



Module Code & Module Title

CS5004NT Emerging Programming Platforms and Technologies

Assessment Weightage & Type

30% Individual Coursework

Year and Semester

2021 Autumn/Spring

Student Name: Alisha Shrestha

London Met ID: 19033571

College ID: np05cp4s200028

Assignment Due Date: 7 May, 2021

Assignment Submission Date: 7 May, 2021

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

Acknowledgement

It gives huge pleasure in performing this assignment. While performing this project many difficulties were faced but with the help and guideline of some respected person and their contribution it was possible to accomplish the task successfully. Firstly I would like to express my excessive gratitude and respect towards our honorable module leader Mr. Pradhumna Dhungana, the man who introduced us to this module and for his constant guidance throughout numerous consultations and every possible help in the overall preparation of this task. Also we would like to thank Ithahari International College and London Metropolitan University for providing us with this wonderful opportunity to get engaged in such task where we can have a good experience in performing the given task. Secondly I would like to express my deepest appreciation to all who have guided directly or indirectly and helped to complete this task within the scheduled time.

Table of Contents

1. Introduction	1
2. Tree Diagram	3
3. Development.....	4
3.1. XML.....	4
3.2. DTD.....	11
3.3. Schema.....	12
3.4. CSS.....	16
4. Difference between Schema and DTD.....	25
5. Testing	27
5.1. Well-formed in web browser.....	27
5.2. Well formed in Xml validation website.....	28
5.3. Validate XML with DTD	29
5.4. Validate Xml with DTD and schema.....	31
5.5. CSS.....	34
5.6. Data type in schema	35
5.7. Enumeration in Schema.....	36
5.8. Pattern for Track length.....	37
5.9. Pattern for Price	38
5.10. Pattern for website	39
6. Development Process	40
7. Critical Analysis.....	41
Conclusion	42
8. Bibliography	44

Table of Figures

Figure 1 Tree Diagram	3
Figure 2 Evidence of test case 1	27
Figure 3 Evidence of uploading the XML file	28
Figure 4 Evidence of no error found	29
Figure 5 Evidence of uploading XML in test case 3	30
Figure 6 Evidence of uploading DTD file in test case 3	30
Figure 7 Evidence of no error found in test case 3	31
Figure 8 Evidence of uploading XML file in test case 4	32
Figure 9 Evidence of uploading DTD file in test case 4	32
Figure 10 Evidence of uploading Schema file in test case 4	33
Figure 11 Evidence of no error found in test case 4	33
Figure 12 Evidence of test case 5	34
Figure 13 First evidence of test case 6	35
Figure 14 Second evidence of test case 6	35
Figure 15 Evidence of test case 7	36
Figure 16 First evidence of test case 6	37
Figure 17 Second evidence of test case 6	37
Figure 18 First evidence of test case 6	38
Figure 19 Second evidence of test case 6	38
Figure 20 Evidence of test case 10	39

Table of Tables

Table 2 Difference between DTD and Schema	25
Table 3 Test case 1	27
Table 4 Test case 2	28
Table 5 Test case 3	29
Table 6 Test case 4	31
Table 7 Test case 5	34
Table 8 Test Case 6	35
Table 9 Test case 7	36
Table 10 Test case 8	37
Table 11 Test case 9	38
Table 12 Test case 10	39

1. Introduction

The second coursework assigned as an individual task for the module Emerging Programming Platforms and Technologies is all about development of the static webpage of Marga Music store with the proper use of XML as an XML developer. Like studying in the module, all the required elements of the XML are used properly.

Marga Music store is very well-known for its collection of music since 2015. The store is located in Gautampath, Dharan and owned by Suman Pariyar. The store has been providing its best collection by mainly focusing on five genres which includes Jazz, Rock, Country Music, Pop Music, and Dubstep.

They provide information about different songs accessible at their store to the customer and in a similar time store is extending their business soon so for that to maintain the minimum expenditure and increase productivity the store asked to design an XML that must be equivalent to the information that will make the store to connect with other platforms in no time. The XML will contain all information to be surpassed in another medium with strict validation using schema.

In this project, XML is used to organize the information and manage how it must be displayed. XML stands for extensible markup language. A markup language is a set of codes, or tags that describes the text in a digital document. XML, a more flexible cousin of HTML, makes it possible to conduct complex business over the Internet. The main benefit of xml is that you can use it to take data from a program like Microsoft SQL, convert it into XML then share that XML with other programs and platforms. You can communicate between two platforms which are generally very difficult. Many corporation use XML interfaces for databases, programming, office application mobile phones and more because of its platform independent feature. (Roche, 2000)

DTD stands for Document Type Definition, here in this project DTD is used to define the structure of an XML document. DTDs check vocabulary and validity of the structure of XML documents against grammatical rules of appropriate XML language. An XML DTD can be either specified inside the document, or it can be kept in a separate document and then linked separately. A DTD is referred to as an internal DTD if elements are

declared within the XML files and in external DTD elements are declared outside the XML file. (tutorialspoint, n.d.)

Schema is used in this task to define the legal building blocks of an XML document elements and attributes that can appear in a document. Schema is a language which is used for expressing constraint about XML documents and it is similar to DTD but provides more control on XML structure. There are two data types of schema simple type and complex type where simple type allows having text-based elements whereas complex type allows holding multiple attributes and elements. (Java T point, n.d.)

Cascading Style Sheets, fondly referred to as CSS is used in this task to display the contents of the XML document in a clear and precise manner. CSS is a simple design language intended to simplify the process of making web pages presentable. CSS handles the look and feel part of a web page. Using CSS, we can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs, and variations in display for different devices and screen sizes as well as a variety of other effects. (geeksforgeeks, n.d.)

