



slington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CS5004NI Emerging Programming Platforms and Technologies

Assessment Weightage & Type

30% Individual Coursework

Year and Semester

2020-21 Autumn

Student Name: Azan Ahmed Siddique

London Met ID: NP01CP4A190120

College ID: 19030779

Assignment Due Date: 7th May 2021

Assignment Submission Date: 7th May 2021

Word Count (Where Required): 1551

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 marks of zero will be awarded.

Table of Contents

Introduction	1
XML Content	3
Tree Diagram.....	3
XML Content.....	4
Schema Content	7
DTD Content	10
Testing	12
Difference between Schema and DTD	19
Development of the coursework.....	21
Critical Analysis	25
Conclusion	28
References.....	29

Table of Figures

Figure 1: Tree Diagram	3
Figure 2: Testing XML, XSD and DTD validation	12
Figure 3: Before adding CSS	13
Figure 4: Adding CSS.....	14
Figure 5: After adding CSS	14
Figure 6: Displaying the XML file through a browser	15
Figure 7: Before Hover	16
Figure 8: After Hover	17
Figure 9: Before removing optional element.....	18
Figure 10: After removing optional element.....	18
Figure 11: Validating after removing optional elements.....	18

Figure 12: Creating Tree diagram with draw.io	21
Figure 13: Writing code for the XML document	21
Figure 14: Creating the XSD file.....	22
Figure 15: Creating the DTD file.....	22
Figure 16: Validating the XML document	23
Figure 17: Creating the CSS file.....	23
Figure 18: XML document after adding CSS.....	24
Figure 19: Error during validation	25
Figure 20: Adding the schema attributes to DTD	25
Figure 21: Validation successful after adding schema attributes to DTD	26
Figure 22: XML content not displaying as expected.....	26
Figure 23: Setting display to flex	27
Figure 24: XML content after setting display to flex in CSS.....	27

Table of Tables

Table 1: Testing XML, XSD and DTD validation	12
Table 2: Display using CSS.....	13
Table 3: Displaying the XML document through a browser	15
Table 4: Displaying Hover Effect	16
Table 5: Testing Optional elements.....	17
Table 6: Difference between Schema and DTD	20

Introduction

This coursework required us to model a system for a music store using XML. A tree diagram has to be created after identifying the necessary data and attributes. The XML will then be created using XML and CSS will be used for formatting the layout of the webpage. Schema and DTD will also be created to define the structure of the XML document and validated.

XML

XML stands for **E**xtensible **M**arkup **L**anguage (Tutorials Point, 2021). It is a text-based markup language derived from Standard Generalized Markup Language (Tutorials Point, 2021).

XML tags identify the data and are used to store and organize the data, rather than specifying how to display it like HTML tags, which are used to display the data (Tutorials Point, 2021). XML is not going to replace HTML in the near future, but it introduces new possibilities by adopting many successful features of HTML (Tutorials Point, 2021).

Schema

An XML schema, commonly known as an XML Schema Definition (XSD), formally describes what a given XML document can contain, in the same way that a database schema describes the data that can be contained in a database (Liquid Technologies, 2021). The XML schema defines the shape, or structure, of an XML document, along with rules for data content and semantics such as what fields an element can contain, which sub elements it can contain and how many items can be present (Liquid Technologies, 2021). It can also describe the type and values that can be placed into each element or attribute (Liquid Technologies, 2021). The XML data constraints are called facets and include rules such as min and max length (Liquid Technologies, 2021).

DTD

The XML Document Type Declaration, commonly known as DTD, is a way to describe XML language precisely (Tutorials Point, 2021). DTDs check vocabulary and validity of the structure of XML documents against grammatical rules of appropriate XML language (Tutorials Point, 2021).

An XML DTD can be either specified inside the document, or it can be kept in a separate document and then linked separately (Tutorials Point, 2021).

CSS

CSS is the language for describing the presentation of Web pages, including colors, layout, and fonts (W3, 2016). It allows one to adapt the presentation to different types of devices, such as large screens, small screens, or printers (W3, 2016). CSS is independent of HTML and can be used with any XML-based markup language (W3, 2016). The separation of HTML from CSS makes it easier to maintain sites, share style sheets across pages, and tailor pages to different environments (W3, 2016). This is referred to as the *separation of structure (or: content) from presentation* (W3, 2016).

