1. Difference between JSON and XML.

	JSON	XML
Stands for	*JSON* means JavaScript Object Notation.	*XML* means Extensible Markup Language.
History	Douglas Crockford and Chip Morningstar released JSON in 2001.	The XML Working Group released XML in 1998.
Format	JSON uses a maplike structure with key-value pairs.	XML stores data in a tree structure with namespaces for different data categories.
Syntax	The syntax of JSON is more compact and easier to read and write.	The syntax of XML substitutes some characters for entity references, making it more verbose.
Parsing	You can parse JSON with a standard JavaScript function.	You need to parse XML with an XML parser.
Schema documentation	JSON is simple and more flexible.	XML is complex and less flexible.
Data types	JSON supports numbers, objects, strings, and Boolean arrays.	XML supports all JSON data types and additional types like Boolean, dates, images, and namespaces.
Ease of use	JSON has smaller file sizes and faster data transmission.	XML tag structure is more complex to write and read and results in bulky files.
Security	JSON is safer than XML.	You should turn off DTD when working with XML to mitigate potential security risks.

2. Create 3 XML and JSON files for department, year, student.

XML Files:

department.xml:

```
<department>
   <id>1</id>
   <name>Computer Science</name>
</department>
```

year.xml:

```
<year>
   <id>2024</id>
   <name>Senior Year</name>
</year>
```

student.xml:

```
<student>
   <id>12345</id>
   <name>John Doe</name>
   <department>Computer Science</department>
   <year>2024</year>
</student>
```

JSON Files:

department.json:

```
{
   "id": 1,
   "name": "Computer Science"
}
```

year.json:

```
{
   "id": 2024,
   "name": "Senior Year"
}
```

student.json:

```
{
   "id": 12345,
   "name": "John Doe",
   "department": "Computer Science",
   "year": 2024
}
```

3. **Create a file with department as root, year as subroot and student as an element.**

combined.xml:

```
<department>
   <name>Computer Science</name>
   <year>
      <name>2024</name>
      <student>
         <id>12345</id>
         <name>John Doe</name>
      </student>
   </year>
</department>
```

4. Difference between Authorization and Authentication.

Authentication	Authorization
In the authentication process, the identity of users are checked for providing the access to the system.	While in authorization process, a the person's or user's authorities are checked for accessing the resources.
In the authentication process, users or persons are verified.	While in this process, users or persons are validated.
It is done before the authorization process.	While this process is done after the authentication process.
It needs usually the user's login details.	While it needs the user's privilege or security levels.
Authentication determines whether the person is user or not.	While it determines **What permission does the user have?**
Generally, transmit information through an ID Token.	Generally, transmit information through an Access Token.
The OpenID Connect (OIDC) protocol is an authentication protocol that is generally in charge of user authentication process.	The OAuth 2.0 protocol governs the overall system of user authorization process.
Popular Authentication Techniques- • Password-Based Authentication • Passwordless Authentication • 2FA/MFA (Two-Factor Authentication / Multi-Factor Authentication) • Single sign-on (SSO) • Social authentication	Popular Authorization Techniques- • Role-Based Access Controls (RBAC) • JSON web token (JWT) Authorization • SAML Authorization • OpenID Authorization • OAuth 2.0 Authorization

Authentication	Authorization
The authentication credentials can be changed in part as and when required by the user.	The authorization permissions cannot be changed by user as these are granted by the owner of the system and only he/she has the access to change it.
The user authentication is visible at user end.	The user authorization is not visible at the user end.
The user authentication is identified with username, password, face recognition, retina scan, fingerprints, etc.	The user authorization is carried out through the access rights to resources by using roles that have been pre-defined.
Example: Employees in a company are required to authenticate through the network before accessing their company email.	**Example:** After an employee successfully authenticates, the system determines what information the employees are allowed to access.