2. Tree Diagram

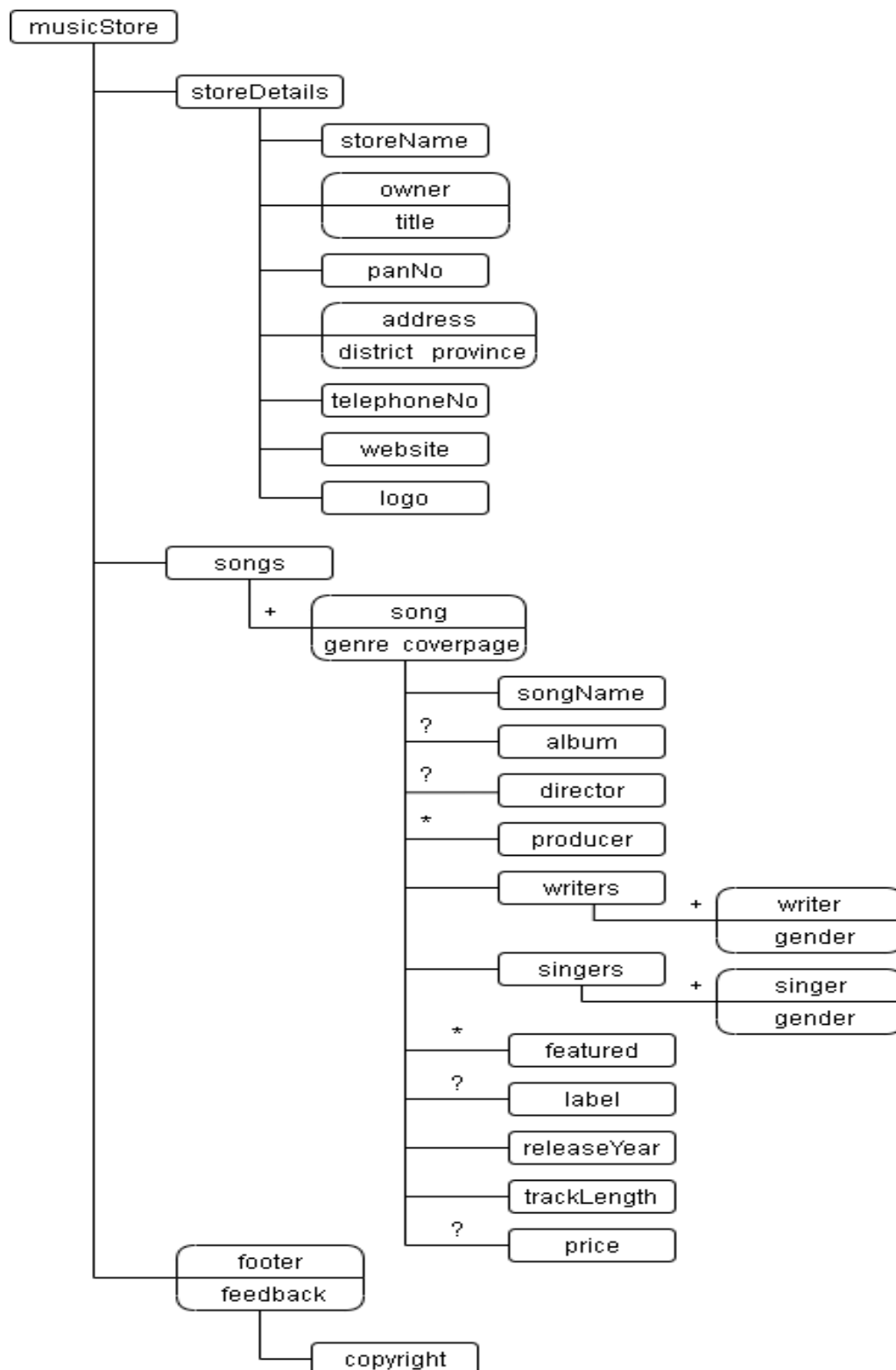


Figure 1 Tree Diagram

3. Development

3.1. XML

```
1.  <?xml version="1.0" encoding="UTF-8"?>
2.  <?xml-stylesheet type="text/css" href="catalog_19033571.css"?>
3.  <!DOCTYPE musicStore SYSTEM "catalog_19033571.dtd">
4.  <musicStore>
5.      <storeDetails>
6.          <storeName>Marga Music Store</storeName>
7.          <owner title="Mr.">Suman Pariyar</owner>
8.          <panNo>1983474</panNo>
9.          <address city="Dharan" province="1">Gauthampath</address>
10.         <telephoneNo>025531724</telephoneNo>
11.         <website>www.margamusic.com</website>
12.         <logo>Marga</logo>
13.     </storeDetails>
14.     <songs>
15.         <song genre="Jazz" coverpage="01">
16.             <songName>"Hit the road jack"</songName>
17.             <album>Ray Charles Greatest Hits</album>
18.             <producer>Sid Feller</producer>
19.             <writers>
20.                 <writer gender="Male">Ray charles</writer>
21.             </writers>
22.             <singers>
23.                 <singer gender="Male">Ray charles</singer>
24.             </singers>
25.             <featured> Margie Hendrix</featured>
26.             <label>ABC-Paramount</label>
27.             <releaseYear>1962</releaseYear>
28.             <trackLength>2:00</trackLength>
29.             <price>240</price>
30.         </song>
31.         <song genre="Jazz" coverpage="02" >
32.             <songName>"Don't Worry be happy"</songName>
33.             <album>Simple Pleasures</album>
34.             <producer>Linda Goldstein</producer>
35.             <writers>
36.                 <writer gender="Male">Bobby McFerrin</writer>
37.             </writers>
```

```
38.          <singers>
39.              <singer gender="Male">Bobby McFerrin</singer>
40.          </singers>
41.          <label>"EMI-Manhattan"</label>
42.          <releaseYear>1998</releaseYear>
43.          <trackLength>4:50</trackLength>
44.          <price>300</price>
45.      </song>
46.      <song genre="Rock" coverpage="03">
47.          <songName>"Hotel California"</songName>
48.          <album>Hotel California</album>
49.          <producer>Bill Szymczyk</producer>
50.          <writers>
51.              <writer gender="Male">Don Felder</writer>
52.              <writer gender="Male">Glenn Frey</writer>
53.          </writers>
54.          <singers>
55.              <singer gender="Male">Eagles</singer>
56.          </singers>
57.          <label>Asylum Records</label>
58.          <releaseYear>1977</releaseYear>
59.          <trackLength>6:30</trackLength>
60.          <price>250</price>
61.      </song>
62.      <song genre="Country Music" coverpage="04" >
63.          <songName>"You are my sun"</songName>
64.          <album>Unearthed</album>
65.          <producer>Milton Okun</producer>
66.          <writers>
67.              <writer gender="Male">Jimmie Davis</writer>
68.              <writer gender="Male">Charles Mitchell</writer>
69.          </writers>
70.          <singers>
71.              <singer gender="Male">Johnny Cash</singer>
72.          </singers>
73.          <featured>Susan Ruskin</featured>
74.          <label>Decca</label>
75.          <releaseYear>1940</releaseYear>
76.          <trackLength>3:26</trackLength>
77.          <price>350</price>
78.      </song>
```

```
79.      <song genre="Pop Music" coverpage="05" >
80.          <songName>"Rolling in the deep"</songName>
81.          <album>Album: 21</album>
82.          <director>Sam Brown</director>
83.          <producer>Paul Epworth</producer>
84.          <writers>
85.              <writer gender="Female">Adele Adkins</writer>
86.          </writers>
87.          <singers>
88.              <singer gender="Female">Adele</singer>
89.          </singers>
90.          <label>XLColumbia</label>
91.          <releaseYear>2010</releaseYear>
92.          <trackLength>3:48</trackLength>
93.          <price>300</price>
94.      </song>
95.      <song genre="Pop Music" coverpage="06" >
96.          <songName>"Thousand Years"</songName>
97.          <producer>Paul Katz</producer>
98.          <writers>
99.              <writer gender="Male">David Hodges</writer>
100.         </writers>
101.         <singers>
102.             <singer gender="Female">Christina Perri</singer>
103.         </singers>
104.         <featured>Twilight Saga: Breaking Dawn</featured>
105.         <label>Atlantic Records</label>
106.         <releaseYear>2011</releaseYear>
107.         <trackLength>5:05</trackLength>
108.         <price>250</price>
109.     </song>
110.     <song genre="Country Music" coverpage="07" >
111.         <songName>"Fast Car"</songName>
112.         <album>Tracy Chapman</album>
113.         <producer>David Kershenbaum</producer>
114.         <writers>
115.             <writer gender="Male">Tracy Chapman</writer>
116.         </writers>
117.         <singers>
118.             <singer gender="Male">Tracy Chapman</singer>
119.         </singers>
```

```
120.          <label>Elektra</label>
121.          <releaseYear>1987</releaseYear>
122.          <trackLength>4:57</trackLength>
123.          <price>150</price>
124.      </song>
125.      <song genre="Pop Music" coverpage="08">
126.          <songName>"Peaches"</songName>
127.          <producer>Shndo</producer>
128.          <writers>
129.              <writer gender="Male">Justin Bieber</writer>
130.              <writer gender="Male">Ashton Simmonds</writer>
131.              <writer gender="Male">Giveon Evansr</writer>
132.              <writer gender="Male">Louis Bellr</writer>
133.          </writers>
134.          <singers>
135.              <singer gender="Male">Justin Bieber</singer>
136.          </singers>
137.          <featured>Daniel Caesar</featured>
138.          <label>Def Jam</label>
139.          <releaseYear>2021</releaseYear>
140.          <trackLength>3:18</trackLength>
141.          <price>350</price>
142.      </song>
143.      <song genre="Rock" coverpage="09">
144.          <songName>"Sweet child"</songName>
145.          <album>Appetite for Destruction</album>
146.          <producer>Mike Clink</producer>
147.          <writers>
148.              <writer gender="Male">Axl Rose</writer>
149.          </writers>
150.          <singers>
151.              <singer gender="Male">Guns N' Roses</singer>
152.          </singers>
153.          <label>Geffen</label>
154.          <releaseYear>1988</releaseYear>
155.          <trackLength>5:55</trackLength>
156.          <price>200</price>
157.      </song>
158.      <song genre="Rock" coverpage="10" >
159.          <songName>"Highway to hell"</songName>
160.          <album> Highway to hell</album>
```

```
161.          <producer>Mutt Lange</producer>
162.          <writers>
163.              <writer gender="Male">Angus Young</writer>
164.              <writer gender="Male">Malcolm Young</writer>
165.              <writer gender="Male">Bon Scott</writer>
166.          </writers>
167.          <singers>
168.              <singer gender="Male">ACDC</singer>
169.          </singers>
170.          <label>Albert Productions</label>
171.          <releaseYear>1979</releaseYear>
172.          <trackLength>3:27</trackLength>
173.          <price>200</price>
174.      </song>
175.      <song genre="Dubstep" coverpage="11" >
176.          <songName>"Scary Monsters and Nice Sprites"</songName>
177.          <producer>Foreign Beggars</producer>
178.          <writers>
179.              <writer gender="Male">Skrillex</writer>
180.          </writers>
181.          <singers>
182.              <singer gender="Male">Skrillex</singer>
183.          </singers>
184.          <label>Big Beat Records</label>
185.          <releaseYear>2010</releaseYear>
186.          <trackLength>4:02</trackLength>
187.          <price>250</price>
188.      </song>
189.      <song genre="Dubstep" coverpage="12" >
190.          <songName>"Me and You"</songName>
191.          <album>Welcome Reality</album>
192.          <producer>Nero</producer>
193.          <writers>
194.              <writer gender="Male"> Daniel Stephens</writer>
195.              <writer gender="Male"> Joseph Ray</writer>
196.          </writers>
197.          <singers>
198.              <singer gender="Male">Nero</singer>
199.          </singers>
200.          <label>Mercury</label>
201.          <releaseYear>2010</releaseYear>
```

```
202.         <trackLength>3:55</trackLength>
203.         <price>250</price>
204.     </song>
205.     <song genre="Dubstep" coverpage="13">
206.         <songName>"I Need Air"</songName>
207.         <album>Magnetic Man</album>
208.         <producer>Angela Hunte</producer>
209.         <writers>
210.             <writer gender="Male">Arthur Smith</writer>
211.             <writer gender="Female">Angela Hunte</writer>
212.             <writer gender="Male">Oliver Jones</writer>
213.         </writers>
214.         <singers>
215.             <singer gender="Male">Magnetic Man</singer>
216.         </singers>
217.         <featured>Angela Hunte</featured>
218.         <label>    Startime</label>
219.         <releaseYear>2010</releaseYear>
220.         <trackLength>4:18</trackLength>
221.         <price>200</price>
222.     </song>
223.     <song genre="Country Music" coverpage="14" >
224.         <songName>"Take me home"</songName>
225.         <album> Poems, prayers and promises</album>
226.         <producer>Susan Ruskin</producer>
227.         <writers>
228.             <writer gender="Male">Bill Danoff</writer>
229.             <writer gender="Male">Taffy Nivert</writer>
230.             <writer gender="Male">John Denver</writer>
231.         </writers>
232.         <singers>
233.             <singer gender="Male">John Denver</singer>
234.         </singers>
235.         <label>RCA</label>
236.         <releaseYear>1971</releaseYear>
237.         <trackLength>3:17</trackLength>
238.         <price>150</price>
239.     </song>
240.     <song genre="Country Music" coverpage="15" >
241.         <songName>"I Walk the Line"</songName>
242.         <album>Johnny Cash with His Hot and Blue Guitar</album>
```

```
243.          <producer>Sam Phillips</producer>
244.          <writers>
245.              <writer gender="Male">Johnny Cash</writer>
246.          </writers>
247.          <singers>
248.              <singer gender="Male">Johnny Cash</singer>
249.          </singers>
250.          <label>Sun</label>
251.          <releaseYear>1956</releaseYear>
252.          <trackLength>2:45</trackLength>
253.          <price>250</price>
254.      </song>
255. </songs>
256. <footer feedback="Rating ★★★★★">
257.     <copyright>Copyright © 2021 All rights reserved</copyright>
258. </footer>
259.</musicStore>
```