XML Content

Tree Diagram

An XML document has a self-descriptive structure (Javatpoint, 2018). It forms a tree structure which is referred as an XML tree. The tree structure makes easy to describe an XML document (Javatpoint, 2018). A tree structure contains root element (as parent), child element and so on (Javatpoint, 2018). It is very easy to traverse all succeeding branches and sub-branches and leaf nodes starting from the root (Javatpoint, 2018).

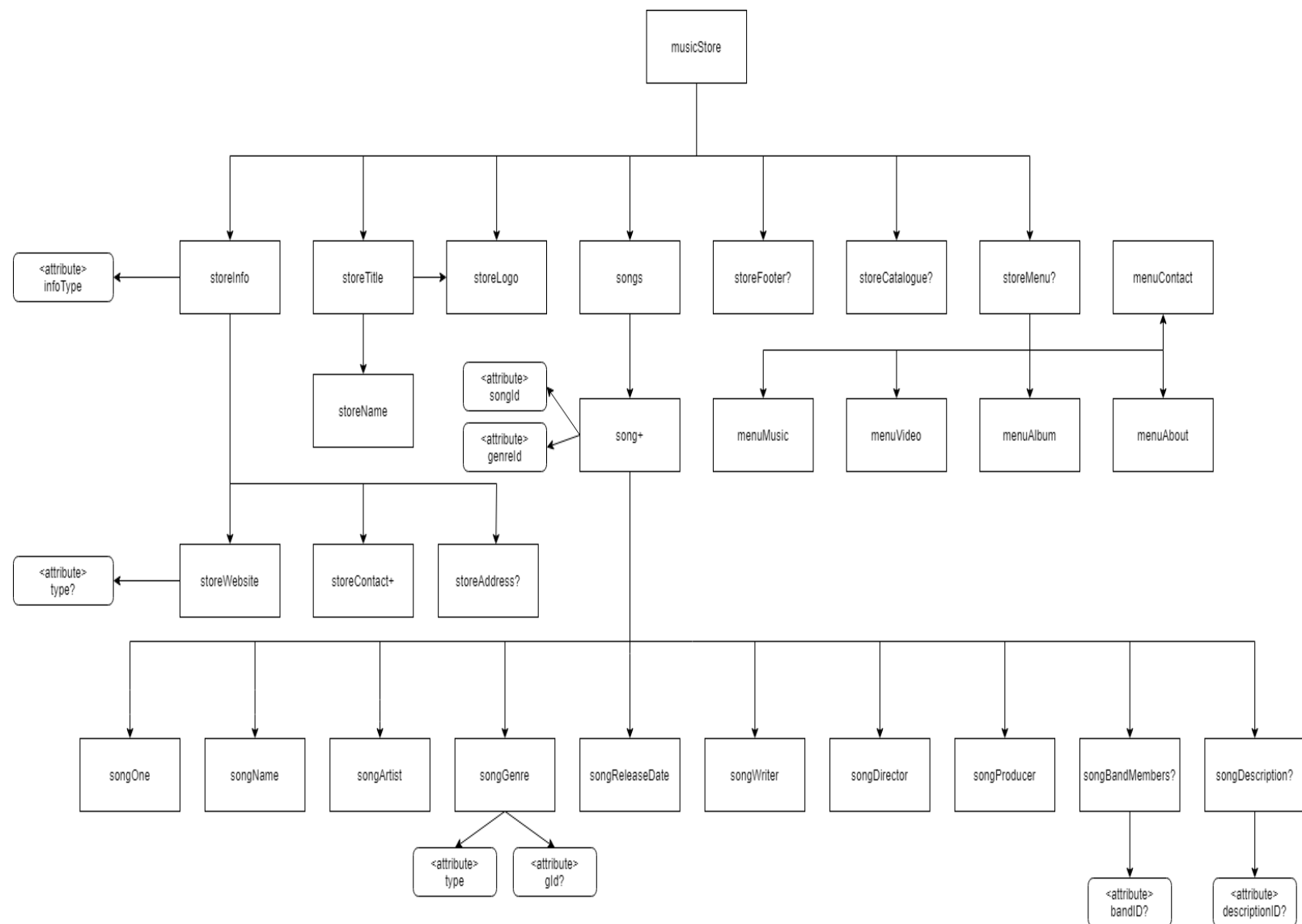


Figure 1: Tree Diagram

XML Content

```

<?xml version="1.0" encoding="UTF-8"?>
<!--Linking XML to CSS-->
<?xml-stylesheet type="text/css" href="catalog_19030779.css"?>
<!--Referencing external dtd-->
<!DOCTYPE musicStore SYSTEM "catalog_19030779.dtd">
<!--Decalring Root element musicStore and Referencing external schema-->
<musicStore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="catalog_19030779.xsd">
    <!--Describing the store title along with a logo for the store-->
    <storeTitle>
        <storeLogo></storeLogo>
        <storeName>Azan's Music Store</storeName>
    </storeTitle>
    <!--Describing information about the store like its address, contact,
etc-->
    <storeInfo infoType="About">
        <storeAddress>Kathmandu, Nepal</storeAddress>
        <storeContact>9803055257</storeContact>
        <storeWebsite
type="url">www.azansmusicstore.com</storeWebsite>
    </storeInfo>
    <!--Menu of the page-->
    <storeMenu>
        <menuMusic>Music</menuMusic>
        <menuVideo>Videos</menuVideo>
        <menuAlbum>Albums</menuAlbum>
        <menuAbout>About</menuAbout>
        <menuContact>Contact</menuContact>
    </storeMenu>
    <!--Creating a Music catalogue title to indicate the beginning of the
songs list-->

    <storeCatalogue>Music Catalogue</storeCatalogue>
    <!--Songs in the store-->
    <songs>
        <!--Describing the information about the song like its name,
artist name, genre, release date, writer, director, etc-->
        <song songId="S01" genreId="G01">
            <songOne></songOne>
            <songName>Thriller</songName>
            <songArtist>Artist: Michael Jackson</songArtist>
            <songGenre type="Popular Music" gId="G01">Genre:
Pop</songGenre>
            <songReleaseDate>Release Date: December 2,
1983</songReleaseDate>
            <songWriter>Writer(s): John Landis</songWriter>
