



Module Code & Module Title

CS5004NT Emerging Programming Platforms and Technologies

Assessment Weightage & Type

30% Individual Coursework

Year and Semester

2021 Autumn/Spring

Student Name: Kushal Shrestha

London Met ID: 19031673

College ID: np05cp4a190072

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.

Table of Contents

1. Introduction	1
2. Tree Diagram	3
3. Development.....	4
3.1. XML.....	4
3.2. DTD.....	10
3.3. Schema.....	11
3.4. CSS.....	15
4. Difference between Schema and DTD.....	24
5. Testing	26
5.1. Well-formed.....	26
5.2. Validating XML Document with DTD and Schema	28
5.3. CSS.....	33
5.4. Patterns.....	35
5.5. Enumeration.....	37
6. Development process	39
7. Critical Analysis and Conclusion	41
8. Bibliography	44

Table of Figures

Figure 1 Tree structure of Music store.....	3
Figure 2 Tree structure of XML file	26
Figure 3 Screenshot of coping XML file in website.....	27
Figure 4 Screenshot of message displayed for XML file	28
Figure 5 Screenshot of XML file coping.....	29
Figure 6 Screenshot of coping DTD file.....	29
Figure 7 Screenshot of message displayed for XML and DTD file	30
Figure 8 Screenshot of coping XML file and adding external Schema file	31
Figure 9 Screenshot of coping DTD file.....	31
Figure 10 Screenshot of coping XML Schema file	32
Figure 11 Screenshot of message displayed for XML, DTD and Schema file	32
Figure 12 Screenshot of uploading CSS file.....	33
Figure 13 Screenshot of message displayed CSS file.....	33
Figure 14 Screenshot of opening the XML file in Chrome browser	34
Figure 15 Screenshot of error message for telephone number element	35
Figure 16 Screenshot of error message for website address element	36
Figure 17 Screenshot of error message for postal code attribute.....	37
Figure 18 Screenshot of error message for genre attribute	38

Table of Tables

Table 1 Difference between Schema and DTD	24
Table 2 Test case for checking XML file is well formed in browser	26
Table 3 Test case for checking XML file is well formed in website	27
Table 4 Test case for validating XML file with DTD file	28
Table 5 Test case for validation XML file with DTD and Schema file	30
Table 6 Test case for validating CSS file in website	33
Table 7 Test case for validating XML with CSS file	34
Table 8 Test case for checking the pattern for telephone number element	35
Table 9 Test case for checking the pattern for website address element	36
Table 10 Test case for checking the enumeration for postal code attribute	37
Table 11 Test case for checking the enumeration for genre attribute	38

1. Introduction

The whole report is based upon the individual coursework provided by the Emerging programming platforms and technologies module as second coursework. In this task, a model system for the Music store is developed by including all the details of the store. Now describing the store, it is a very notable store for its assortment of songs for very long. The store has been incorporated with various genres of songs and is been providing the in a very reasonable price. Based on the given scenario the structure is designed and concentrating in the module every necessary component of the XML is utilized appropriately. The basic purpose of the use of XML was specially designed to store and transport data.

The Extensible Markup Language (XML) is a simple text-based format for representing structured information: documents, data, configuration, books, transactions, invoices, and much more. It was derived from an older standard format called SGML in order to be more suitable for Web use. It is used widely for sharing information structure between programs, between people, and between computers. If you are already familiar with HTML, you can see that XML is very similar. However, the syntax rules of XML are strict. (W3, 2021)

The coursework used XML because of its readability and the presence of element and attribute names in XML means that people looking at an XML document can often get a head start on understanding the format.

Document Type Definition (DTD) defines the tags and attributes used in an XML or HTML document. Any elements defined in a DTD can be used in these documents, along with the predefined tags and attributes that are part of each markup language.

An XML DTD can be either specified inside the document, or it can be kept in a separate document and then linked separately. There are two types of DTD and they are Internal and External DTD, for this coursework External DTD. Here internal DTD refers to the elements which are declared inside the XML files whereas external DTD means the elements are declared outside the XML file. (Tech Terms, 2021)

An XML Schema is a language for expressing constraints about XML documents and it is an alternative to DTD. An XML document is considered “well-formed” and “valid” if it is successfully validated against XML Schema. Information in schema documents is often used by XML-aware editing systems so that they can offer users the most likely elements to occur at any given location in a document. (Singh, 2021)

The reason for using schema in the course work is to provide a list of elements and attributes in a vocabulary, to associate types, such as integer, string, etc., or more specifically with values found in documents, and most importantly to provide documentation that is both human-readable and machine-processable. The schema uses XML as language so you don't have to learn new syntax.

Cascading Style Sheets (CSS) is a declarative language that controls how web pages look in the browser. The browser applies CSS-style declarations to selected elements to display them properly. A style declaration contains the properties and their values, which determine how a webpage looks. CSS is a language for specifying how documents are presented to users and now moving forward to display the XML document with CSS it is necessary to know CSS Style and by preferring CSS it is easier to have text-centric contents in the File. (Padamkar, 2021) It is said that Some Style rules should be done to the XML document like styling font-size, color, margin, Font-weight which is followed by CSS. Once the CSS file is linked into the XML file, the file does append the style sheet in the XML document and displays the result in style format depending on the properties implemented on the CSS file. (Webplatform, 2021)