3.2. DTD

1. <!ELEMENT musicStore (storeDetails, songs, footer)>
2. <!ELEMENT storeDetails (storeName,owner, panNo, address, telephoneNo, website, logo)>
3. <!ELEMENT storeName (#PCDATA)>
4. <!ELEMENT owner (#PCDATA)>
5. <!ELEMENT panNo (#PCDATA)>
6. <!ELEMENT address (#PCDATA)>
7. <!ELEMENT telephoneNo (#PCDATA)>
8. <!ELEMENT website (#PCDATA)>
9. <!ELEMENT logo (#PCDATA)>
10. <!ELEMENT songs (song+)>
11. <!ELEMENT song (songName, album?, director?, producer*, writers, singers, featured*, label?, releaseYear, trackLength, price?)>
12. <!ELEMENT songName (#PCDATA)>
13. <!ELEMENT album (#PCDATA)>
14. <!ELEMENT director (#PCDATA)>
15. <!ELEMENT producer (#PCDATA)>
16. <!ELEMENT writers (writer+)>
17. <!ELEMENT writer (#PCDATA)>
18. <!ELEMENT singers (singer+)>
19. <!ELEMENT singer (#PCDATA)>
20. <!ELEMENT featured (#PCDATA)>
21. <!ELEMENT label (#PCDATA)>
22. <!ELEMENT releaseYear (#PCDATA)>
23. <!ELEMENT trackLength (#PCDATA)>
24. <!ELEMENT price (#PCDATA)>
25. <!ELEMENT footer (copyright)>
26. <!ELEMENT copyright (#PCDATA)>
- 27.
28. <!ATTLIST owner title (Mr.|Mrs.) #REQUIRED>
29. <!ATTLIST address city CDATA #REQUIRED>
30. <!ATTLIST address province CDATA #REQUIRED>
31. <!ATTLIST song genre CDATA #REQUIRED>
32. <!ATTLIST song coverpage CDATA #REQUIRED>
33. <!ATTLIST writer gender (Male|Female) #REQUIRED>
34. <!ATTLIST singer gender (Male|Female) #REQUIRED>
35. <!ATTLIST footer feedback CDATA #REQUIRED>

3.3. Schema

```
1.  <?xml version="1.0" encoding="UTF-8"?>
2.  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
3.
4.    <xs:element name="musicStore">
5.      <xs:complexType>
6.        <xs:sequence>
7.          <xs:element ref="storeDetails"/>
8.          <xs:element ref="songs"/>
9.          <xs:element ref="footer"/>
10.        </xs:sequence>
11.      </xs:complexType>
12.    </xs:element>
13.
14.    <xs:element name="storeDetails">
15.      <xs:complexType>
16.        <xs:sequence>
17.          <xs:element ref="storeName"/>
18.          <xs:element ref="owner"/>
19.          <xs:element ref="panNo"/>
20.          <xs:element ref="address"/>
21.          <xs:element ref="telephoneNo"/>
22.          <xs:element ref="website"/>
23.          <xs:element ref="logo"/>
24.        </xs:sequence>
25.      </xs:complexType>
26.    </xs:element>
27.
28.    <xs:element name="storeName" type="xs:string"/>
29.    <xs:element name="telephoneNo" type="xs:integer"/>
30.    <xs:element name="logo" type="xs:string"/>
31.
32.    <xs:element name="panNo">
33.      <xs:simpleType>
34.        <xs:restriction base="xs:integer">
35.          <xs:enumeration value="1983474"/>
36.        </xs:restriction>
37.      </xs:simpleType>
38.    </xs:element>
39.
```

```
40. <xs:element name="website">
41.   <xs:simpleType>
42.     <xs:restriction base="xs:string">
43.       <xs:pattern value="[a-z]*.[a-z]*.[a-z]*"/>
44.     </xs:restriction>
45.   </xs:simpleType>
46. </xs:element>
47.
48. <xs:element name="owner">
49.   <xs:complexType mixed="true">
50.     <xs:attribute name="title" type="xs:string" use="required"/>
51.   </xs:complexType>
52. </xs:element>
53.
54. <xs:element name="address">
55.   <xs:complexType>
56.     <xs:simpleContent>
57.       <xs:extension base="xs:string">
58.         <xs:attribute name="city" type="xs:string"/>
59.         <xs:attribute name="province" type="xs:int"/>
60.       </xs:extension>
61.     </xs:simpleContent>
62.   </xs:complexType>
63. </xs:element>
64.
65. <xs:element name="songs">
66.   <xs:complexType>
67.     <xs:sequence>
68.       <xs:element ref="song" maxOccurs="unbounded"/>
69.     </xs:sequence>
70.   </xs:complexType>
71. </xs:element>
72.
73. <xs:element name="song">
74.   <xs:complexType>
75.     <xs:sequence>
76.       <xs:element ref="songName"/>
77.       <xs:element ref="album" minOccurs="0"/>
78.       <xs:element ref="director" minOccurs="0"/>
79.       <xs:element ref="producer" minOccurs="0"/>
80.       <xs:element ref="writers"/>
```

```
81.    <xs:element ref="singers"/>
82.    <xs:element ref="featured" minOccurs="0"/>
83.    <xs:element ref="label"/>
84.    <xs:element ref="releaseYear"/>
85.    <xs:element ref="trackLength"/>
86.    <xs:element ref="price"/>
87.    </xs:sequence>
88.    <xs:attribute name="genre" type="xs:string"/>
89.    <xs:attribute name="coverpage" type="xs:string"/>
90.    </xs:complexType>
91. </xs:element>
92.
93. <xs:element name="songName" type="xs:string"/>
94. <xs:element name="album" type="xs:string"/>
95. <xs:element name="director" type="xs:string"/>
96. <xs:element name="producer" type="xs:string"/>
97. <xs:element name="featured" type="xs:string"/>
98. <xs:element name="label" type="xs:string"/>
99. <xs:element name="releaseYear" type="xs:integer"/>
100.
101. <xs:element name="price">
102.   <xs:simpleType>
103.     <xs:restriction base="xs:integer">
104.       <xs:pattern value="[0-9]{3}"/>
105.     </xs:restriction>
106.   </xs:simpleType>
107. </xs:element>
108.
109. <xs:element name="trackLength">
110.   <xs:simpleType>
111.     <xs:restriction base="xs:string">
112.       <xs:pattern value="[0-9]{1}:[0-9]{2}"/>
113.     </xs:restriction>
114.   </xs:simpleType>
115. </xs:element>
116.
117. <xs:element name="writers">
118.   <xs:complexType>
119.     <xs:sequence>
120.       <xs:element ref="writer" maxOccurs="unbounded"/>
121.     </xs:sequence>
```

```
122. </xs:complexType>
123. </xs:element>
124.
125. <xs:element name="writer">
126.   <xs:complexType mixed="true">
127.     <xs:attribute name="gender" type="xs:string" use="required"/>
128.   </xs:complexType>
129. </xs:element>
130.
131. <xs:element name="singers">
132.   <xs:complexType>
133.     <xs:sequence>
134.       <xs:element ref="singer" maxOccurs="unbounded"/>
135.     </xs:sequence>
136.   </xs:complexType>
137. </xs:element>
138.
139. <xs:element name="singer">
140.   <xs:complexType mixed="true">
141.     <xs:attribute name="gender" type="xs:string" use="required"/>
142.   </xs:complexType>
143. </xs:element>
144.
145. <xs:element name="footer">
146.   <xs:complexType>
147.     <xs:sequence>
148.       <xs:element ref="copyright"/>
149.     </xs:sequence>
150.     <xs:attribute name="feedback" type="xs:string"/>
151.   </xs:complexType>
152. </xs:element>
153.
154. <xs:element name="copyright" type="xs:string"/>
155.
156.</xs:schema>
```