            <songDirector>Director: John Landis</songDirector>
            <songProducer>Producer: George Folsey Jr</songProducer>

```

```

        <songBandMembers bandId="B01">Band Members: Michael
Boddicker, Bruce Cannon, Leon Chancler</songBandMembers>
        <songDescription descriptionId="D01">
            <![CDATA[
                Description: "Thriller" is a single by American
singer Michael Jackson. It was released as a single by Epic Records in
November 1983 in the UK, and on January 23, 1984 in the U.S.
            ]]>
        </songDescription>
    </song>
    <!--Describing the information about the song like its name,
artist name, genre, release date, writer, director, etc-->
    <song songId="S02" genreId="G02">
        <songTwo></songTwo>
        <songName>Absolutely Everybody</songName>
        <songArtist>Artist: Vanessa Amorosi</songArtist>
        <songGenre type="Jazz Music" gId="G02">Genre:
Jazz</songGenre>
        <songReleaseDate>Release Date: November 15,
1999</songReleaseDate>
        <songWriter>Writer(s): Mark Holden</songWriter>
        <songDirector>Director: Axel Breitung</songDirector>
        <songProducer>Producer: Axel Breitung </songProducer>
        <songBandMembers bandId="B02">Band Members: Anthony
John Hicks, James Roy Ingram</songBandMembers>
        <songDescription descriptionId="D02">
            <![CDATA[
                Description: "Absolutely Everybody" is a song
by Vanessa Amorosi, released as the second single from her debut album, The
Power, on 15 November 1999 by Transistor Music Australia.
            ]]>
        </songDescription>
    </song>
    <!--Describing the information about the song like its name,
artist name, genre, release date, writer, director, etc-->
    <song songId="S03" genreId="G03">
        <songThree></songThree>
        <songName>Perfect</songName>
        <songArtist>Artist: Ed Sheeran</songArtist>
        <songGenre type="Country Music," gId="G03">Genre:
Country</songGenre>
        <songReleaseDate>Release Date: September 26,
2017</songReleaseDate>
        <songWriter>Writer(s): Ed Sheeran</songWriter>
        <songDirector>Director: Will Hicks</songDirector>
        <songProducer>Producer: Will Hicks</songProducer>
        <songBandMembers bandId="B03">Band Members: Roger
Meadows Taylor, Ed Sheeran</songBandMembers>
        <songDescription descriptionId="D03">
            <![CDATA[