2. Tree Diagram

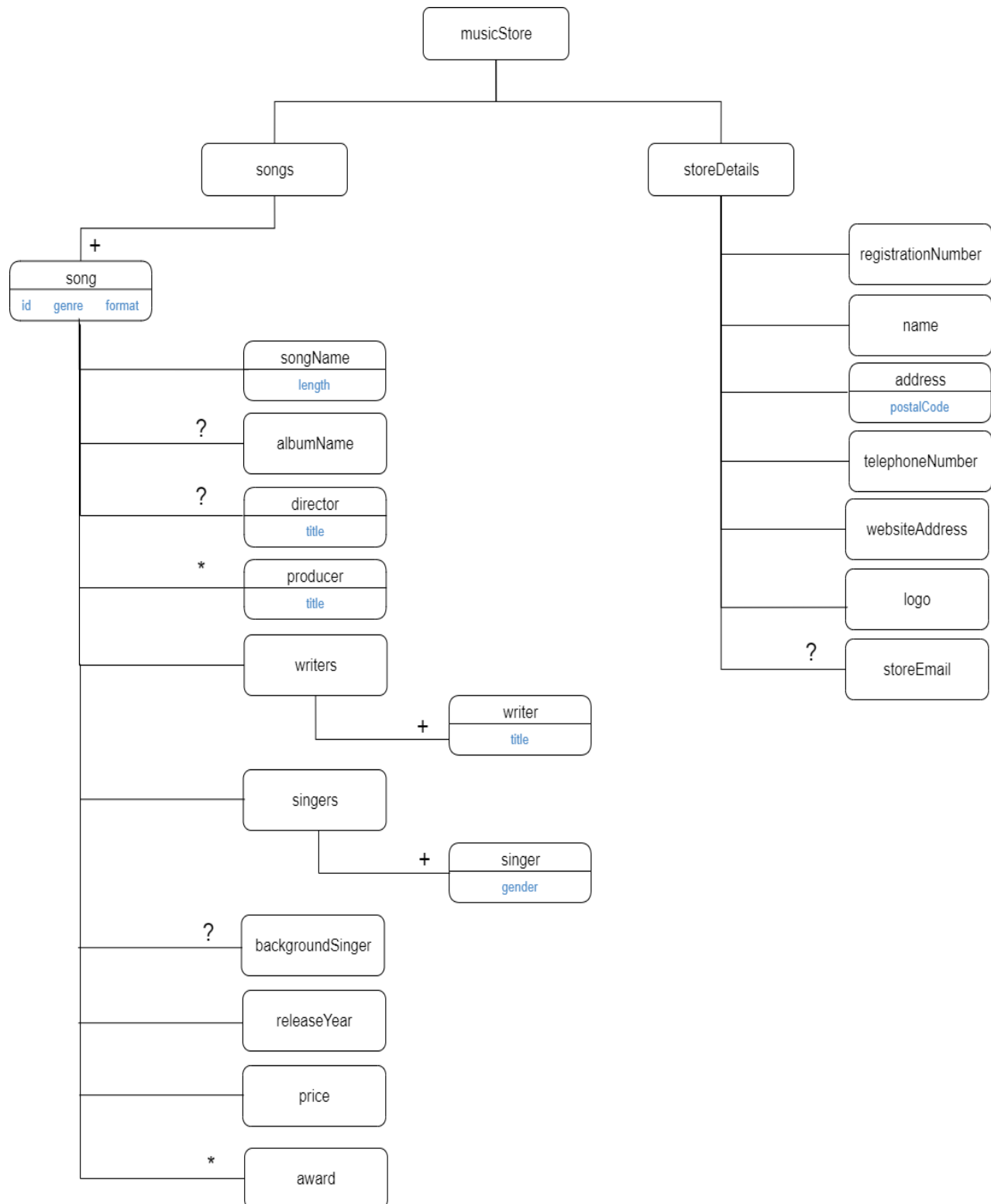


Figure 1 Tree structure of Music store

3. Development

3.1. XML

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <?xml-stylesheet type="text/css" href="catalog_19031673.css"?>
3. <!DOCTYPE musicStore SYSTEM "catalog_19031673.dtd">
4. <musicStore>
5.   <storeDetails>
6.     <registrationNumber>DH-99111632</registrationNumber>
7.     <name>Brother Music Store</name>
8.     <address postalCode="DH-56700">Dhrana-10, Sainik Chowk</address>
9.     <telephoneNumber>025-520420</telephoneNumber>
10.    <websiteAddress>www.brothermusicstore.com</websiteAddress>
11.    <logo></logo>
12.    <storeEmail>brotherstore12@gmail.com</storeEmail>
13.  </storeDetails>
14.  <songs>
15.    <song id="S-01" genre="Rap" format="Online">
16.      <songName length="5:23">"Lose Yourself"</songName>
17.      <producer title="Mr.">Eminem</producer>
18.      <producer title="Mr.">Luis Resto</producer>
19.      <producer title="Mr.">Jeff Bass</producer>
20.      <writers>
21.        <writer title="Mr.">Eminem</writer>
22.      </writers>
23.      <singers>
24.        <singer gender="Male">Eminem</singer>
25.      </singers>
26.      <releaseYear>2015</releaseYear>
27.      <price>Rs 245</price>
28.      <award>Academy Award</award>
29.      <award>MTV Award</award>
30.      <award>Grammy Award</award>
31.    </song>
32.    <song id="S-02" genre="Jazz" format="Cassette">
33.      <songName length="3:15">"All The Things You Are"</songName>
34.      <albumName>Ella Fitzgerald</albumName>
35.      <director title="Mr.">Jerome kern</director>
36.      <producer title="Mr.">Michael Cuscuna</producer>
37.      <writers>
```



```
38.         <writer title="Mr.">Oscar Hammerstein II</writer>
39.     </writers>
40.     <singers>
41.         <singer gender="Male">Jerome Kern</singer>
42.     </singers>
43.     <releaseYear>1963</releaseYear>
44.     <price>Rs 130</price>
45. </song>
46. <song id="S-03" genre="Pop" format="CD">
47.     <songName length="3:39">"Because of You"</songName>
48.     <albumName>Breakaway</albumName>
49.     <director title="Mr.">Vadim Perelman</director>
50.     <producer title="Mr.">David Hodges</producer>
51.     <writers>
52.         <writer title="Ms.">Kelly Clarkson</writer>
53.     </writers>
54.     <singers>
55.         <singer gender="Female">Kelly Clarkson</singer>
56.     </singers>
57.     <releaseYear>2005</releaseYear>
58.     <price>Rs 150</price>
59. </song>
60. <song id="S-04" genre="Jazz" format="CD">
61.     <songName length="3:10">"Round Midnight"</songName>
62.     <producer title="Mr.">George Avakian</producer>
63.     <writers>
64.         <writer title="Mr.">Thelonious Monk</writer>
65.     </writers>
66.     <singers>
67.         <singer gender="Female">Julie London</singer>
68.     </singers>
69.     <backgroundSinger>Jackie Paris</backgroundSinger>
70.     <releaseYear>1944</releaseYear>
71.     <price>Rs 130</price>
72.     <award>Academy Award</award>
73. </song>
74. <song id="S-05" genre="Rock" format="CD">
75.     <songName length="6:48">"Paradise City"</songName>
76.     <albumName>Appetite for Destruction</albumName>
77.     <director title="Mr.">Ash Avildsen</director>
78.     <producer title="Mr.">Mike Clink</producer>
```

```
79.         <writers>
80.             <writer title="Mr.">Axl Rose</writer>
81.             <writer title="Mr.">Izzy Strandin</writer>
82.             <writer title="Mr.">Steven Adler</writer>
83.         </writers>
84.         <singers>
85.             <singer gender="Male">Axl Rose</singer>
86.         </singers>
87.         <releaseYear>1989</releaseYear>
88.         <price>Rs 190</price>
89.     </song>
90.     <song id="S-06" genre="Rock" format="CD">
91.         <songName length="6:28">Nothing Else Matter</songName>
92.         <albumName>Metallica</albumName>
93.         <producer title="Mr.">James Hetfield Lars Ulrich</producer>
94.         <writers>
95.             <writer title="Mr.">James Hetfield Lars Ulrich</writer>
96.         </writers>
97.         <singers>
98.             <singer gender="Male">James Hetfield Lars Ulrich</singer>
99.         </singers>
100.        <releaseYear>1992</releaseYear>
101.        <price>Rs 135</price>
102.    </song>
103.    <song id="S-07" genre="Pop" format="CD">
104.        <songName length="3:48">All I Need</songName>
105.        <albumName>Moon Safari</albumName>
106.        <producer title="Mr.">Glen Ballard Clif Magness</producer>
107.        <writers>
108.            <writer title="Mr.">Nicolas Godin</writer>
109.        </writers>
110.        <singers>
111.            <singer gender="Male">Jack Wagner</singer>
112.        </singers>
113.        <releaseYear>1984</releaseYear>
114.        <price>Rs 300</price>
115.    </song>
116.    <song id="S-08" genre="Country Music" format="Online">
117.        <songName length="3:57">Shallow</songName>
118.        <producer title="Mr.">Ben Rice</producer>
119.        <writers>
```

```
120.         <writer title="Mr.">Andrew Wyatt</writer>
121.         <writer title="Mr.">Mark Ronson</writer>
122.     </writers>
123.     <singers>
124.         <singer gender="Male">Bradley Cooper</singer>
125.         <singer gender="Female">Lady Gaga</singer>
126.     </singers>
127.     <releaseYear>2018</releaseYear>
128.     <price>Rs 250</price>
129.     <award>Academy Award</award>
130. </song>
131. <song id="S-09" genre="Childrens Music" format="Online">
132.     <songName length="4:02">"A Whole New World"</songName>
133.     <director title="Mr.">Alan Menken</director>
134.     <producer title="Mr.">Alan Menken</producer>
135.     <producer title="Mr.">Tim Rice</producer>
136.     <writers>
137.         <writer title="Mr.">Tim Rice</writer>
138.     </writers>
139.     <singers>
140.         <singer gender="Male">Zayn Malik</singer>
141.         <singer gender="Female">Zhavia Ward</singer>
142.     </singers>
143.     <releaseYear>2018</releaseYear>
144.     <price>Rs 300</price>
145. </song>
146. <song id="S-10" genre="Rock" format="Online">
147.     <songName length="6:38">"Time Waits for No one"</songName>
148.     <albumName>It's Only Rock'n Roll</albumName>
149.     <director title="Mr.">Mamoru Hosoda's</director>
150.     <producer title="Mr.">Dave Clark</producer>
151.     <writers>
152.         <writer title="Mr.">Roger Hodgson</writer>
153.         <writer title="Mr.">Clifford Dinsmore</writer>
154.     </writers>
155.     <singers>
156.         <singer gender="Male">Freddie Mercury</singer>
157.     </singers>
158.     <backgroundSinger>Roger Hodgson</backgroundSinger>
159.     <releaseYear>1974</releaseYear>
160.     <price>Rs 240</price>
```