3.4. CSS

```
1.  * {
2.    background-color: white;
3.    display: block;
4.    font-family: Georgia, serif;
5.  }
6.
7.  musicStore {
8.    border-style: solid;
9.    border-color: rgb(146, 106, 153);
10.   margin: 9px 100px 9px 100px;
11.   background: url(/Images/background.png) no-repeat;
12.   background-attachment: fixed;
13.   background-size: 100% 100%
14. }
15.
16. storeDetails {
17.   padding-top: 90px;
18.   padding-left: 30px;
19. }
20. storeName {
21.   font-family: Courier New, sans-serif;
22.   font-weight: bold;
23.   font-size: 45px;
24.   color: rgb(5, 16, 115);
25.   position: absolute;
26.   top: 20px;
27.   left: 34%;
28. }
29.
30. owner {
31.   font-family: arial;
32. }
33.
34. owner::before {
35.   content: "Owner name: " attr(title) " ";
36. }
37.
38. panNo {
39.   position: absolute;
```

```
40. top: 18px;
41. left: 110px;
42. }
43.
44. panNo::before {
45.   content: "Pan No: ";
46.   color: rgb(202, 4, 1);
47.
48. }
49.
50. address::before {
51.   content: "Address: ";
52. }
53. address::after {
54.   content: ", " attr(city) "" ;
55. }
56.
57. telephoneNo::before {
58.   content: "Tel Ph: "
59. }
60.
61. website {
62.   position: absolute;
63.   top: 70px;
64.   left: 45%;
65.   color: rgb(3, 4, 216);
66.   border-bottom: 1px solid rgb(3, 4, 216);
67. }
68.
69. logo {
70.   float: none;
71.   background: url(/Images/logo.jpg) no-repeat right;
72.   background-size: 180px;
73.   width: 500px;
74.   height: 200px;
75.   position: absolute;
76.   right: 130px;
77.   bottom: 380px;
78.
79. }
80.
```

```
81. songs {
82.   border-style: groove;
83.   margin-left: 15%;
84.   margin-right: 15%;
85.   margin-bottom: 5px;
86.   background:rgb(230, 230, 255);
87. }
88.
89. song {
90.   border-style: dotted;
91.   border-radius: 20px;
92.   border-color: rgb(146, 106, 153);
93.   width: 700px;
94.   height: 340px;
95.   margin-top: 40px;
96.   margin-left: 40px;
97.   margin-right: 40px;
98.   margin-bottom: 40px;
99.   text-align: 50%;
100.  padding-top: 30px;
101.  font-family: Georgia, serif;
102.}
103.
104.song::before {
105.  content: "Genre: " attr(genre) "";
106.  position: relative;
107.  left: 100px;
108.  top: 15px;
109.  border: 2px dotted black;
110.  padding: 10px;
111.  border-radius: 25px;
112.  border-color: rgb(146, 106, 153);
113.}
114.
115.song[coverpage="01"] {
116.  background: url(/Images/Image1.png) no-repeat;
117.  background-size: 200px;
118.  background-position: 70px;
119.}
120.song[coverpage="02"] {
121.  background: url(/Images/Image2.jpg) no-repeat;
```

```
122. background-size: 200px;
123. background-position: 70px;
124.}
125.song[coverpage="03"] {
126. background: url(/Images/Image3.jpg) no-repeat;
127. background-size: 200px;
128. background-position: 70px;
129.}
130.song[coverpage="04"] {
131. background: url(/Images/Image4.jpg) no-repeat;
132. background-size: 200px;
133. background-position: 70px;
134.}
135.song[coverpage="05"] {
136. background: url(/Images/Image5.jpeg) no-repeat;
137. background-size: 200px;
138. background-position: 70px;
139.}
140.song[coverpage="06"] {
141. background: url(/Images/Image6.jpg) no-repeat;
142. background-size: 200px;
143. background-position: 70px;
144.}
145.song[coverpage="07"] {
146. background: url(/Images/Image7.jpg) no-repeat;
147. background-size: 200px;
148. background-position: 70px;
149.}
150.song[coverpage="08"] {
151. background: url(/Images/Image8.jpg) no-repeat;
152. background-size: 200px;
153. background-position: 70px;
154.}
155.song[coverpage="09"] {
156. background: url(/Images/Image9.jpg) no-repeat;
157. background-size: 200px;
158. background-position: 70px;
159.}
160.song[coverpage="10"] {
161. background: url(/Images/Image10.jfif) no-repeat;
162. background-size: 200px;
```



```
163. background-position: 70px;
164.}
165.song[coverpage="11"] {
166. background: url(/Images/Image11.jpg) no-repeat;
167. background-size: 200px;
168. background-position: 70px;
169.}
170.song[coverpage="12"] {
171. background: url(/Images/Image12.jpg) no-repeat;
172. background-size: 200px;
173. background-position: 70px;
174.}
175.song[coverpage="13"] {
176. background: url(/Images/Image13.jpg) no-repeat;
177. background-size: 200px;
178. background-position: 70px;
179.}
180.song[coverpage="14"] {
181. background: url(/Images/Image14.jpg) no-repeat;
182. background-size: 200px;
183. background-position: 70px;
184.}
185.song[coverpage="15"] {
186. background: url(/Images/Image15.jpg) no-repeat;
187. background-size: 200px;
188. background-position: 70px;
189.}
190.
191.songName {
192. font-family: 'Brush Script MT', cursive;
193. font-size: 30px;
194. padding-left: 20px;
195. margin-left: 380px;
196. background:rgb(230, 230, 255);
197. display: list-item;
198.}
199.
200.album {
201. margin-left: 350px;
202. padding-top: 10px;
203. background:rgb(230, 230, 255);
```

```
204.}
205.
206.director {
207.  margin-left: 350px;
208.  padding-top: 5px;
209.  background:rgb(230, 230, 255);
210.}
211.
212.producer {
213.  margin-left: 350px;
214.  padding-top: 5px;
215.  background:rgb(230, 230, 255);
216.}
217.
218.writers {
219.  margin-left: 350px;
220.  background:rgb(230, 230, 255);
221.}
222.
223.writer {
224.  padding-top: 5px;
225.  background:rgb(230, 230, 255);
226.}
227.
228.singers {
229.  margin-left: 350px;
230.  background:rgb(230, 230, 255);
231.}
232.
233.singer {
234.  padding-top: 5px;
235.  background:rgb(230, 230, 255);
236.}
237.
238.featured {
239.  margin-left: 350px;
240.  padding-top: 5px;
241.  background:rgb(230, 230, 255);
242.}
243.
244.label {
```

```
245. margin-left: 350px;
246. padding-top: 5px;
247. background:rgb(230, 230, 255);
248.}
249.
250.releaseYear {
251. margin-left: 350px;
252. padding-top: 5px;
253. background:rgb(230, 230, 255);
254.}
255.
256.trackLength {
257. margin-left: 350px;
258. padding-top: 5px;
259. background:rgb(230, 230, 255);
260.}
261.
262.price {
263. margin-left: 350px;
264. padding-top: 5px;
265. background:rgb(230, 230, 255);
266.}
267.
268.album::before {
269. content: "Album: ";
270. background:rgb(230, 230, 255);
271.
272.}
273.
274.director::before {
275. content: "Director: ";
276. background:rgb(230, 230, 255);
277.}
278.
279.producer::before {
280. content: "Producer: ";
281. background:rgb(230, 230, 255);
282.}
283.
284.writer::before {
285. content: "Writer: ";
```

```
286. background:rgb(230, 230, 255);
287.}
288.
289.writer::after {
290.  content: " Gender: " attr(gender) " ";
291.}
292.
293.singer::before {
294.  content: "Singer: ";
295.  background:rgb(230, 230, 255);
296.}
297.
298.singer::after {
299.  content: " Gender: " attr(gender) " ";
300.}
301.
302.featured::before {
303.  content: "Featured: ";
304.  background:rgb(230, 230, 255);
305.}
306.
307.label::before {
308.  content: "label: ";
309.  background:rgb(230, 230, 255);
310.}
311.
312.releaseYear::before {
313.  content: "Release Year: ";
314.  background:rgb(230, 230, 255);
315.}
316.
317.releaseYear::after {
318.  content: " AD";
319.}
320.
321.trackLength::before {
322.  content: "Track Length: ";
323.  background:rgb(230, 230, 255);
324.}
325.
326.price::before {
```

```
327. content: "Price: Rs. ";
328. background:rgb(230, 230, 255);
329.}
330.
331.footer::before {
332. content: "Feedback: " attr(feedback) "";
333. float: left;
334.}
335.
336.copyright {
337. text-align: center;
338. margin-bottom: 20px;
339. margin-top: 20px;
340.}
```

