user

document.getElementById("add-user").addEventListener("click", function() {

// Placeholder for adding user functionality

alert("Adding a new user");

});

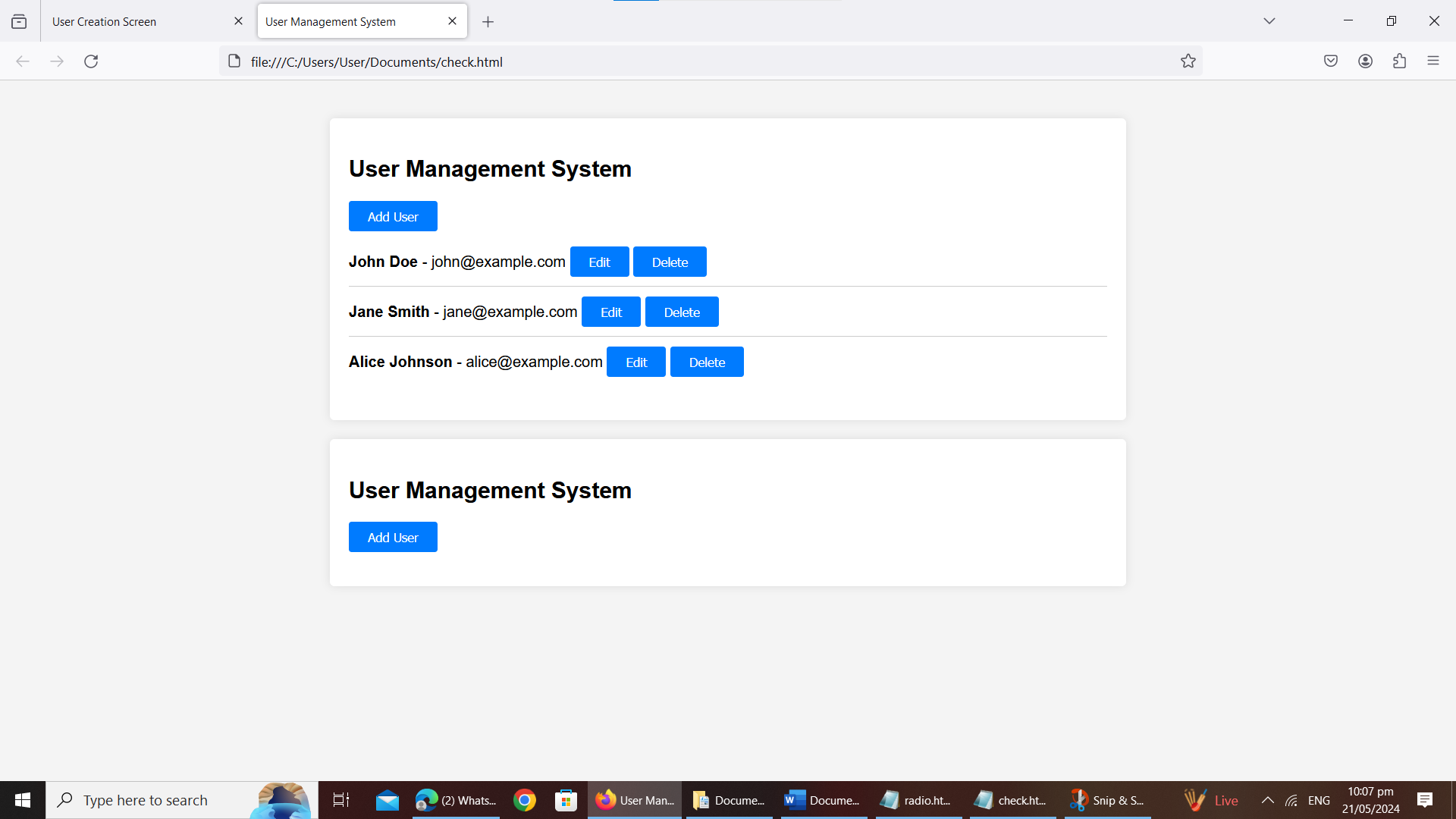
// Initial rendering of user list

renderUserList();