```
161.         </song>
162.         <song id="S-11" genre="Rock" format="Cassette">
163.             <songName length="3:01">"Jump In My Car"</songName>
164.             <albumName>Here We are</albumName>
165.             <producer title="Mr.">Harry Vanda</producer>
166.             <writers>
167.                 <writer title="Mr.">Ted Mulry</writer>
168.             </writers>
169.             <singers>
170.                 <singer gender="Male">Harry Vanda</singer>
171.             </singers>
172.             <backgroundSinger>Les Hall</backgroundSinger>
173.             <releaseYear>1974</releaseYear>
174.             <price>Rs 290</price>
175.         </song>
176.         <song id="S-12" genre="Dubset" format="Online">
177.             <songName length="4:59">"Like This"</songName>
178.             <director title="Mr.">Jeremy Sazon</director>
179.             <writers>
180.                 <writer title="Mr.">Jeremy Sazon</writer>
181.             </writers>
182.             <singers>
183.                 <singer gender="Male">Jeremy Sazon</singer>
184.             </singers>
185.             <releaseYear>2019</releaseYear>
186.             <price>Rs 250</price>
187.         </song>
188.         <song id="S-13" genre="Country Music" format="Online">
189.             <songName length="4:03">"Drunk"</songName>
190.             <albumName>And I dont Wanna Go Home</albumName>
191.             <producer title="Mr.">Brandon Paddock</producer>
192.             <writers>
193.                 <writer title="Mrs.">Ellen King</writer>
194.             </writers>
195.             <singers>
196.                 <singer gender="Female">Elle King</singer>
197.                 <singer gender="Female">Miranda Lambert</singer>
198.             </singers>
199.             <releaseYear>2021</releaseYear>
200.             <price>Rs 300</price>
201.         </song>
```

```
202.         <song id="S-14" genre="Country Music" format="Online">
203.             <songName length="6:43">"Brad Paisley"</songName>
204.             <producer title="Ms.">Alison Krauss</producer>
205.             <writers>
206.                 <writer title="Mr.">Jon Randall</writer>
207.             </writers>
208.             <singers>
209.                 <singer gender="Male">Whiskey Lullaby</singer>
210.             </singers>
211.             <backgroundSinger>Alison Krauss</backgroundSinger>
212.             <releaseYear>2010</releaseYear>
213.             <price>Rs 320</price>
214.         </song>
215.         <song id="S-15" genre="Rap" format="CD">
216.             <songName length="4:17">"Mockingbird"</songName>
217.             <writers>
218.                 <writer title="Mr.">Eminem</writer>
219.             </writers>
220.             <singers>
221.                 <singer gender="Male">Eminem</singer>
222.             </singers>
223.             <releaseYear>2009</releaseYear>
224.             <price>Rs 265</price>
225.         </song>
226.     </songs>
227. </musicStore>
228.
```

3.2. DTD

1. <!ELEMENT musicStore (storeDetails,songs)>
2. <!ELEMENT storeDetails
(registrationNumber,name,address,telephoneNumber,websiteAddress,logo,storeEmail?)>
3. <!ELEMENT registrationNumber (#PCDATA)>
4. <!ELEMENT name (#PCDATA)>
5. <!ELEMENT address (#PCDATA)>
6. <!ELEMENT telephoneNumber (#PCDATA)>
7. <!ELEMENT websiteAddress (#PCDATA)>
8. <!ELEMENT logo (#PCDATA)>
9. <!ELEMENT storeEmail (#PCDATA)>
- 10.
11. <!ATTLIST address postalCode CDATA #REQUIRED>
- 12.
13. <!ELEMENT songs (song+)>
14. <!ELEMENT song
(songName,albumName?,director?,producer*,writers,singers,backgroundSinger?,releaseYear,price,award*)>
15. <!ELEMENT songName (#PCDATA)>
16. <!ELEMENT albumName (#PCDATA)>
17. <!ELEMENT director (#PCDATA)>
18. <!ELEMENT producer (#PCDATA)>
19. <!ELEMENT writers (writer+)>
20. <!ELEMENT writer (#PCDATA)>
21. <!ELEMENT singers (singer+)>
22. <!ELEMENT singer (#PCDATA)>
23. <!ELEMENT backgroundSinger (#PCDATA)>
24. <!ELEMENT releaseYear (#PCDATA)>
25. <!ELEMENT price (#PCDATA)>
26. <!ELEMENT award (#PCDATA)>
- 27.
28. <!ATTLIST song id CDATA #REQUIRED>
29. <!ATTLIST song genre CDATA #REQUIRED>
30. <!ATTLIST song format (CD|Cassette|Online) #IMPLIED>
31. <!ATTLIST songName length CDATA #REQUIRED>
32. <!ATTLIST director title (Mr.|Ms.|Mrs.) #IMPLIED>
33. <!ATTLIST producer title (Mr.|Ms.|Mrs.) #IMPLIED>
34. <!ATTLIST writer title (Mr.|Ms.|Mrs.) #IMPLIED>
35. <!ATTLIST director title (Mr.|Ms.|Mrs.) #IMPLIED>
36. <!ATTLIST singer gender (Male|Female) #REQUIRED>

3.3. Schema

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
3.   <xs:element name="musicStore">
4.     <xs:complexType>
5.       <xs:sequence>
6.         <xs:element name="storeDetails" type="storeType"/>
7.         <xs:element name="songs" type="songsType"
maxOccurs="unbounded"/>
8.       </xs:sequence>
9.     </xs:complexType>
10.  </xs:element>
11.
12.  <xs:complexType name="storeType">
13.    <xs:sequence>
14.      <xs:group ref="storeGroup"/>
15.    </xs:sequence>
16.  </xs:complexType>
17.
18.  <xs:group name="storeGroup">
19.    <xs:sequence>
20.      <xs:element name="registrationNumber" type="xs:string"/>
21.      <xs:element name="name" type="xs:string"/>
22.      <xs:element name="address" type="addressType"/>
23.      <xs:element name="telephoneNumber" type="telephoneNumberType"/>
24.      <xs:element name="websiteAddress" type="websiteAddressType"/>
25.      <xs:element name="logo" type="xs:string"/>
26.      <xs:element name="storeEmail" type="storeEmailType"/>
27.    </xs:sequence>
28.  </xs:group>
29.
30.  <xs:complexType name="addressType">
31.    <xs:simpleContent>
32.      <xs:extension base="xs:string">
33.        <xs:attributeGroup ref="codeType"/>
34.      </xs:extension>
35.    </xs:simpleContent>
36.  </xs:complexType>
37.
38.  <xs:attributeGroup name="codeType">
```

```
39.         <xs:attribute name="postalCode" type="postalCodeType"/>
40.     </xs:attributeGroup>
41.
42.
43.     <xs:simpleType name="postalCodeType">
44.         <xs:restriction base="xs:string">
45.             <xs:enumeration value="DH-56700"/>
46.         </xs:restriction>
47.     </xs:simpleType>
48.
49.     <xs:simpleType name="telephoneNumberType">
50.         <xs:restriction base="xs:string">
51.             <xs:pattern value="[0-9]{3}-[0-9]{6}"/>
52.         </xs:restriction>
53.     </xs:simpleType>
54.
55.     <xs:simpleType name="websiteAddressType">
56.         <xs:restriction base="xs:string">
57.             <xs:pattern value="[a-z]*.[a-z]*.[a-z]*/>
58.         </xs:restriction>
59.     </xs:simpleType>
60.
61.     <xs:simpleType name="storeEmailType">
62.         <xs:restriction base="xs:string">
63.             <xs:pattern value="[a-z0-9]*@[a-z]*.[a-z]*/>
64.         </xs:restriction>
65.     </xs:simpleType>
66.
67.     <xs:complexType name="songsType">
68.         <xs:sequence>
69.             <xs:group ref="songsGroup"/>
70.         </xs:sequence>
71.     </xs:complexType>
72.
73.     <xs:group name="songsGroup">
74.         <xs:sequence>
75.             <xs:element name="song" type="songType" maxOccurs="unbounded"/>
76.         </xs:sequence>
77.     </xs:group>
78.
```



```
79. <xs:complexType name="songType">
80.     <xs:sequence>
81.         <xs:group ref="songGroup"/>
82.     </xs:sequence>
83.     <xs:attributeGroup ref="songAttGroup"/>
84. </xs:complexType>
85.
86. <xs:group name="songGroup">
87.     <xs:sequence>
88.         <xs:element name="songName" type="songNameType"/>
89.         <xs:element name="albumName" type="xs:string" minOccurs="0"/>
90.         <xs:element name="director" type="directorType" minOccurs="0"/>
91.         <xs:element name="producer" type="producerType" minOccurs="0"
maxOccurs="unbounded"/>
92.         <xs:element name="writers" type="writersType"/>
93.         <xs:element name="singers" type="singersType"/>
94.         <xs:element name="backgroundSinger" type="xs:string" minOccurs="0"/>
95.         <xs:element name="releaseYear" type="xs:string"/>
96.         <xs:element name="price" type="xs:string"/>
97.         <xs:element name="award" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
98.     </xs:sequence>
99. </xs:group>
100.
101. <xs:attributeGroup name="songAttGroup">
102.     <xs:attribute name="id" type="xs:string"/>
103.     <xs:attribute name="genre" type="genreType"/>
104.     <xs:attribute name="format" type="xs:string"/>
105. </xs:attributeGroup>
106.
107. <xs:simpleType name="genreType">
108.     <xs:restriction base="xs:string">
109.         <xs:enumeration value="Rap"/>
110.         <xs:enumeration value="Jazz"/>
111.         <xs:enumeration value="Pop"/>
112.         <xs:enumeration value="Rock"/>
113.         <xs:enumeration value="Country Music"/>
114.         <xs:enumeration value="Childrens Music"/>
115.         <xs:enumeration value="Dubset"/>
116.     </xs:restriction>
117. </xs:simpleType>
```

```
118.
119. <xs:complexType name="songNameType">
120.     <xs:simpleContent>
121.         <xs:extension base="xs:string">
122.             <xs:attribute name="length" type="xs:string" use="required"/>
123.         </xs:extension>
124.     </xs:simpleContent>
125. </xs:complexType>
126.
127. <xs:complexType name="directorType">
128.     <xs:simpleContent>
129.         <xs:extension base="xs:string">
130.             <xs:attribute name="title" type="xs:string" use="optional"/>
131.         </xs:extension>
132.     </xs:simpleContent>
133. </xs:complexType>
134.
135. <xs:complexType name="producerType">
136.     <xs:simpleContent>
137.         <xs:extension base="xs:string">
138.             <xs:attribute name="title" type="xs:string" use="optional"/>
139.         </xs:extension>
140.     </xs:simpleContent>
141. </xs:complexType>
142.
143. <xs:complexType name="writersType">
144.     <xs:sequence>
145.         <xs:element name="writer" type="writerType" maxOccurs="unbounded"/>
146.     </xs:sequence>
147. </xs:complexType>
148.
149. <xs:complexType name="writerType">
150.     <xs:simpleContent>
151.         <xs:extension base="xs:string">
152.             <xs:attribute name="title" type="xs:string"/>
153.         </xs:extension>
154.     </xs:simpleContent>
155. </xs:complexType>
156.
157. <xs:complexType name="singersType">
158.     <xs:sequence>
```

```
159.         <xs:element name="singer" type="singerType" maxOccurs="unbounded"/>
160.     </xs:sequence>
161. </xs:complexType>
162.
163. <xs:complexType name="singerType">
164.     <xs:simpleContent>
165.         <xs:extension base="xs:string">
166.             <xs:attribute name="gender" type="xs:string"/>
167.         </xs:extension>
168.     </xs:simpleContent>
169. </xs:complexType>
170.</xs:schema>
171.
```