4. Difference between Schema and DTD

Table 1 Difference between DTD and Schema

DTD	Schema
DTD stands for Documentation Type Definition.	XSD stands for XML Schema Documentation.
It is derived from SMGL syntax.	It is written in XML
DTD has entities.	Schema does not have entities.
The customization of data type is not allowed.	The customization of data type is allowed.
It is difficult to develop mixed content in DTD.	It is easy to develop mixed content in Schema.
It does not support data types.	It supports data types for elements and attributes.
For DTD, there are no such things as a namespace declaration or namespace use.	For Schema. Elements and attributes in XML schema namespace are used.
DTDs do not support datatypes other than character strings.	XML schema specifies a set of built-in datatypes.
DTD does not define order for child elements.	XML schema defines order for child elements
DTD is not extensible.	XML schema is extensible.

DTD poorly support reusability.	XML support reusability as it can create its own data types.
DTD provides less control on XML structure.	XML schema provides more control on XML structure.
DTD does not validate the content of the data types.	Schema defines the data type of certain element.
DTD is suitable for small XML data.	Schema is suitable for larger data.
DTD allows the inline definition.	Schema does not allow the inline definition. (InformIT, 2001)

5. Testing

5.1. Well-formed in web browser

Table 2 Test case 1

Objective	To check the XML document is well-formed in chrome browser.
Action	The XML file is opened in chrome browser without the link of CSS file.
Expected Result	The browser should display the XML file In the form of tree structure.
Actual Result	The browser displayed the XML file in the form of tree structure without any error.
Conclusion	Test has been successfully completed.