```



```

        Description: Perfect" is a song by English
singer-songwriter Ed Sheeran from his third album. After the album's release,
it charted at number four on the UK Singles Chart.
    ]]>
</songDescription>
</song>
<!--Describing the information about the song like its name,
artist name, genre, release date, writer, director, etc-->
<song songId="S04" genreId="G04">
    <songFour></songFour>
    <songName>Bohemian Rhapsody</songName>
    <songArtist>Artist: Queen</songArtist>
    <songGenre type="Rock Music" gId="G04">Genre:
Progressive rock</songGenre>
    <songReleaseDate>Release Date: October 31,
1975</songReleaseDate>
    <songWriter>Writer(s): Freddie Mercury</songWriter>
    <songDirector>Director: Roy Thomas Baker</songDirector>
    <songProducer>Producer: Roy Thomas Baker</songProducer>
    <songBandMembers bandId="B04">Band Members: Brian May,
Roger Taylor, John Deacon</songBandMembers>
    <songDescription descriptionId="D04">
        <![CDATA[
            Description: "Bohemian Rhapsody" is a song by
the British rock band Queen. It was written by Freddie Mercury. for the
band's 1975 album A Night at the Opera.
        ]]>
    </songDescription>
</song>
<!--Describing the information about the song like its name,
artist name, genre, release date, writer, director, etc-->
<song songId="S05" genreId="G05">
    <songFive></songFive>
    <songName>Here Without You</songName>
    <songArtist>Artist: 3 Doors Down</songArtist>
    <songGenre type="Classical Music" gId="G05">Genre:
Classical</songGenre>
    <songReleaseDate>Release Date: August 11,
2003</songReleaseDate>
    <songWriter>Writer(s): Brad Arnold, Todd
Harrell</songWriter>
    <songDirector>Director: Chris Henderson</songDirector>
    <songProducer>Producer: Rick Parashar</songProducer>
    <songBandMembers bandId="B05">Band Members: Brad
Arnold, Chris Henderson, Greg Upchurch</songBandMembers>
    <songDescription descriptionId="D05">
        <![CDATA[
            Description: "Here Without You" is a song by
American rock band 3 Doors Down. It was released on August 11, 2003, from
the second studio album Away from the Sun (2002).
        ]]>
    </songDescription>

```

```

        </song>
        <!--Describing the information about the song like its name,
artist name, genre, release date, writer, director, etc-->
        <song songId="S06" genreId="G06">
            <songSix></songSix>
            <songName>Hallelujah</songName>
            <songArtist>Artist: Leonard Cohen</songArtist>
            <songGenre type="Folk Music" gId="G06">Genre: Folk
rock</songGenre>
            <songReleaseDate>Release Date: December 14,
1984</songReleaseDate>
            <songWriter>Writer(s): Leonard Cohen</songWriter>
            <songDirector>Director: Spike Jonze</songDirector>
            <songProducer>Producer: John Lissauer</songProducer>
            <songBandMembers bandId="B06">Band Members: Roscoe
Beck, Neil Larsen, Leonard Cohen</songBandMembers>
            <songDescription descriptionId="D06">
                <![CDATA[
                    Description: "Hallelujah" is a song written by
Canadian singer Leonard Cohen. Achieving little initial success, the song
found acclaim through a recording by John Cale.
                ]]>
            </songDescription>
        </song>
    </songs>
    <!--Describing the footer of the page!-->
    <storeFooter>Copyright &#xA9; Azan Ahmed Siddique 2021</storeFooter>
</musicStore>