</script>

</body>

</html>



**9.create a html file to upload a file**

html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>File Upload Example</title>

</head>

<body>

<h1>File Upload Example</h1>

<form action="/upload" method="post" enctype="multipart/form-data">

<label for="file">Select a file:</label>

<input type="file" id="file" name="file" accept=".txt, .pdf, .doc, .docx"> <!-- Add or remove file types as needed -->

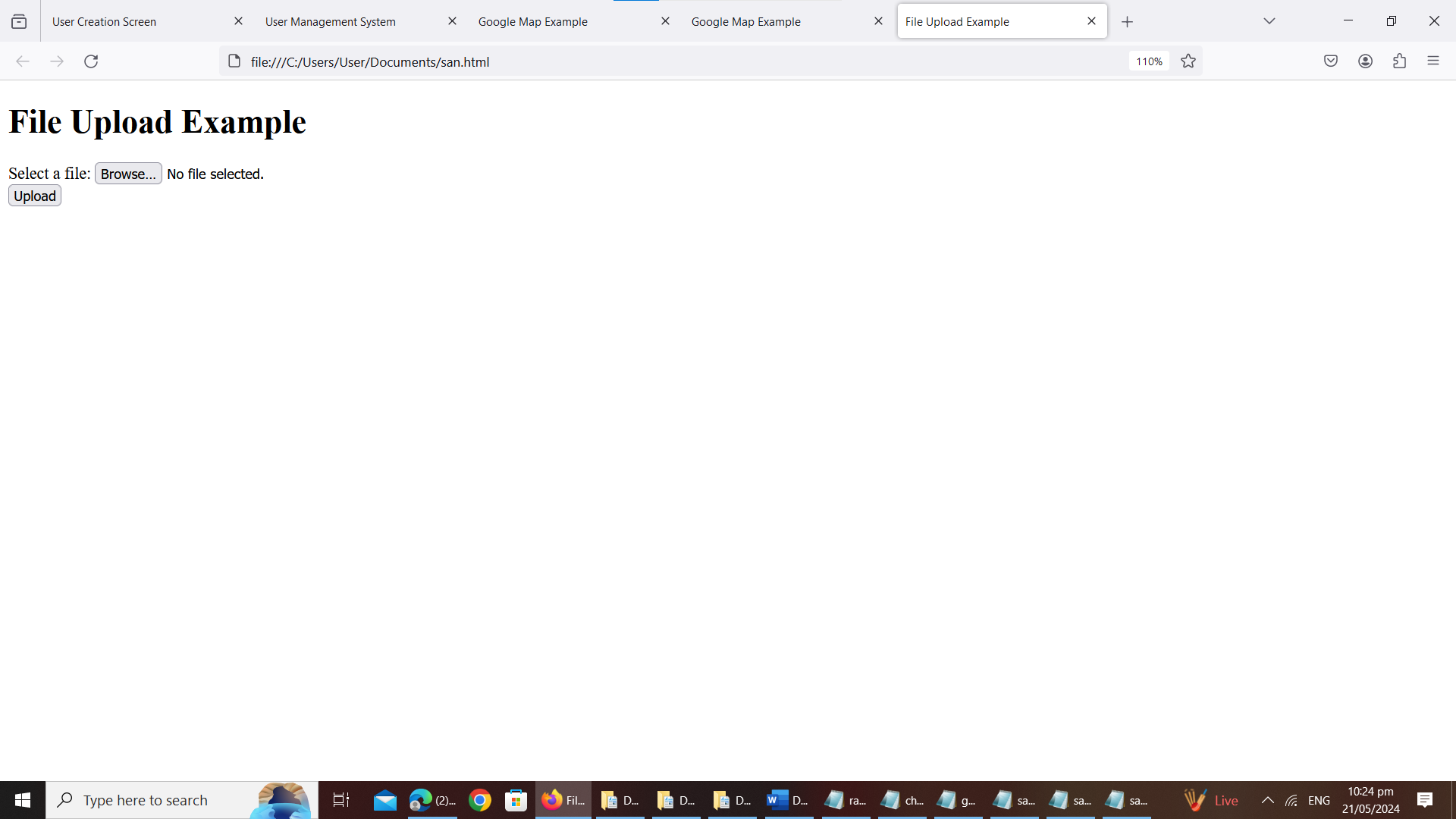