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```

▼ <musicStore>
  ▼ <storeDetails>
    <storeName>Marga Music Store</storeName>
    <owner title="Mr.">Alisha Shrestha</owner>
    <panNo>1983474</panNo>
    <address city="Dharan" province="1">Laxmi Chowk</address>
    <telephoneNo>025531724</telephoneNo>
    <website>www.margamusic.com</website>
    <logo>Marga</logo>
  </storeDetails>
  ▼ <songs>
    ▼ <song genre="Jazz" coverpage="01">
      <songName>"Hit the road jack"</songName>
      <album>Ray Charles Greatest Hits</album>
      <producer>Sid Feller</producer>
      ▼ <writers>
        <writer gender="Male">Ray charles</writer>
      </writers>
      ▼ <singers>
        <singer gender="Male">Ray charles</singer>
      </singers>
      <featured> Margie Hendrix</featured>
      <label>ABC-Paramount</label>
      <releaseYear>1962</releaseYear>
      <trackLength>2:00</trackLength>
      <price>240</price>
    </song>
  </songs>

```

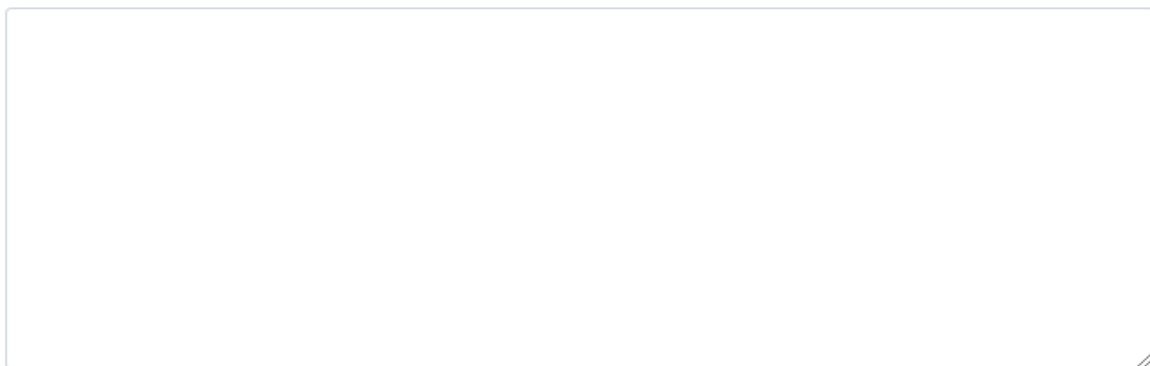
Figure 2 Evidence of test case 1

5.2. Well formed in Xml validation website

Table 3 Test case 2

Objective	To check the XML file is well-formed through online validation tool.
Action	The XML file is uploaded in the XML validator.
Expected Result	The XML validator should display the message with no error found.
Actual Result	The XML validator displayed the message with no error found.
Conclusion	Test has been successfully completed.

Please copy your XML document in here:



Or upload it:

catalog_19033571.xml

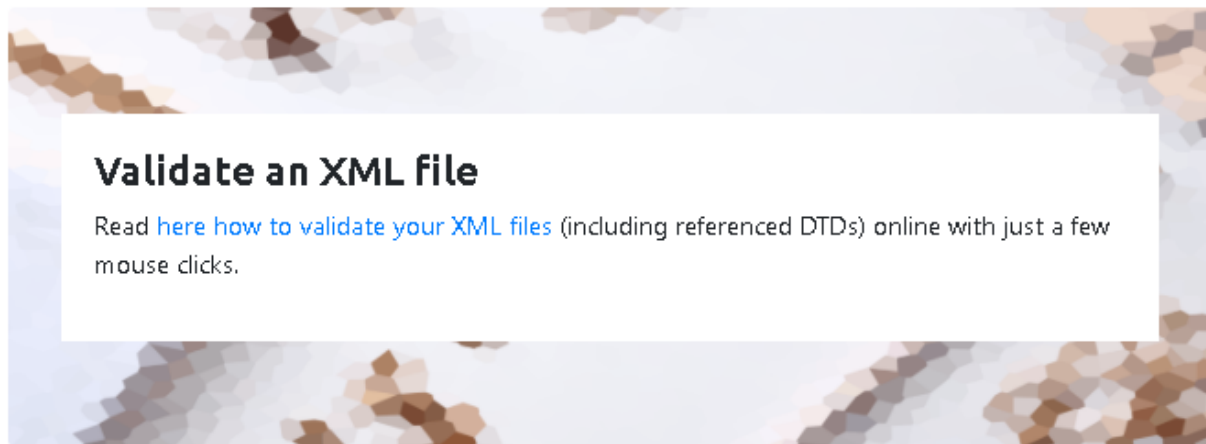
The validation check is performed against any XML schema or DTD declared inside the XML document.

If neither an XML schema nor a DTD is declared, only a syntax check is performed.

To validate the XML document against an external XML schema, click below.

☐ Validate against external XML schema

Figure 3 Evidence of uploading the XML file



No errors were found

The following files have been uploaded so far:

[XML document:](#) 

Click on any file name if you want to edit the file.

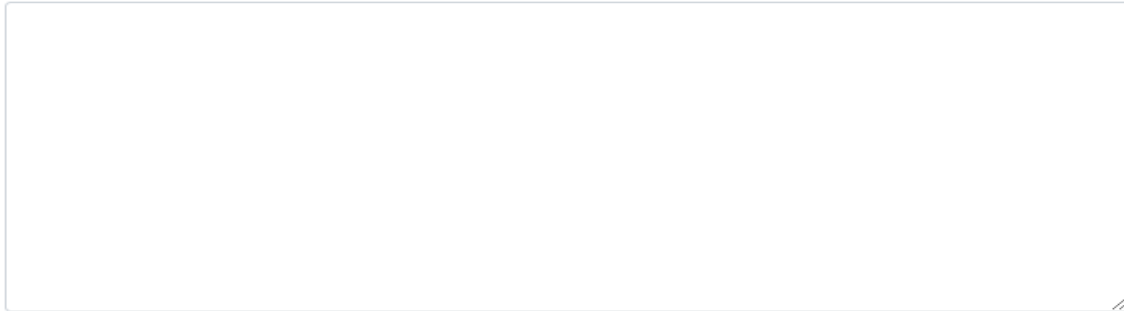
Figure 4 Evidence of no error found

5.3. Validate XML with DTD

Table 4 Test case 3

Objective	To validate the XML file using DTD.
Action	Upload the XML file and DTD file in XML validator.
Expected Result	The XML validator should display no error found.
Actual Result	The XML validator displayed no error found.
Conclusion	Test has been successfully completed.

Please copy your XML document in here:



Or upload it:

catalog_19033571.xml

The validation check is performed against any XML schema or DTD declared inside the XML document.

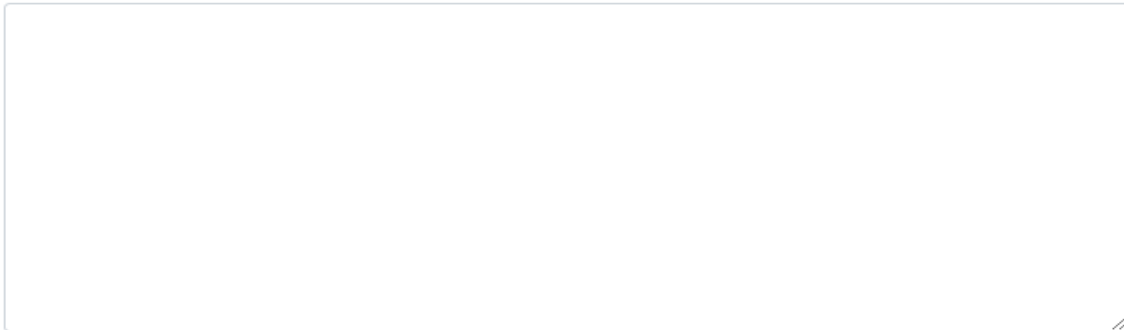
If neither an XML schema nor a DTD is declared, only a syntax check is performed.

To validate the XML document against an external XML schema, click below.

☐ Validate against external XML schema

Figure 5 Evidence of uploading XML in test case 3

The file catalog_19033571.dtd is being referenced. Please copy it in here, so that the validation can continue:



Or upload it:

catalog_19033571.dtd

The following files have been uploaded so far:

[XML document:](#) 

Click on any file name if you want to edit the file.

Figure 6 Evidence of uploading DTD file in test case 3

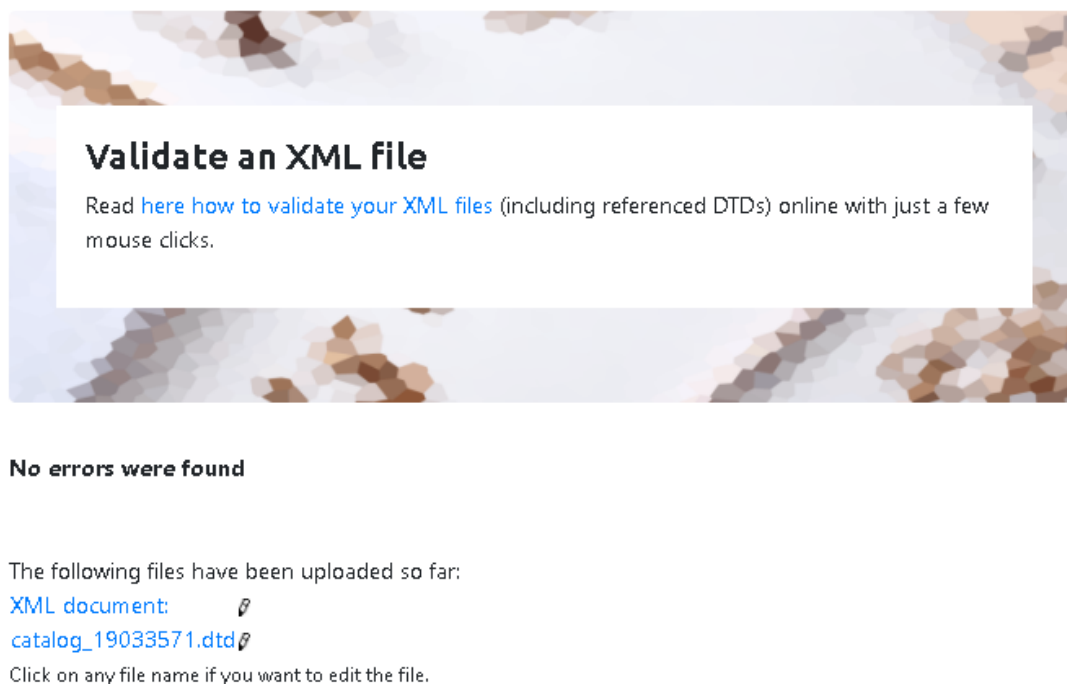


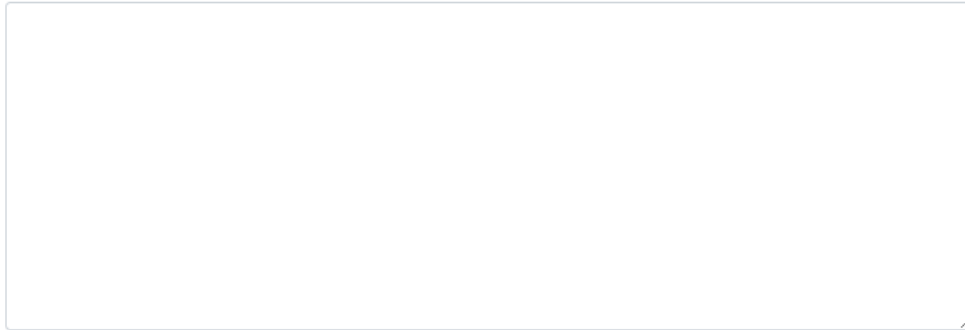
Figure 7 Evidence of no error found in test case 3

5.4. Validate Xml with DTD and schema

Table 5 Test case 4

Objective	To validate the XML file with DTD and schema with XML validator.
Action	Upload the XML, DTD and Schema file in XML validator.
Expected Result	The XML validator should display no error found.
Actual Result	The XML validator displayed no error found.
Conclusion	Test has been successfully completed.

Please copy your XML document in here:



Or upload it:

catalog_19033571.xml

The validation check is performed against any XML schema or DTD declared inside the XML document.

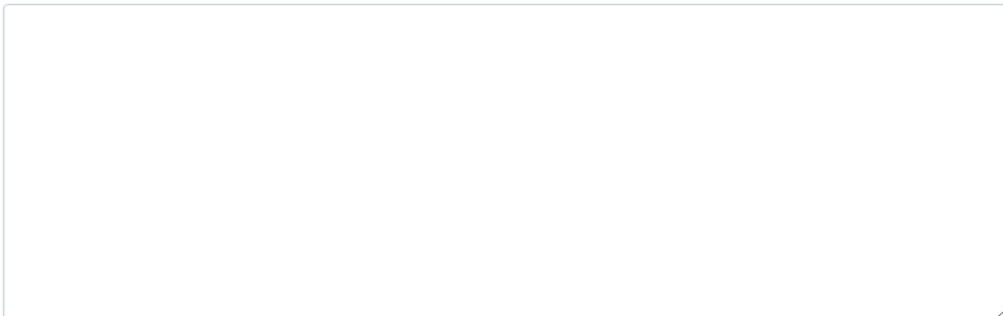
If neither an XML schema nor a DTD is declared, only a syntax check is performed.

To validate the XML document against an external XML schema, click below.

☒ Validate against external XML schema

Figure 8 Evidence of uploading XML file in test case 4


The file catalog_19033571.dtd is being referenced. Please copy it in here, so that the validation can continue:



Or upload it:

catalog_19033571.dtd

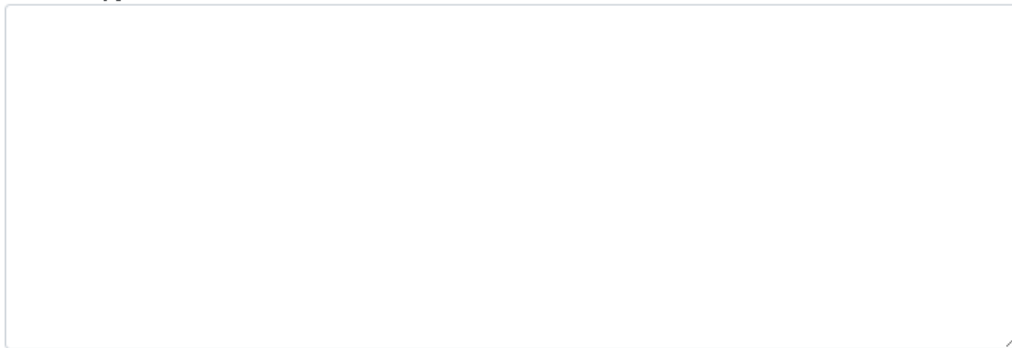
The following files have been uploaded so far:

[XML document:](#) 

Click on any file name if you want to edit the file.

Figure 9 Evidence of uploading DTD file in test case 4


Please copy the XML schema in here:



Or upload it:

catalog_19033571.xsd

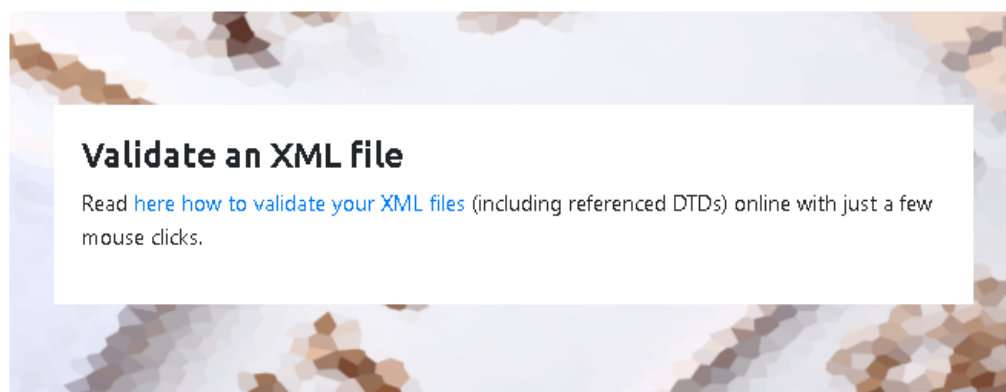
The following files have been uploaded so far:

[XML document:](#) 

[catalog_19033571.dtd](#) 


Click on any file name if you want to edit the file.


Figure 10 Evidence of uploading Schema file in test case 4




No errors were found

The following files have been uploaded so far:

[XML document:](#) 

[XML schema:](#) 

[catalog_19033571.dtd](#) 

Click on any file name if you want to edit the file.

Figure 11 Evidence of no error found in test case 4

5.5. CSS

Table 6 Test case 5

Objective	To check the XML file is linked with CSS file.
Action	Open in the XML file in chrome browser.
Expected Result	The attractive design should display the web browser.
Actual Result	The attractive design is displayed In the web browser.
Conclusion	Test has been successfully completed.

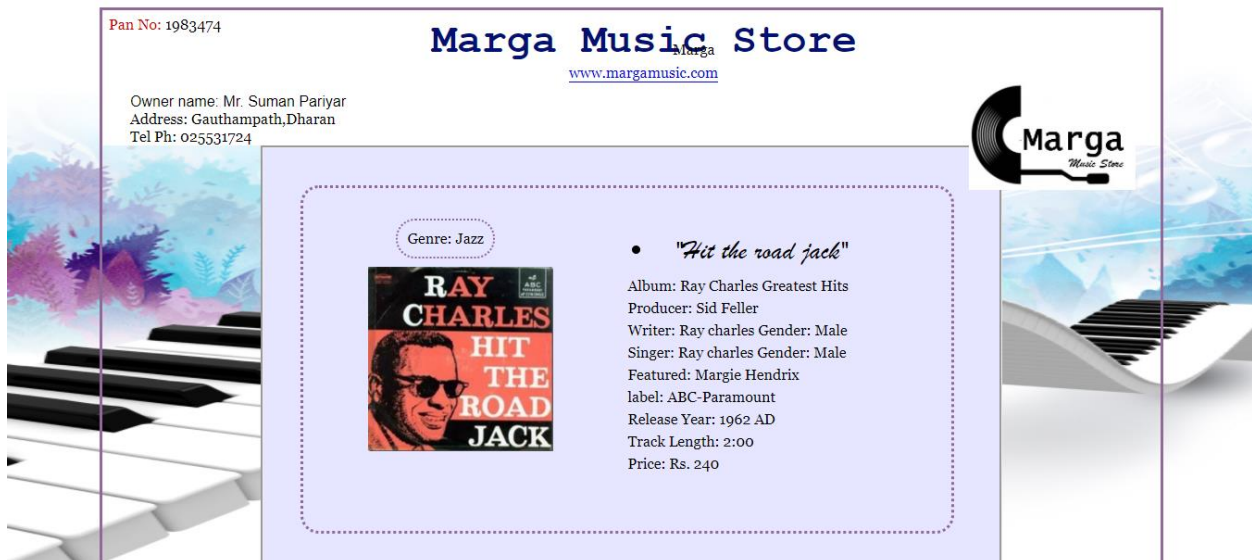




Figure 12 Evidence of test case 5

5.6. Data type in schema

Table 7 Test Case 6

Objective	To check the data type of release year element.
Action	Enter wrong data type for release year element.
Expected Result	The XML validator should display an error message.
Actual Result	The XML validator displayed an error message.
Conclusion	Test has been successfully completed.

2 errors have been found!

Click on  to jump to the error. In the document, you can point at  with your mouse to see the error message.

Errors in the XML document:

- ❌ 26:50 cvc-datatype-valid.1.2.1: 'Nineteen Ninety Six' is not a valid value for 'integer'.
- ❌ 26:50 cvc-type.3.1.3: The value 'Nineteen Ninety Six' of element 'releaseYear' is not valid.

XML document:

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE musicStore SYSTEM "catalog_19033571.dtd">
3 <musicStore>
4 <storeDetails>
5 <storeName>Marga Music Store</storeName>
6 <owner title="Mr.">Alisha Shrestha</owner>
7 <panNo>1983474</panNo>