```

Schema Content

```

<!--schema start-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <!--musicStore is Root Element-->
    <xs:element name="musicStore">
        <xs:complexType>
            <xs:sequence>
                <!--Element storeTitle contains the logo and Name of the store which
are of type string-->
                <xs:element name="storeTitle">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="storeLogo" type="xs:string"/>
                            <xs:element name="storeName" type="xs:string"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <!--Element storeTitle contains the informations about the store
which are of types string and integers-->
                <xs:element name="storeInfo">

```

```

        <xs:complexType mixed="true">
            <xs:sequence>
                <xs:element name="storeAddress" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
                <xs:element name="storeContact" type="xs:integer"/>
                <xs:element name="storeWebsite">
                    <xs:complexType>
                        <xs:simpleContent>
                            <xs:extension base="xs:anyURI">
                                <!--describing the attribute named type-->
                                <xs:attribute name="type" type="xs:string"/>
                            </xs:extension>
                        </xs:simpleContent>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
            <!--describing the attribute named infoType-->
            <xs:attribute name="infoType" type="xs:string"/>
        </xs:complexType>
    </xs:element>
    <!--Element storeMenu contains the menu list which are of type
string-->
    <xs:element name="storeMenu" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="menuMusic" type="xs:string"/>
                <xs:element name="menuVideo" type="xs:string"/>
                <xs:element name="menuAlbum" type="xs:string"/>
                <xs:element name="menuAbout" type="xs:string"/>
                <xs:element name="menuContact" type="xs:string"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <!--Element storCatalogue describes the music catalogue which is of
type string-->
    <xs:element name="storeCatalogue" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
    <!--Element songs contains lists of songs -->
    <xs:element name="songs">
        <xs:complexType>
            <xs:sequence>
                <!--Element song contains information about the song which are
of type string-->
                <xs:element name="song" minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType mixed="true">
                        <xs:sequence>
                            <xs:element name="songOne" type="xs:string"
minOccurs="0"/>
                            <xs:element name="songTwo" type="xs:string"
minOccurs="0"/>
                            <xs:element name="songThree" type="xs:string"
minOccurs="0"/>

```

```

minOccurs="0"/>
minOccurs="0"/>
minOccurs="0"/>
<xs:element name="songFour" type="xs:string"
<xs:element name="songFive" type="xs:string"
<xs:element name="songSix" type="xs:string"
<xs:element name="songName" type="xs:string"/>
<xs:element name="songArtist" type="xs:string"/>
<xs:element name="songGenre">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <!--describing the attributes named type and gId-
->
          <xs:attribute name="type" type="xs:string"
          <xs:attribute name="gId" type="xs:string"
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="songReleaseDate" type="xs:string"/>
  <xs:element name="songWriter" type="xs:string"/>
  <xs:element name="songDirector" type="xs:string"/>
  <xs:element name="songProducer" type="xs:string"/>
  <xs:element name="songBandMembers" minOccurs="0">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <!--describing the attribute named bandId-->
          <xs:attribute name="bandId" type="xs:string"
use="optional"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="songDescription" minOccurs="0">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <!--describing the attribute named descriptionId-
->
            <xs:attribute name="descriptionId"
type="xs:string" use="optional"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
</xs:sequence>
<!--describing the attributes named songId and genreId-->

```

```

        <xs:attribute name="songId" type="xs:string"
use="required"/>
        <xs:attribute name="genreId" type="xs:string"
use="required"/>
    </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<!--Element storeFooter contains the footer of the page-->
<xs:element name="storeFooter" minOccurs="0"/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>