<br>

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

</form>

</body>

</html>



**10.create a html page with audio**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Audio Player Example</title>

</head>

<body>

<h1>Welcome to My Audio Player</h1>

<p>Listen to this cool audio clip:</p>

<audio controls>

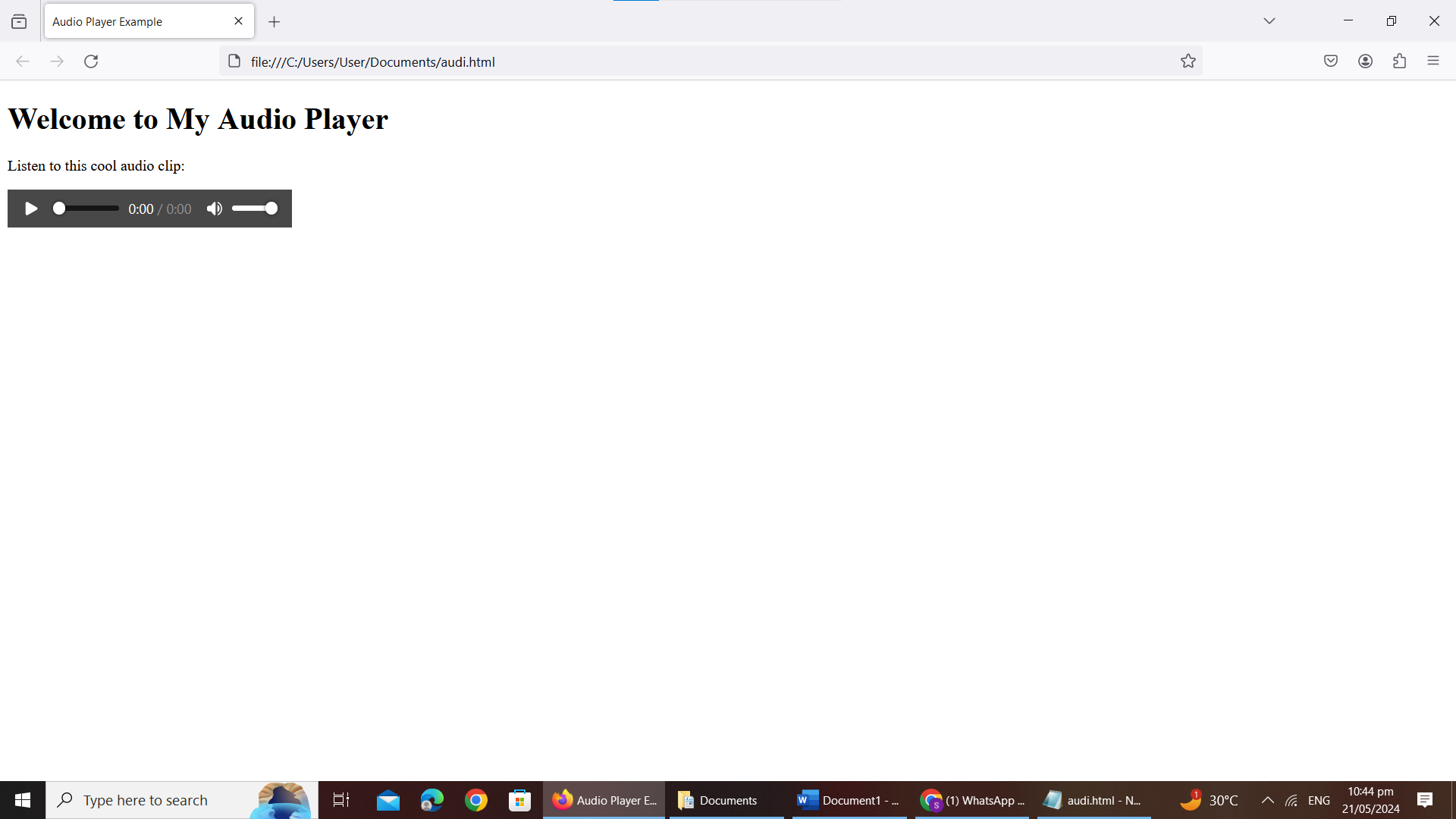
<source src="path/to/your-audio-file.mp3" type="audio/mpeg">

Your browser does not support the audio element.