```

Figure 13 First evidence of test case 6

```

21 <singers>
22 <singer gender="Male">Ray charles</singer>
23 </singers>
24 <featured> Margie Hendrix</featured>
25 <label>ABC-Paramount</label>
    <releaseYear>Nineteen Ninety Six</releaseYear>
26 ❌
    ❌
27 <trackLength>2:00</trackLength>
28 <price>240</price>

```



Figure 14 Second evidence of test case 6

5.7. Enumeration in Schema



Table 8 Test case 7

Objective	To check the enumeration of pan number element.
Action	Enter wrong pan number for panNo element.
Expected Result	The XML validator should display an error message.
Actual Result	The XML validator displayed an error message.
Conclusion	Test has been successfully completed.

2 errors have been found!



Click on  to jump to the error. In the document, you can point at  with your mouse to see the error message.

Errors in the XML document:

-  7:28 cvc-enumeration-valid: Value '1983474123' is not facet-valid with respect to enumeration '[1983474]'. It must be a value from the enumeration.
-  7:28 cvc-type.3.1.3: The value '1983474123' of element 'panNo' is not valid.

XML document:

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE musicStore SYSTEM "catalog_19033571.dtd">
3 <musicStore>
4 <storeDetails>
5 <storeName>Marga Music Store</storeName>
6 <owner title="Mr.">Alisha Shrestha</owner>
  <panNo>1983474123</panNo>
7 
  
8 <address city="Dharan" province="1">Laxmi Chowk</address>

```



Figure 15 Evidence of test case 7

5.8. Pattern for Track length



Table 9 Test case 8

Objective	To check the pattern of track length element.
Action	Enter wrong pattern for track length element.
Expected Result	The XML validator should display an error message.
Actual Result	The XML validator displayed an error message.
Conclusion	Test has been successfully completed.

2 errors have been found!

Click on  to jump to the error. In the document, you can point at  with your mouse to see the error message.

Errors in the XML document:

-  27:38 cvc-pattern-valid: Value '2:01:56' is not facet-valid with respect to pattern '[0-9]{1};[0-9]{2}' for type 'null'.
-  27:38 cvc-type.3.1.3: The value '2:01:56' of element 'trackLength' is not valid.

XML document:



```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE musicStore SYSTEM "catalog_19033571.dtd">
3 <musicStore>
4 <storeDetails>
5 <storeName>Marga Music Store</storeName>
6 <owner title="Mr.">Alisha Shrestha</owner>
7 <panNo>1983474</panNo>

```

Figure 16 First evidence of test case 6

```

21 <singers>
22 <singer gender="Male">Ray charles</singer>
23 </singers>
24 <featured> Margie Hendrix</featured>
25 <label>ABC-Paramount</label>
26 <releaseYear>1962</releaseYear>
   <trackLength>2:01:56</trackLength>
27 
   