3.4. CSS

```
1.  * {
2.    background-color: rgb(208, 208, 208);
3.    display: block;
4.    font-family: sans-serif;
5.  }
6.
7.  logo {
8.    float: none;
9.    background: url(music.png) no-repeat right;
10.   background-size: 130px;
11.   width: 1327px;
12.   height: 200px;
13.   position: absolute;
14.   top: -10px;
15. }
16.
17. registrationNumber {
18.   color: rgb(11, 108, 154);
19.   font-family: serif;
20.   font-size: 13px;
21.   text-align: right;
22.   margin-right: 30px;
23.   margin-top: 15px;
24. }
```

```
25. registrationNumber::before {
26.   content: "Reg No: ";
27. }
28.
29. name {
30.   color: rgb(155, 54, 46);
31.   font-family: sans-serif;
32.   font-weight: bold;
33.   font-size: 40px;
34.   text-align: center;
35. }
36.
37. address, telephoneNumber {
38.   font-family: sans-serif;
39.   font-style: italic;
40.   font-size: 14px;
41.   text-align: center;
42.   padding-top: 2px;
43. }
44.
45. address::before {
46.   content: "(" attr(postalCode) " ";
47. }
48.
49. telephoneNumber::before {
50.   content: "Tel ph: ";
51. }
52.
53. websiteAddress {
54.   color: rgb(11, 108, 154);
55.   font-family: serif;
56.   font-size: 13px;
57.   text-align: center;
58. }
59.
60. storeEmail {
61.   color: rgb(11, 108, 154);
62.   font-family: serif;
63.   font-size: 13px;
64.   position: absolute;
65.   top: 155px;
```

```
66.   right: 10px;
67. }
68.
69. storeEmail::before {
70.   content: "Email: ";
71. }
72.
73. songs {
74.   border: 2px solid;
75.   border-radius: 20px;
76.   background: rgb(224, 224, 209);
77.   height: 1800px;
78.   margin-left: 8px;
79.   margin-right: 8px;
80.   margin-bottom: 8px;
81.   margin-top: 45px;
82. }
83.
84. song[id="S-01"] {
85.   border: 2px solid;
86.   border-radius: 20px;
87.   position: absolute;
88.   height: 260px;
89.   width: 300px;
90.   top: 250px;
91.   left: 100px;
92.   padding-top: 40px;
93.   padding-left: 40px;
94. }
95.
96. song[id="S-02"] {
97.   border: 2px solid;
98.   border-radius: 20px;
99.   position: absolute;
100.  height: 260px;
101.  width: 300px;
102.  top: 250px;
103.  left: 480px;
104.  padding-top: 40px;
105.  padding-left: 40px;
106.}
```

```
107.  
108.song[id="S-03"] {  
109.  border: 2px solid;  
110.  border-radius: 20px;  
111.  position: absolute;  
112.  height: 260px;  
113.  width: 300px;  
114.  top: 250px;  
115.  left: 860px;  
116.  padding-top: 40px;  
117.  padding-left: 40px;  
118.}  
119.  
120.song[id="S-04"] {  
121.  border: 2px solid;  
122.  border-radius: 20px;  
123.  position: absolute;  
124.  height: 260px;  
125.  width: 300px;  
126.  top: 580px;  
127.  left: 100px;  
128.  padding-top: 40px;  
129.  padding-left: 40px;  
130.}  
131.  
132.song[id="S-05"] {  
133.  border: 2px solid;  
134.  border-radius: 20px;  
135.  position: absolute;  
136.  height: 260px;  
137.  width: 300px;  
138.  top: 580px;  
139.  left: 480px;  
140.  padding-top: 40px;  
141.  padding-left: 40px;  
142.}  
143.  
144.song[id="S-06"] {  
145.  border: 2px solid;  
146.  border-radius: 20px;  
147.  position: absolute;
```

```
148. height: 260px;
149. width: 300px;
150. top: 580px;
151. left: 860px;
152. padding-top: 40px;
153. padding-left: 40px;
154.}
155.
156.song[id="S-07"] {
157. border: 2px solid;
158. border-radius: 20px;
159. position: absolute;
160. height: 260px;
161. width: 300px;
162. top: 910px;
163. left: 100px;
164. padding-top: 40px;
165. padding-left: 40px;
166.}
167.
168.song[id="S-08"] {
169. border: 2px solid;
170. border-radius: 20px;
171. position: absolute;
172. height: 260px;
173. width: 300px;
174. top: 910px;
175. left: 480px;
176. padding-top: 40px;
177. padding-left: 40px;
178.}
179.
180.song[id="S-09"] {
181. border: 2px solid;
182. border-radius: 20px;
183. position: absolute;
184. height: 260px;
185. width: 300px;
186. top: 910px;
187. left: 860px;
188. padding-top: 40px;
```

```
189. padding-left: 40px;
190.}
191.
192.song[id="S-10"] {
193.  border: 2px solid;
194.  border-radius: 20px;
195.  position: absolute;
196.  height: 260px;
197.  width: 300px;
198.  top: 1240px;
199.  left: 100px;
200.  padding-top: 40px;
201.  padding-left: 40px;
202.}
203.
204.song[id="S-11"] {
205.  border: 2px solid;
206.  border-radius: 20px;
207.  position: absolute;
208.  height: 260px;
209.  width: 300px;
210.  top: 1240px;
211.  left: 480px;
212.  padding-top: 40px;
213.  padding-left: 40px;
214.}
215.
216.song[id="S-12"] {
217.  border: 2px solid;
218.  border-radius: 20px;
219.  position: absolute;
220.  height: 260px;
221.  width: 300px;
222.  top: 1240px;
223.  left: 860px;
224.  padding-top: 40px;
225.  padding-left: 40px;
226.}
227.
228.song[id="S-13"] {
229.  border: 2px solid;
```



```
230. border-radius: 20px;
231. position: absolute;
232. height: 260px;
233. width: 300px;
234. top: 1570px;
235. left: 100px;
236. padding-top: 40px;
237. padding-left: 40px;
238.}
239.
240.song[id="S-14"] {
241. border: 2px solid;
242. border-radius: 20px;
243. position: absolute;
244. height: 260px;
245. width: 300px;
246. top: 1570px;
247. left: 480px;
248. padding-top: 40px;
249. padding-left: 40px;
250.}
251.
252.song[id="S-15"] {
253. border: 2px solid;
254. border-radius: 20px;
255. position: absolute;
256. height: 260px;
257. width: 300px;
258. top: 1570px;
259. left: 860px;
260. padding-top: 40px;
261. padding-left: 40px;
262.}
263.
264.songName {
265. color: rgb(183, 107, 46);
266. text-align: center;
267. padding-bottom: 20px;
268. font-family: 'ludcida Handwriting';
269. font-size: 25px;
270. font-style: italic;
```

```
271.}
272.
273.songName::after {
274.  content: " (" attr(length) ")" ;
275.  font-size: 10px;
276.  color: black;
277.}
278.
279.song::after {
280.  color: rgb(183, 107, 46);
281.  content: "Genre: " attr(genre) "";
282.  font-size: 12px;
283.  float: right;
284.  margin-right: 20px;
285.  border-style: dotted dashed solid double;
286.  padding: 3px 10px 3px 10px;
287.}
288.
289.albumName::before {
290.  content: "Album name: ";
291.  font-weight: bold;
292.}
293.
294.director::before {
295.  content: "Director: ";
296.  font-weight: bold;
297.}
298.
299.producer::before {
300.  content: "Producer: ";
301.  font-weight: bold;
302.}
303.
304.writer::before {
305.  content: "Writer: ";
306.  font-weight: bold;
307.}
308.
309.singer::before {
310.  content: "Singer: ";
311.  font-weight: bold;
```

```
312.}
313.
314.singer::after {
315.  content: " ("attr(gender) ")";
316.  font-weight: bold;
317.  font-size: 10px;
318.}
319.
320.backgroundSinger::before {
321.  content: "Backgorund Singer: ";
322.  font-weight: bold;
323.}
324.
325.releaseYear::before {
326.  content: "Release Year: ";
327.  font-weight: bold;
328.}
329.
330.price::before {
331.  content: "Price: ";
332.  font-weight: bold;
333.}
334.
335.award::before {
336.  content: "Award: ";
337.  font-weight: bold;
338.}
339.
340.albumName,director,producer,writer,singer {
341.  display: list-item;
342.}
343.
344.backgroundSinger,releaseYear,price, award {
345.  display: list-item;
346.}
347.
```