```

DTD Content

```

<!--musicStore is the root element, it stores 6 child elements data-->
<!ELEMENT musicStore (storeTitle, storeInfo, storeMenu?, storeCatalogue?,
songs, storeFooter?)>
<!--Element storeTitle stores two child elements which are of type 'EMPTY'
and '#PCDATA' respectively-->
<!ELEMENT storeTitle (storeLogo, storeName)>
<!ELEMENT storeLogo EMPTY>
<!ELEMENT storeName (#PCDATA)>
<!--element storeTitle describes the store information, it stores three child
elements data which are of type '#PCDATA'-->
<!ELEMENT storeInfo (storeAddress?,storeContact+,storeWebsite)>
<!ELEMENT storeAddress (#PCDATA)>
<!ELEMENT storeContact (#PCDATA)>
<!ELEMENT storeWebsite (#PCDATA)>
<!--element storeMenu describes the menu, it stores six child elements data
which are of type '#PCDATA'-->
<!ELEMENT storeMenu (menuMusic, menuVideo, menuAlbum, menuAbout,
menuContact)>
<!ELEMENT menuMusic (#PCDATA)>
<!ELEMENT menuVideo (#PCDATA)>
<!ELEMENT menuAlbum (#PCDATA)>
<!ELEMENT menuAbout (#PCDATA)>
<!ELEMENT menuContact (#PCDATA)>
<!--element storeCatalogue describes the music catalogue, it stores data of
type '#PCDATA'-->
<!ELEMENT storeCatalogue (#PCDATA)>
<!--element storeMenu describes the songs, it stores one child element data--
>
<!ELEMENT songs (song+)>
<!--element song describes the information about the song, it contains 10
child elements which are of type 'EMPTY', '#PCDATA' and 'ANY'-->

```

```

<!ELEMENT song ((songOne|songTwo|songThree|songFour|songFive|songSix),
songName, songArtist, songGenre, songReleaseDate, songWriter, songDirector,
songProducer, songBandMembers?, songDescription?)>
<!ELEMENT songOne EMPTY>
<!ELEMENT songTwo EMPTY>
<!ELEMENT songThree EMPTY>
<!ELEMENT songFour EMPTY>
<!ELEMENT songFive EMPTY>
<!ELEMENT songSix EMPTY>
<!ELEMENT songName (#PCDATA)>
<!ELEMENT songArtist (#PCDATA)>
<!ELEMENT songGenre (#PCDATA)>
<!ELEMENT songReleaseDate (#PCDATA)>
<!ELEMENT songWriter (#PCDATA)>
<!ELEMENT songDirector (#PCDATA)>
<!ELEMENT songProducer (#PCDATA)>
<!ELEMENT songBandMembers (#PCDATA)>
<!ELEMENT songDescription ANY>
<!ELEMENT storeFooter (#PCDATA)>

<!--element musicStore contains schema attributes, the attributes are of type
'CDATA' and the values are '#REQUIRED'-->
<!ATTLIST musicStore xmlns:xsi CDATA #REQUIRED xsi:noNamespaceSchemaLocation
CDATA '#REQUIRED'>
<!--element storeInfo contains attribute infoType, the attribute is of type
'CDATA' and the value is '#FIXED'-->
<!ATTLIST storeInfo infoType CDATA #FIXED "About">
<!--element storeWebsite contains attribute type, the attribute is of type
'CDATA' and the value is '#IMPLIED'-->
<!ATTLIST storeWebsite type CDATA #IMPLIED>
<!--element song contains attribute songId and genreId, the attributes are of
type 'CDATA' and 'ID' and the values are '#REQUIRED'-->
<!ATTLIST song songId CDATA #REQUIRED genreId ID #REQUIRED>
<!--element songGenre contains attribute type and gId, the attributes are of
type 'CDATA' IDREF' and the values are 'REQUIRED' and '#IMPLIED'-->
<!ATTLIST songGenre type CDATA #REQUIRED gId IDREF #IMPLIED>
<!--element songBandMembers contains attribute bandID, the attribute is of
type 'ID' and the value is '#IMPLIED'-->
<!ATTLIST songBandMembers bandId ID #IMPLIED>
<!--element songDescription contains attribute descriptionId, the attribute
is of type 'CDATA' and the value is '#IMPLIED'-->
<!ATTLIST songDescription descriptionId CDATA #IMPLIED>

```

Testing

Test 1	Testing XML, XSD and DTD validation
Action	Adding XML, XSD and DTD file to xmlvalidation.com
Expected Output	No errors should be found
Actual Output	No errors were found
Result	Successful