28 <price>240</price>
29 </song>

```



Figure 17 Second evidence of test case 6

5.9. Pattern for Price

Table 10 Test case 9

Objective	To check the pattern of price element.
Action	Enter wrong pattern for price element.
Expected Result	The XML validator should display an error message.
Actual Result	The XML validator displayed an error message.
Conclusion	Test has been successfully completed.

2 errors have been found!

Click on  to jump to the error. In the document, you can point at  with your mouse to see the error message.

Errors in the XML document:

- ✖ 28:23 cvc-pattern-valid: Value '1240' is not facet-valid with respect to pattern '[0-9]{3}' for type 'null'.
- ✖ 28:23 cvc-type.3.1.3: The value '1240' of element 'price' is not valid.

XML document:

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE musicStore SYSTEM "catalog_19033571.dtd">
3 <musicStore>
4 <storeDetails>
5 <storeName>Marga Music Store</storeName>

```

Figure 18 First evidence of test case 6

```

20 </writers>
21 <singers>
22 <singer gender="Male">Ray charles</singer>
23 </singers>
24 <featured> Margie Hendrix</featured>
25 <label>ABC-Paramount</label>
26 <releaseYear>1962</releaseYear>
27 <trackLength>2:00</trackLength>
   <price>1240</price>
28 ✖
   ✖
29 </song>
30 <song genre="Jazz" coverpage="02" >

```



Figure 19 Second evidence of test case 6

5.10. Pattern for website

Table 11 Test case 10

Objective	To check the pattern of website element.
Action	Enter wrong pattern for website element.
Expected Result	The XML validator should display an error message.
Actual Result	The XML validator displayed an error message.
Conclusion	Test has been successfully completed.

2 errors have been found!

Click on  to jump to the error. In the document, you can point at  with your mouse to see the error message.

Errors in the XML document:

- ❌ 10:42 cvc-pattern-valid: Value 'www.margamusic01.com' is not facet-valid with respect to pattern '[a-z]*.[a-z]*.[a-z]*' for type 'null'.
- ❌ 10:42 cvc-type.3.1.3: The value 'www.margamusic01.com' of element 'website' is not valid.

XML document:

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE musicStore SYSTEM "catalog_19033571.dtd">
3 <musicStore>
4 <storeDetails>
5 <storeName>Marga Music Store</storeName>
6 <owner title="Mr.">Alisha Shrestha</owner>
7 <panNo>1983474</panNo>
8 <address city="Dharan" province="1">Laxmi Chowk</address>
9 <telephoneNo>025531724</telephoneNo>
  <website>www.margamusic01.com</website>
10 ❌
  ❌
11 <logo>Marga</logo>
12 </storeDetails>
13 <songs>
```

Figure 20 Evidence of test case 10

6. Development Process

Firstly, the development process was begun from gathering the requirement and analysis to develop a well-defined structure of XML. Various researches and finding was carried out for including elements, data and attributes which were required in the scenario. As a second process of the development, document structure well known as tree diagram was drawn using a development tool called “Draw.io” whereas the development tools that were used for the completion of the whole project was Sublime Text Editor, Online XML and CSS Validator.

Draw.io for Tree Diagram

As many things in life and technology related life, cost not always equals quality because this tool has been around for some while, and it has evolve to be an easy to use and powerful tool that allows any type of diagram, its ability to integrate with most of the important cloud storage platforms gives a big plus to the experience. Draw.io is among the best in the art software industry for free online solutions. With this particular app, you are able to do precisely all the things you need as addition Draw.io interface is familiar to everyone that knows about art and presentation software. It is especially created for drawing diagrams. This software was used for the preparation of the tree diagram as every component is easy available which are required for the tree diagram.

Sublime text Editor for XML, DTD, Schema and CSS

Sublime Text has been around for over a decade, making it a powerhouse within the industry. A Text Editor allows you to make the most out of your coding time. Especially with Sublime, it strips away the excess while giving you an environment that allows you to code. The editor is primarily known for its speed and its reliability, which comes from having a solid development team. Sublime Text Editor 3.0 was used of creating XML, Schema, DTD and CSS.

XML Validator for XML, Schema, and DTD

Validation is a process by which an XML document is validated. An XML document is said to be valid if its contents match with the elements, attributes and associated document type declaration (DTD), and if the document complies with the constraints expressed in it. It's good to develop an eye for clean XML code, which makes it possible to develop XML documents with a minimal amount of errors. However, it's difficult for any human to perform such a technical task flawlessly, which is where XML validation tools come into play. In this project XML validator tool is used to analyze the contents of XML documents to make sure they conform to a schema.

7. Critical Analysis

The overall development of the static webpage was quite challenging but fun at the same time. The report is entirely focused upon the development of the system and while writing the entire report numerous discoveries came across which helped in gaining more knowledge mainly on the functionality. To have a complete successful system many discussion and analysis were performed before starting the project.

The major challenges faced during the coursework were designing the system using restriction in schema, more precisely uses with difference aspects of restriction. This task helped to learn many terms related to XML and while performing this task gained a lot of experience regarding the development process. Although many difficulties were arise doing the coursework as it was done during the lockdown period but with the proper help of the teacher and with some of friend, made the work much easier.

The coursework kept busy even in this pandemic situation which eventually helped to reduce the dullness that is faced by many students during the lockdown. Many researches were carried out for the coursework because of the lack of direct interaction with the teacher and that also comes under a good point as it improved a lot in research skills.

There was different scenario provided in the coursework which was really interesting part throughout the task. Learning and doing coursework for XML, DTD and Scheme

was a new experience but CSS was much familiar as it was practiced during the coursework provided in first year. The designing part was time consuming as well as the most interesting portion during the whole task but implementing the CSS in XML was quite difficult thing which required additional research. While writing the XML many new terms got familiar and many difficulties were come across every step but were soon overcome. Moving ahead from this problem various deep research especially regarding schema was carried out

During the development of the tree diagram the most difficult part encountered was maintaining the modifying symbols in required element but proper guidance from the module leader left with zero confusion. The whole task was performed based on the real-life scenario for a music store and researching about the songs that are to be used for data was an interesting part.

Along with the completion of the system, knowledge related to different technologies such as XML, CSS, DTD, Schema along with real time implementation of the combined technologies were learned. The coursework was completed successfully with a good research skill, regular practice and great labor.

Conclusion

The overall documentation was about all the work done for developing a static webpage for the music store. The coursework was firstly started by carrying out researches regarding different relevant topics used in the task and evaluating the time for the completion of the whole task. The development of the tree diagram was the first and foremost step to be done as XML was generated according to the diagram.

While constructing the data structure everything was going fine but maintaining the modifying symbols according to the requirement of the scenario was difficult. The requirement of the coursework was fulfilled by including data, attributes and optional elements. The XML document was created using the data structure in sublime text editor along with DTD and Schema.

DTD was created to define the data structure of the XML documentation whereas Schema was used to describe the structure and organize the data of the XML

documentation. The schema was created following flat catalogue structure, the reason for choosing flat catalogue is that it was the first type taught in the class and was interested from the first day as it was easy to use and was familiar. CSS was used to display the contents of the XML document in a clear and precise manner as it also makes the design attractive and more presentable for the user.

The documentation of this coursework also includes testing process of the program which includes the validation of the system using XML validation Tool. The test cases are tested and presented in the test section of the documentation. With the writing of the Critical analysis the coursework comes to the end by leaving various knowledge regarding different topics which were practiced throughout the coursework.

The successful completion of the entire project assigned as the individual coursework was only possible within the given time limit with lots of hardship, researches, time management and most importantly guidance of the module leader. This coursework has created an opportunity for each individual to implement one's knowledge in practical field.

8. Bibliography

geeksforgeeks. (n.d.) *www.geeksforgeeks.org* [Online]. Available from: <https://www.geeksforgeeks.org/displaying-xml-using-css/#:~:text=CSS%20can%20be%20used%20to,style%20to%20whole%20XML%20document.&text=Define%20the%20style%20rules%20for,%2C%20font%2Dweight%2C%20etc.>

InformIT. (2001) *InformIT* [Online]. Available from: <https://www.informit.com/articles/article.aspx?p=24614&seqNum=3#:~:text=The%20critical%20difference%20between%20DTDs,syntax%20itself%20is%20quite%20terse.>

Java T point. (n.d.) *www.javatpoint.com* [Online]. Available from: <https://www.javatpoint.com/xml-schema.>

Roche, E. (2000) *Harvard Business Review Home* [Online]. Available from: <https://hbr.org/2000/07/explaining-xml#>.

tutorialspoint. (n.d.) *www.tutorialspoint.com* [Online]. Available from: https://www.tutorialspoint.com/xml/xml_dtds.htm.