4. Difference between Schema and DTD

Table 1 Difference between Schema and DTD

S.N	DTD	Schema (XSD)
1.	Abbreviation: DTD stands for Document Type Definition.	Abbreviation: XSD stands for XML Schema Definition.
2.	About DTD: DTD is the declarations that define a document type of Standard Generalized Markup Language. In short, it is derived from SGML syntax.	About Schema: XSD describes the elements in an XML document. In short, it is written in XML.
3.	Reusability: DTD poorly supports reusability but it can be used by parameter entities so it is very limited for reusability.	Reusability: Schema can be reused by defining named types. It can create its own derived data types and reference multiple schemas from the same document.
4.	Namespace: It does not support namespace but it owns a set of a keyword for defining a schema.	Namespace: It supports namespace and also has its own set of namespaces and elements for defining the schema.
5.	Datatype support and validation: It does not support datatypes. It has only #PCDATA as the datatype for the elements. It has no such restrictions.	Datatype support and validation: It supports datatypes for elements and attributes. It provides a flexible set of datatypes i.e. primitive data type (string, decimal, float, boolean) and custom data types (complex type and simple type) so, It provides a multifold key cross-

		reference. It allows specific restriction on data
6.	Strong or weak type: DTD is a weak type. It does not validate the content of the data types.	Strong or weak type: Schema is a strong type. It defines the data type of a certain element even constrains it to within specific length or values also ensures that the data stores in the XML are accurate.
7.	Provide inline definitions: It allows the inline definition which is good for small files but for a large file, it is disadvantageous because as we pull content every time, we retrieve it from a schema that can lead to serious overhead for degrading the performance.	Provide inline definitions: It does not allow inline definitions.
8.	Uses of DTD: It is more suitable for small XML data and also it provides less control on XML structure. DTD is not extensible and doesn't define an order for child elements. It is easier to understand as compare to a schema.	Uses of Schema: It is used in large XML data like Web services and it provides more control on XML structure. Schema is extensible and defines an order for child elements. It is more complex than DTD because the notion of types adds an extra layer for confusion.

(Javatpoint, 2021)

5. Testing

5.1. Well-formed

Table 2 Test case for checking XML file is well formed in browser

Objective	To check the XML is well-formed in browser with tree structure.
+Action	Opening the XML file in google chrome without CSS file.
Expected Result	The chrome browser should display the XML file in tree structure.
Actual Result	The chrome browser displayed the XML file in tree structure without error.
Conclusion	Test successfully completed.

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

```
<!-- <?xml-stylesheet type="text/css" href="catalog_19031673.css"?>
<!DOCTYPE musicStore SYSTEM "catalog_19031673.dtd" -->
<musicStore>
  <storeDetails>
    <registrationNumber>DH-99111632</registrationNumber>
    <name>Brother Music Store</name>
    <address postalCode="DH-567000">Dhrana-10, Sainik Chowk</address>
    <telephoneNumber>025-520420</telephoneNumber>
    <websiteAddress>www.brothermusicstore.com</websiteAddress>
    <logo/>
    <storeEmail>brotherstore12@gmail.com</storeEmail>
  </storeDetails>
  <songs>
    <song id="S-01" genre="Rap" format="Online">
      <songName length="5:23">"Lose Yourself"</songName>
      <producer title="Mr.">Eminem</producer>
      <producer title="Mr.">Luis Resto</producer>
      <producer title="Mr.">Jeff Bass</producer>
      <writers>
        <writer title="Mr.">Eminem</writer>
      </writers>
      <singers>
        <singer gender="Male">Eminem</singer>
      </singers>
      <releaseYear>2015</releaseYear>
      <price>Rs 245</price>
      <award>Academy Award</award>
      <award>MTV Award</award>
      <award>Grammy Award</award>
    </song>
    <song id="S-02" genre="Jazz" format="Cassette">
      <songName length="3:15">"All The Things You Are"</songName>
      <albumName>Ella Fitzgerald</albumName>
      <director title="Mr.">Jerome kern</director>
      <producer title="Mr.">Michael Cuscuna</producer>
      <writers>
        <writer title="Mr.">Oscar Hammerstein II</writer>
      </writers>
    </song>
  </songs>
</musicStore>
```

Figure 2 Tree structure of XML file

Table 3 Test case for checking XML file is well formed in website

Objective	To check the XML is well-formed in xmlvalidation website.
Action	Copy the XML document in xmlvalidation website and click the validation button.
Expected Result	The xmlvalidation website should display a message with no error found
Actual Result	The xmlvalidation website displayed a message no error found.
Conclusion	Test successfully completed.

Please copy your XML document in here:

```
<?xml version="1.0" encoding="UTF-8"?>
<musicStore>
  <storeDetails>
    <registrationNumber>DH-99111632</registrationNumber>
    <name>Brother Music Store</name>
    <address postalCode="DH-567000">Dhrana-10, Sainik Chowk</address>
    <telephoneNumber>025-520420</telephoneNumber>
    <websiteAddress>www.brothermusicstore.com</websiteAddress>
    <logo></logo>
    <storeEmail>brotherstore12@gmail.com</storeEmail>
```

Or upload it:

No file chosen

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 Screenshot of coping XML file in website