Table 1: Testing XML, XSD and DTD validation

Screenshot

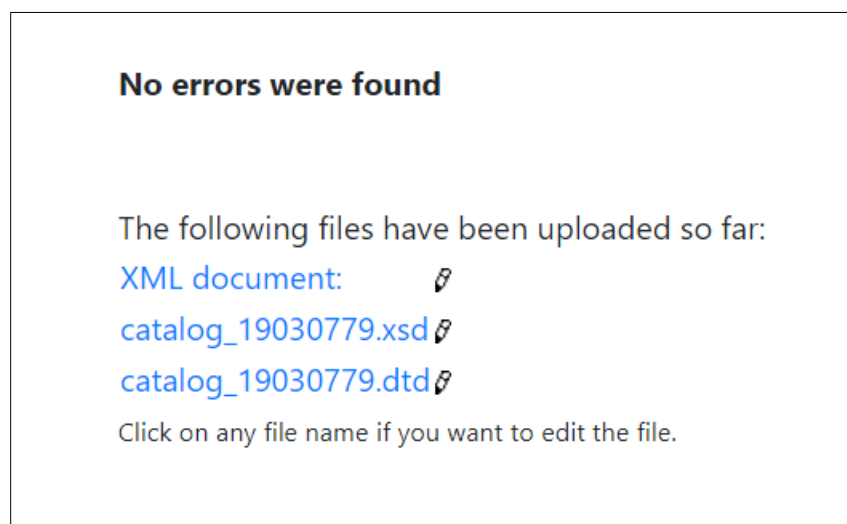


Figure 2: Testing XML, XSD and DTD validation

Test 2	Display using CSS
Action	Linking CSS to XML
Expected Output	XML document should be formatted
Actual Output	XML document should be formatted
Result	Successful

Table 2: Display using CSS

Screenshot

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

```

<!-- Referencing external dtd -->
<!-- Declaring Root element musicStore and Referencing external schema -->
<musicStore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="catalog_19030779.xsd">
  <!-- Describing the store title along with a logo for the store -->
  <storeTitle>
    <storeLogo/>
    <storeName>Azan's Music Store</storeName>
  </storeTitle>
  <!-- Describing information about the store like its address, contact, etc -->
  <storeInfo infoType="About">
    <storeAddress>Kathmandu, Nepal</storeAddress>
    <storeContact>9803055257</storeContact>
    <storeWebsite type="url">www.azansmusicstore.com</storeWebsite>
  </storeInfo>
  <!-- Menu of the page -->
  <storeMenu>
    <menuMusic>Music</menuMusic>
    <menuVideo>Videos</menuVideo>
    <menuAlbum>Albums</menuAlbum>
    <menuAbout>About</menuAbout>
    <menuContact>Contact</menuContact>
  </storeMenu>
  <!-- Creating a Music catalogue title to indicate the beginning of the songs list -->
  <storeCatalogue>Music Catalogue</storeCatalogue>
  <!-- Songs in the store -->
  <songs>
    <!-- Describing the information about the song like its name, artist name, genre, release date, writer, director, etc -->
    <song songId="S01" genreId="G01">
      <songOne/>
      <songName>Thriller</songName>
      <songArtist>Artist: Michael Jackson</songArtist>
      <songGenre type="Popular Music" gId="G01">Genre: Pop</songGenre>
      <songReleaseDate>Release Date: December 2, 1983</songReleaseDate>
      <songWriter>Writer(s): John Landis</songWriter>
      <songDirector>Director: John Landis</songDirector>
      <songProducer>Producer: George Folsey Jr</songProducer>
    </song>
  </songs>
</musicStore>

```

Figure 3: Before adding CSS


```

E:\College Work\Coursework Year 2 Sem 2\Emerging\Development\catalog_19030779.xml - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

catalog_19030779.xml x catalog_19030779.xsd x catalog_19030779.dtd x catalog_19030779.css x
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!--Linking XML to CSS-->
3 <?xml-stylesheet type="text/css" href="catalog_19030779.css"?>
4 <!--Referencing external dtd-->
5 <!DOCTYPE musicStore SYSTEM "catalog_19030779.dtd">
6 <!--Declaring Root element musicStore and Referencing external schema-->