5. Create a Login Screen.

login.html:

```
<!DOCTYPE html>
<html>
<head>
  <title>Login</title>
</head>
<body>
  <h2>Login Screen</h2>
  <form action="/login" method="post">
```

```
        <label for="username">Username:</label><br>
        <input type="text" id="username" name="username"><br>
        <label for="password">Password:</label><br>
        <input type="password" id="password" name="password"><br><br>
        <input type="submit" value="Login">
    </form>
</body>
</html>
```

6. Create a User Creation Screen by using all elements in it (like List, Radio button, Drop down, CheckBox).

create_user.html:

```
<!DOCTYPE html>

<html>

<head>

  <title>Create User</title>

</head>

<body>

  <h2>User Creation Screen</h2>

  <form action="/create_user" method="post">

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

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



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

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



    <label for="email">Email:</label><br>
```

```
<input type="email" id="email" name="email"><br><br>

<label for="gender">Gender:</label><br>
<input type="radio" id="male" name="gender" value="male">
<label for="male">Male</label><br>
<input type="radio" id="female" name="gender" value="female">
<label for="female">Female</label><br><br>

<label for="role">Role:</label><br>
<select id="role" name="role">
   <option value="admin">Admin</option>
   <option value="user">User</option>
</select><br><br>

<label for="interests">Interests:</label><br>
<input type="checkbox" id="sports" name="interests" value="sports">
<label for="sports">Sports</label><br>
<input type="checkbox" id="music" name="interests" value="music">
<label for="music">Music</label><br>
<input type="checkbox" id="reading" name="interests" value="reading">
<label for="reading">Reading</label><br><br>

<input type="submit" value="Create User">
```

```
    </form>
</body>
</html>
```

7. List all Users, Update user and Delete user (Popup for confirmation e.g., Are you sure you want to delete).

users.html:

```
<!DOCTYPE html>
<html>
<head>
    <title>User Management</title>
    <script>
        function confirmDelete(userId) {
            if (confirm("Are you sure you want to delete this user?")) {
                // Logic to delete user
                window.location.href = "/delete_user?id=" + userId;
            }
        }
    </script>
</head>
<body>
    <h2>Users List</h2>
    <table>
```

```
        <tr>
            <th>ID</th>
            <th>Username</th>
            <th>Actions</th>
        </tr>
        <tr>
            <td>1</td>
            <td>john_doe</td>
            <td>
                <a href="/update_user?id=1">Update</a>
                <button onclick="confirmDelete(1)">Delete</button>
            </td>
        </tr>
    </table>
</body>
</html>
```

8. Create a HTML page with Google Map.

map.html:

```
<!DOCTYPE html>
<html>
<head>
    <title>Google Map</title>
```

```
    <script src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY"></script>
    <script>
      function initMap() {
        var location = {lat: -34.397, lng: 150.644};
        var map = new google.maps.Map(document.getElementById('map'), {
          zoom: 8,
          center: location
        });
        var marker = new google.maps.Marker({
          position: location,
          map: map
        });
      }
    </script>
</head>
<body onload="initMap()">
    <h2>Google Map</h2>
    <div id="map" style="width:100%;height:500px;"></div>
</body>
</html>
```

9. Create a HTML page with Video file.

video.html:

```
<!DOCTYPE html>
<html>
<head>
  <title>Video File</title>
</head>
<body>
  <h2>Video File</h2>
  <video width="600" controls>
    <source src="movie.mp4" type="video/mp4">
    Your browser does not support the video tag.
  </video>
</body>
</html>
```

10. Create a HTML page with Audio file.

audio.html:

```
<!DOCTYPE html>
<html>
<head>
  <title>Audio File</title>
```

```
</head>
<body>
  <h2>Audio File</h2>
  <audio controls>
    <source src="audio.mp3" type="audio/mpeg">
    Your browser does not support the audio tag.
  </audio>
</body>
</html>
```

11. Create a HTML page to upload a file.

upload.html:

```
<!DOCTYPE html>
<html>
<head>
  <title>File Upload</title>
```

```
</head>
<body>
    <h2>Upload a File</h2>
    <form action="/upload" method="post" enctype="multipart/form-data">
        <input type="file" name="file" id="file"><br><br>
        <input type="submit" value="Upload File">
    </form>
</body>
</html>
```