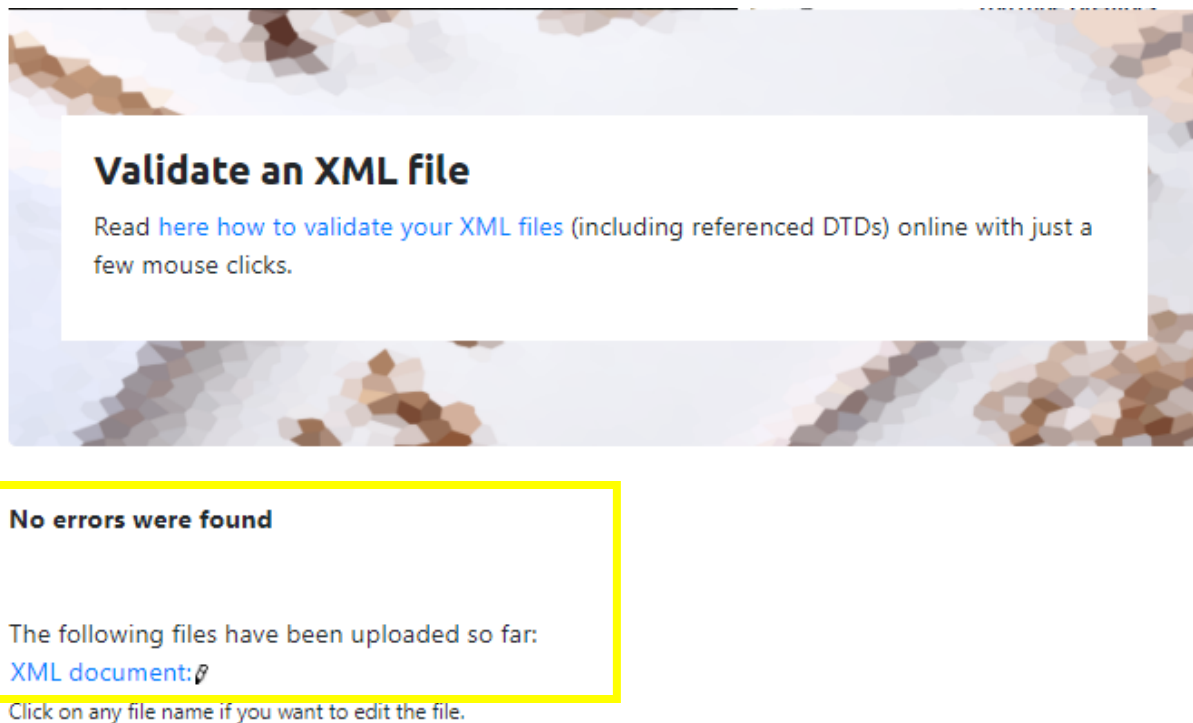


Figure 4 Screenshot of message displayed for XML file

5.2. Validating XML Document with DTD and Schema

Table 4 Test case for validating XML file with DTD file

Objective	To valid XML document with DTD file.
Action	Copy the XML document in xmlvalidation website, click the validation button and again copy the DTD file and click the continue validation button.
Expected Result	The xml validation website should not defect error from both XML and DTD file.
Actual Result	The xml validation website displayed no error found from both XML and DDT file.
Conclusion	Test successfully completed.

Please copy your XML document in here:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE musicStore SYSTEM "catalog_19031673.dtd">
<musicStore>
  <storeDetails>
    <registrationNumber>DH-99111632</registrationNumber>
    <name>Brother Music Store</name>
    <address postalCode="DH-567000">Dhrana-10, Sainik Chowk</address>
    <telephoneNumber>025-520420</telephoneNumber>
    <websiteAddress>www.brothermusicstore.com</websiteAddress>
    <logo></logo>
```

Or upload it:

No file chosen

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 Screenshot of XML file coping

The file catalog_19031673.dtd is being referenced. Please copy it in here, so that the validation can continue:

```
<!ELEMENT musicStore (storeDetails,songs)>
<!ELEMENT storeDetails
(registrationNumber,name,address,telephoneNumber,websiteAddress,logo,storeEmail?)>
<!ELEMENT registrationNumber (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address (#PCDATA)>
<!ELEMENT telephoneNumber (#PCDATA)>
<!ELEMENT websiteAddress (#PCDATA)>
<!ELEMENT logo (#PCDATA)>
<!ELEMENT storeEmail (#PCDATA)>
```

Or upload it:

No file chosen

The following files have been uploaded so far:

[XML document:](#)

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

Figure 6 Screenshot of coping DTD file

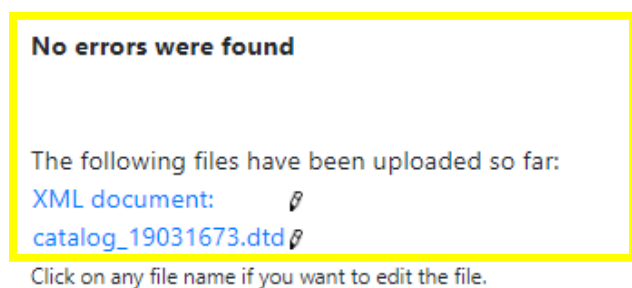
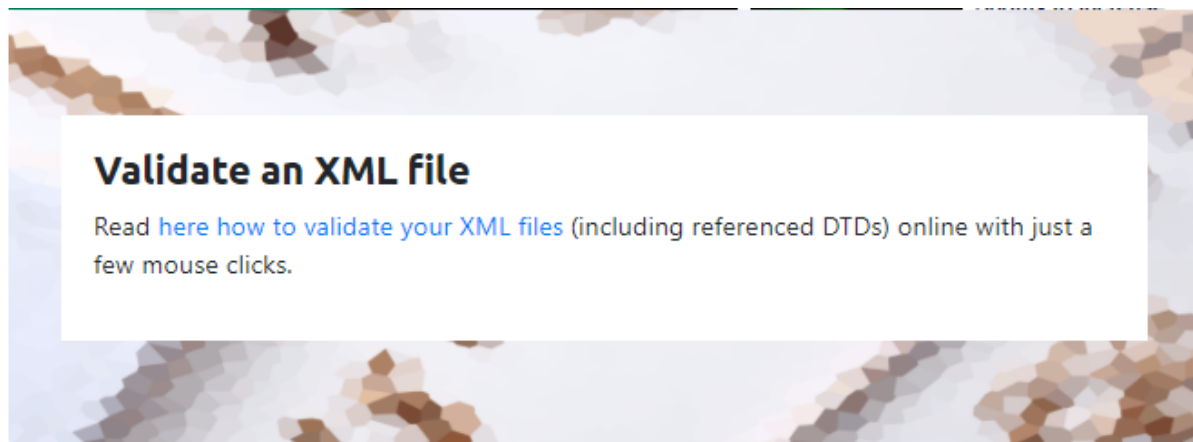


Figure 7 Screenshot of message displayed for XML and DTD file

Table 5 Test case for validation XML file with DTD and Schema file

Objective	To valid XML document with DTD and Schema file.
Action	Copy the XML document in xmlvalidation website with DTD file and click the continue validation button and lastly copy the schema file and click the continue validation button.
Expected Result	The xml validation website should not defect error from XML, DTD and Schema file.
Actual Result	The xml validation website displayed no error found from XML, DDT and Schema file.
Conclusion	Test successfully completed.

Please copy your XML document in here:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE musicStore SYSTEM "catalog_19031673.dtd">
<musicStore>
  <storeDetails>
    <registrationNumber>DH-99111632</registrationNumber>
    <name>Brother Music Store</name>
    <address postalCode="DH-567000">Dhrana-10, Sainik Chowk</address>
    <telephoneNumber>025-520420</telephoneNumber>
    <websiteAddress>www.brothermusicstore.com</websiteAddress>
    <logo></logo>
```

Or upload it:

No file chosen

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 Screenshot of coping XML file and adding external Schema file


The file catalog_19031673.dtd is being referenced. Please copy it in here, so that the validation can continue:

```
<!ELEMENT musicStore (storeDetails,songs)>
<!ELEMENT storeDetails
(registrationNumber,name,address,telephoneNumber,websiteAddress,logo,storeEmail?)>
<!ELEMENT registrationNumber (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address (#PCDATA)>
<!ELEMENT telephoneNumber (#PCDATA)>
<!ELEMENT websiteAddress (#PCDATA)>
<!ELEMENT logo (#PCDATA)>
<!ELEMENT storeEmail (#PCDATA)>
```

Or upload it:

No file chosen

The following files have been uploaded so far:

XML document: 

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

Figure 9 Screenshot of coping DTD file


Please copy the XML schema in here:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="musicStore">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="storeDetails" type="storeType"/>
        <xs:element name="songs" type="songsType" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

Or upload it:

No file chosen

The following files have been uploaded so far:

XML document: 

catalog_19031673.dtd 

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

Figure 10 Screenshot of coping XML Schema file

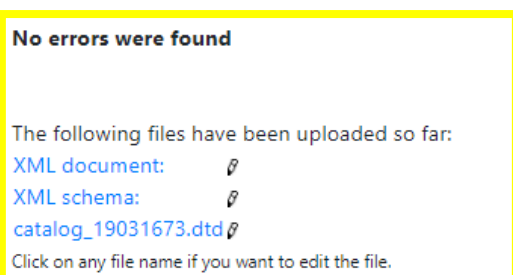
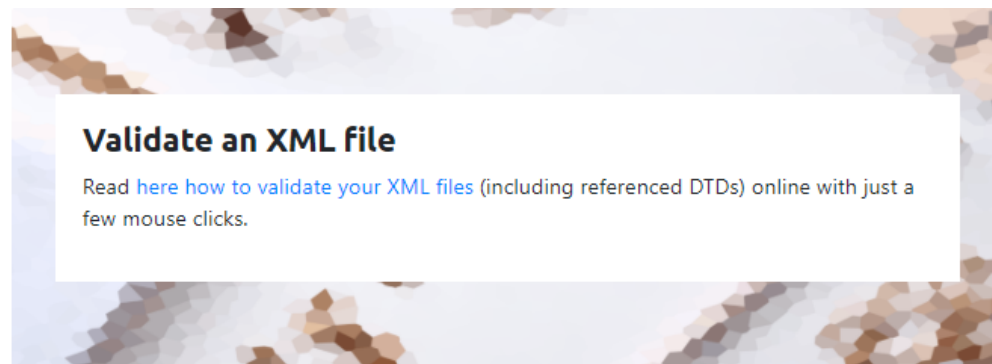


Figure 11 Screenshot of message displayed for XML, DTD and Schema file

5.3. CSS

Table 6 Test case for validating CSS file in website

Objective	To valid CSS file in online website.
Action	Upload the CSS file in the online website and click the check button. Website name: https://jigsaw.w3.org/css-validator/#validate_by_upload
Expected Result	The online website CSS validator should not defect error from CSS file.
Actual Result	The online website CSS validator displayed no error found from CSS file.
Conclusion	Test successfully completed.

W3C CSS Validation Service
Check Cascading Style Sheets (CSS) and (X)HTML documents with style sheets

By URI By file upload By direct input

Validate by file upload

Choose the document you would like validated:

Local CSS file:

More Options

Check

Figure 12 Screenshot of uploading CSS file

W3C The W3C CSS Validation Service
W3C CSS Validator results for catalog_19031673.css (CSS level 3 + SVG)

Jump to: Validated CSS

W3C CSS Validator results for catalog_19031673.css (CSS level 3 + SVG)

Congratulations! No Error Found.

This document validates as [CSS level 3 + SVG](#)!

Figure 13 Screenshot of message displayed CSS file

Table 7 Test case for validating XML with CSS file

Objective	To validate the XML file with CSS file
Action	Open the XML file linked with CSS file in the chrome browser.
Expected Result	The chrome browser should display a design for XML file.
Actual Result	The chrome browser displayed a design for XML file.
Conclusion	Test successfully completed.

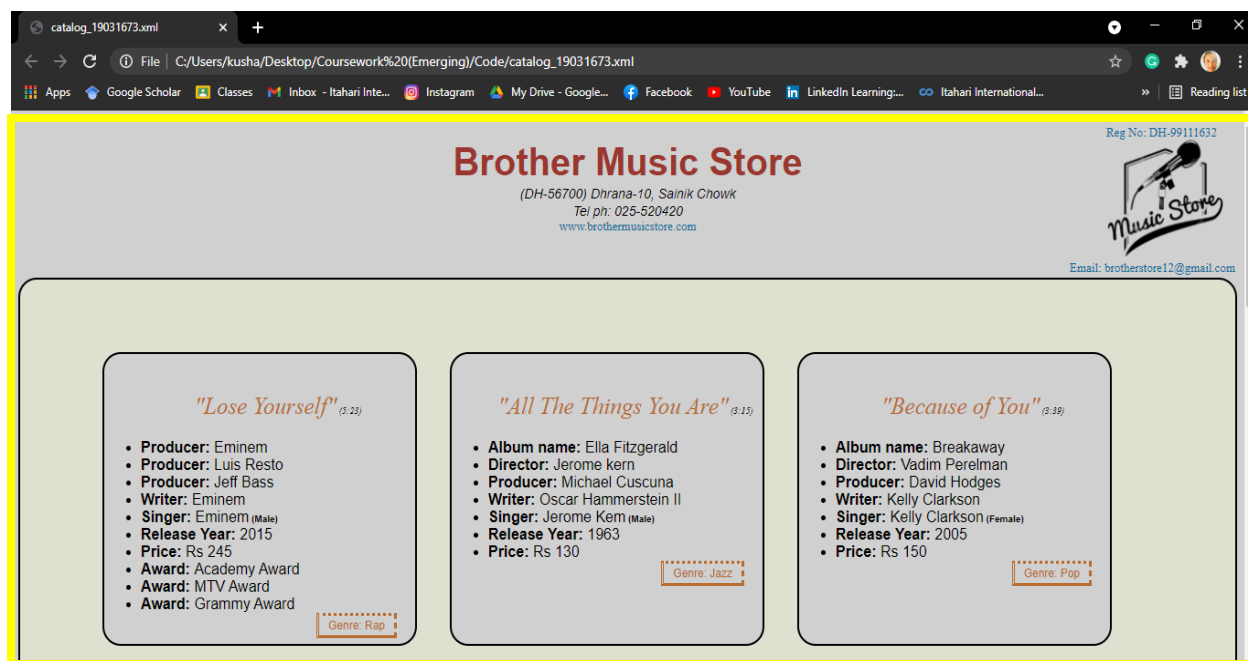




Figure 14 Screenshot of opening the XML file in Chrome browser

5.4. Patterns



Table 8 Test case for checking the pattern for telephone number element

Objective	To check the pattern made for telephone number.
Action	Entering the wrong value for element telephone number <u>Value:</u> 025-52042012
Expected Result	The xml validator should display the error message for telephone number element because the pattern created for telephone number is [0-9]{3}-[0-9]{6}.
Actual Result	The xml validator displayed the error message for telephone number element.
Conclusion	Test 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:

-  9: 50 cvc-pattern-valid: Value '025-52042012' is not facet-valid with respect to pattern '[0-9]{3}-[0-9]{6}' for type 'telephoneNumberType'.
-  9: 50 cvc-type.3.1.3: The value '025-52042012' of element 'telephoneNumber' is not valid.

XML document:

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <?xml-stylesheet type="text/css" href="catalog_19031673.css"?>
3 <!DOCTYPE musicStore SYSTEM "catalog_19031673.dtd">
4 <musicStore>
5 <storeDetails>
6 <registrationNumber>DH-99111632</registrationNumber>
7 <name>Brother Music Store</name>
8 <address postalCode="DH-56700">Dhrana-10, Sainik Chowk</address>
9 <telephoneNumber>025-52042012</telephoneNumber>
10 <websiteAddress>www.brothermusicstore.com</websiteAddress>



```

Figure 15 Screenshot of error message for telephone number element



Table 9 Test case for checking the pattern for website address element

Objective	To check the pattern made for website address.
Action	Entering the wrong value for element website address. <u>Value:</u> WWW.BROTHERMUSICSTORE.COM
Expected Result	The xml validator should display the error message for website address element because the pattern created for website address is [a-z]*.[a-z]*.[a-z]*.
Actual Result	The xml validator displayed the error message for website address element.
Conclusion	Test 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: 61 cvc-pattern-valid: Value 'WWW.BROTHERMUSICSTORE.COM' is not facet-valid with respect to pattern '[a-z]*.[a-z]*.[a-z]*' for type 'websiteAddressType'.
-  10: 61 cvc-type.3.1.3: The value 'WWW.BROTHERMUSICSTORE.COM' of element 'websiteAddress' is not valid.

XML document:

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <?xml-stylesheet type="text/css" href="catalog_19031673.css"?>
3 <!DOCTYPE musicStore SYSTEM "catalog_19031673.dtd">
4 <musicStore>
5 <storeDetails>
6 <registrationNumber>DH-99111632</registrationNumber>
7 <name>Brother Music Store</name>
8 <address postalCode="DH-56700">Dhrana-10, Sainik Chowk</address>
9 <telephoneNumber>025-520420</telephoneNumber>

```

```

    <websiteAddress>WWW.BROTHERMUSICSTORE.COM</websiteAddress>

```

```

10 

```

```



```

```

11 </logo></logo>

```



Figure 16 Screenshot of error message for website address element

5.5. Enumeration



Table 10 Test case for checking the enumeration for postal code attribute

Objective	To check the enumeration made for postal code attribute.
Action	Entering the wrong value for attribute postal code. <u>Value:</u> WWW.BROTHERMUSICSTORE.COM
Expected Result	The xml validator should display the error message for postal code attribute.
Actual Result	The xml validator displayed the error message for postal code attribute.
Conclusion	Test 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:

-  8:34 cvc-attribute.3: The value 'IT-56700' of attribute 'postalCode' on element 'address' is not valid with respect to its type, 'postalCodeType'.
-  8:34 cvc-enumeration-valid: Value 'IT-56700' is not facet-valid with respect to enumeration '[DH-56700]'. It must be a value from the enumeration.