</audio>

</body>

</html>



**2.differences b/w Authentication and Authorization**

| **Aspect** | **Authentication** | **Authorization** |
| --- | --- | --- |
| Definition | Process of verifying the identity of a user or system. | Process of determining if a user or system has permission to access a resource or perform an action. |
| Purpose | Ensures the user is who they claim to be. | Controls access rights and permissions after successful authentication. |
| Example | Entering username/password, biometric scans, tokens. | Setting user roles, access control lists (ACLs), permissions. |
| Focus | Verifies identity credentials. | Manages permissions and access levels. |
| Outcome | Grants access upon successful authentication. | Defines what actions or resources a user can access. |

**11. create a html page with video**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Video Example</title>

<style>

body {

font-family: Arial, sans-serif;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

background-color: #f0f0f0;

}

video {

max-width: 100%;

height: auto;

border: 1px solid #ccc;

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

}

</style>

</head>

<body>

<video controls>

<source src="C:\Users\akshi\OneDrive\Desktop\video.mp4" type="video/mp4">

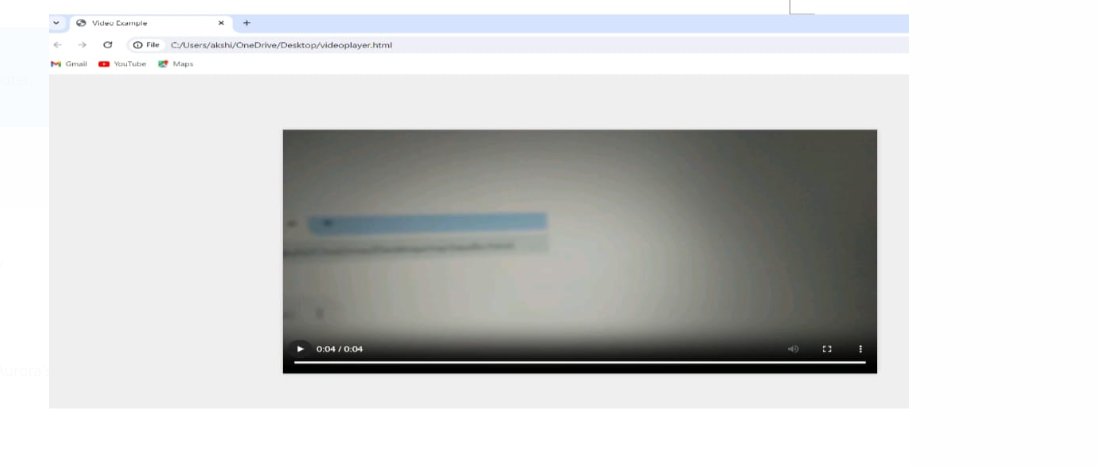
<source src="C:\Users\akshi\OneDrive\Desktop\video.mp4" type="video/ogg">

Your browser does not support the video tag.

</video>

</body>

</html>



**1.differenes between JSON and XML**

| **Aspect** | **JSON** | **XML** |
| --- | --- | --- |
| Syntax | Lightweight, uses key-value pairs separated by commas. | Heavier, uses tags to define elements structured hierarchically. |
| Readability | Generally more readable to humans due to simplicity. | Can be verbose, making it less readable especially in large files. |
| Data Types | Supports fewer data types: strings, numbers, booleans, null. | Supports a wider range of data types, including custom ones. |
| Parsing | Parsing is generally faster due to its simpler structure. | Parsing is more complex and can be slower due to nested elements. |
| Usage | Widely used for APIs and data interchange in web development. | Commonly used in document storage, configuration, and web services. |