XML document:

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <?xml-stylesheet type="text/css" href="catalog_19031673.css"?>
3 <!DOCTYPE musicStore SYSTEM "catalog_19031673.dtd">
4 <musicStore>
5 <storeDetails>
6 <registrationNumber>DH-99111632</registrationNumber>
7 <name>Brother Music Store</name>
  <address postalCode="IT-56700">
8 
  <Dhrana-10, Sainik Chowk</address>
9 <telephoneNumber>025-520420</telephoneNumber>



```

Figure 17 Screenshot of error message for postal code attribute



Table 11 Test case for checking the enumeration for genre attribute

Objective	To check the enumeration made for attribute attribute
Action	Entering the wrong value for attribute genre. <u>Value:</u> Hip hop song
Expected Result	The xml validator should display the error message for genre attribute.
Actual Result	The xml validator displayed the error message for genre attribute.
Conclusion	Test 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:

-  15:56 cvc-attribute.3: The value 'Hip Hop Song' of attribute 'genre' on element 'song' is not valid with respect to its type, 'genreType'.
-  15:56 'cvc-enumeration-valid: Value 'Hip Hop Song' is not facet-valid with respect to enumeration '[Rap, Jazz, Pop, Rock, Country Music, Childrens Music, Dubset]'. It must be a value from the enumeration.'

XML document:

```


1 <?xml version="1.0" encoding="UTF-8"?>
2 <?xml-stylesheet type="text/css" href="catalog_19031673.css"?>
3 <!DOCTYPE musicStore SYSTEM "catalog_19031673.dtd">
4 <musicStore>
5 <storeDetails>
6 <registrationNumber>DH-99111632</registrationNumber>
7 <name>Brother Music Store</name>
8 <address postalCode="DH-56700">Dhrana-10, Sainik Chowk</address>
9 <telephoneNumber>025-520420</telephoneNumber>
10 <websiteAddress>www.musicstore.com</websiteAddress>
11 <logo></logo>
12 <storeEmail>brotherstore12@gmail.com</storeEmail>
13 </storeDetails>
14 <songs>
    <song id="S-01" genre="Hip Hop Song" format="Online">
15 
    </song>
16 <songName length="5:23">"Lose Yourself"</songName>
  
```

Figure 18 Screenshot of error message for genre attribute

6. Development process

As the center focus point of the store is to share music data to various stages for the genuine usage of time and resources. The coursework was created with basic thinking on the continuous working design of the store in regards to the components to be utilized and the limitations to be made for the approval of the information. First and foremost, requirements and investigation were done to know the necessities for the clear design of XML. Various tools were utilized for the fulfillment of the coursework such as Sublime Text Editor, XML validator, and Draw.io for designing the tree diagram. In order to complete the coursework three tools were often practiced and the tools are described briefly.

Draw.io:

In this course work the first process was to design a tree diagram based on the given scenario requirement and for that purpose Draw.io tool was used which is the best decision for designing the tree diagram as it is particularly made for drawing charts. Draw.io is recognizable to everybody that thinks about designing models for their programs. It is a very good and friendly interface which is easy to use since the first time of use and it doesn't require installation as it is completely web-based. Maybe the best part of this is that it's completely free and it integrates with Google Drive so your work autosaves with every change.

Sublime Text Editor

Sublime Text is a sophisticated text editor for code, markup, and prose. You'll love the slick user interface, extraordinary features, and amazing performance. It is used in the coursework to code in the best way as it is auto-complete which helps a lot to reduce syntax error and the best thing is its easy file open feature along with the functionality to compare the code with split-screen which is very helpful to organize work. Its color-changing action helps a lot to understand code easily and get access to every part without any problem. Suitable for any coding language and it supports plain text which allows writing notes in Sublime too. Gives the best project management environment help to

organize our whole data and work area also it is free of cost so helps to save money, take less space helpful in storage saving.

XML Validator

XML validator is used to validate the files and it is an important process because it finds out the errors in XML. In this coursework, an XML validator was used to check the approval of the XML, DTD, and schema to affirm the appropriate validation. It makes possible to create XML reports with a negligible measure of mistakes. It is hard for anyone to perform a particularly specialized task faultlessly, which is the place where the XML validator tool becomes an integral factor. The XML validator in this task checks the XML syntax and also performs data validation. (Microsoft, 2021)

7. Critical Analysis and Conclusion

Critical Analysis

Learning is always a good experience for any students and through this given coursework many knowledges regarding different topics was learned. While doing this course work new terms were introduced and it made an enthusiasm of researching those terms. During the learning process of the module both physical and virtual classes were practiced as at the middle of the semester lockdown was announced due to increasing cases of Covid. Everything was going well while attending the practical classes but due to unexpected event the classes ended up with virtual classes which was a bit drawback for carrying out the coursework. Although the challenges, all the requirements of the coursework were completed with good research and proper assistance from the module leader.

For this coursework, different researches were done to fulfill the requirement of the scenario which made the mind even sharper and updated. The whole coursework was neither too tough nor too simple, simple in the sense as more terms were familiar before. The challenging factor of this task was DTD and schema because it was a new language to be practiced whereas markup language, data structure and CSS was very familiar.

During the construction of tree diagram main difficulties that were faced was identifying the elements and placing the attributes correctly to maintain the structure of the diagram. Writing XML and DTD was fully based on the document structure also known as tree diagram and it was not tough as it just required to follow the XML structure whereas Schema was quite difficult compared to both because it includes restriction of element for XML document. The difficulties were overcome with proper evaluation of the problem and required research.

As the part of research section, many journals, articles, websites, books were looked along with the consult from the module leader. Taking about the CSS part, as it was already familiar but some difficulties were faced during the design for making the XML document more presentable to the user.

The entire experience gained from this coursework was not only helpful for completing the task but it will also help in pursuing the upcoming task based on developing webpages in future for any firm or institute. With the successful completion of the coursework, it provided different idea regarding managing time, carrying out research, gaining experience and made the ability to undertake any further projects with sufficient knowledge.

Conclusion

To wrap up overall documentation of Music store, the system was developed successfully with the proper implementation of the resources that were required for the coursework. The whole coursework is based on the scenario required for the project and it was completed with proper research. The system was developed to store the data of the songs for the music store which includes different genre, name of the some and other details. The main objective of developing the project was to benefit the Music store with management facility and provided information of the songs to the customers.

To have a complete successful system many researches and analysis were performed before starting the project. The project started with the development of Tree diagram where different elements, attributes and modifying symbols were used, tree diagram plays a vital role as XML and DTD are written with its reference. After constructing the Tree diagram, XML, DTD, Schema and CSS are written using Sublime Text Editor 3.0. Schema was developed following Venetian Blind where it is done by grouping the element so that it can be reused again.

Furthermore, CSS in this coursework was used for designing the XML document which was the most fun part of the whole coursework and also it was already familiar. After completing all the coding part, the XML document was validated using XML validator to check the codes are correct or not. XML validator also includes validation of DTD and Schema as well. the code of the CSS was also validated using an online tool called CSS validator which checked the lines of code in the CSS whether it is correct or not.

The documentation was carried out for describing the overall project and its development process where it includes, a brief introduction to the coursework, its development process,

Testing and most importantly the critical analysis. Testing was done to test the restriction of the XML document with the help of DTD and Schema. The comparison of DTD and Schema was also carried out in the documentation part to make the clear view of the both terms. In a nutshell, it was a very pleasant experience completing the project by gaining enough potential to overcome different obstacles and be able to carry out challenging projects that will be assigned in the future projects.

8. Bibliography

Javatpoint. (2021) *Javatpoint* [Online]. Available from: <https://www.javatpoint.com/dtd-vs-xsd>.

Microsoft. (2021) *docs.microsoft.com* [Online]. Microsoft Available at: <https://docs.microsoft.com/en-us/visualstudio/xml-tools/xml-document-validation?view=vs-2019>.

Pedamkar, P. (2021) *Educba* [Online]. Available from: <https://www.educba.com/xml-with-css/>.

Singh, C. (2021) *BeginnersBook* [Online]. Available from: <https://beginnersbook.com/2018/11/xml-schema/>.

Tech Terms. (2021) *TechTerms* [Online]. Available from: <https://techterms.com/definition/dtd>.

W3. (2021) *W3C* [Online]. Available from: <https://www.w3.org/standards/xml/core>.

Webplatform. (2021) *Web platform* [Online]. Available from: https://webplatform.github.io/docs/tutorials/styling_xml_with_css/.