```

Figure 4: Adding CSS

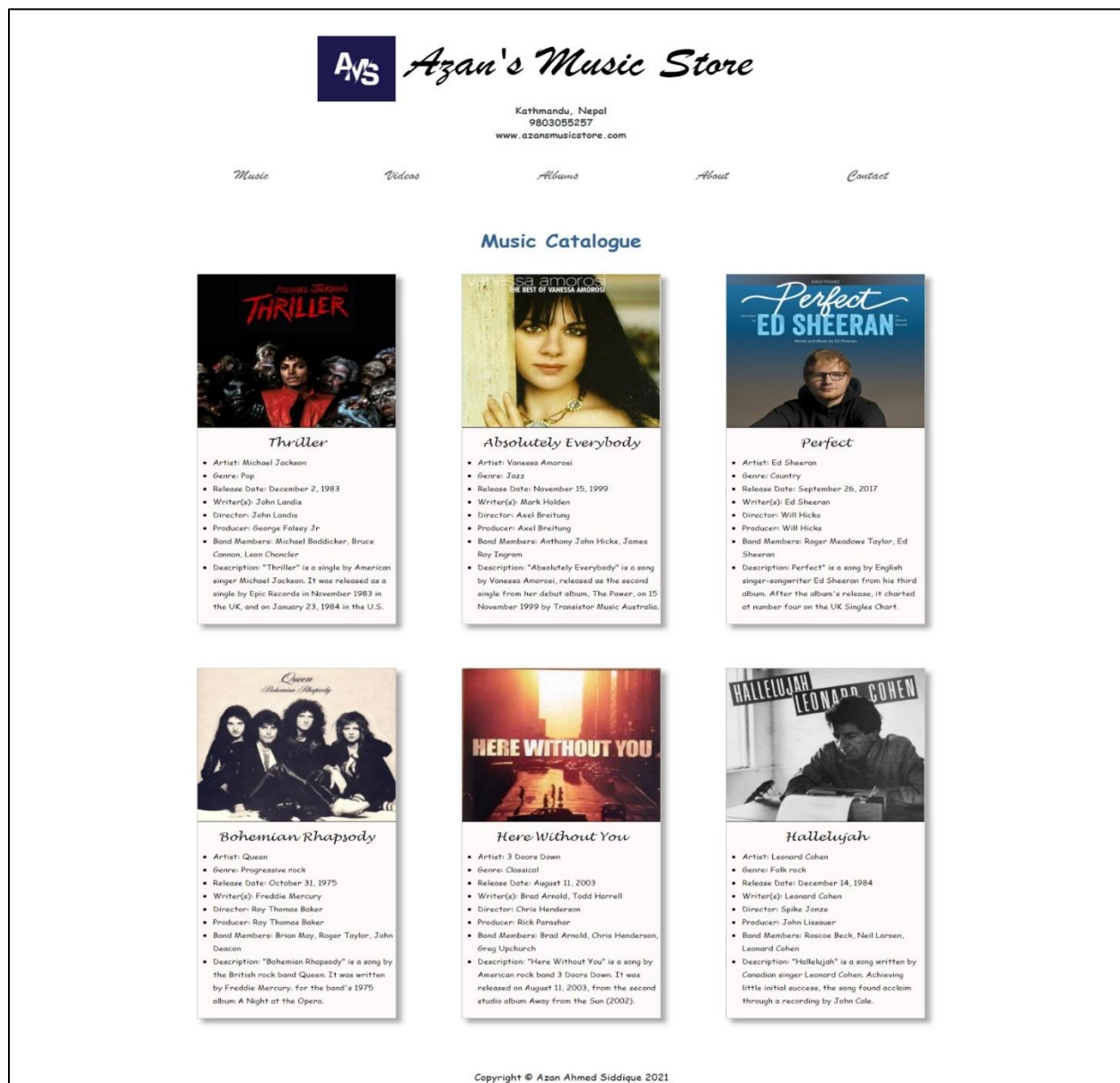


Figure 5: After adding CSS

Test 3	Displaying the XML document through a browser
Action	Open the XML document using a web browser
Expected Output	The XML webpage should be displayed
Actual Output	The XML webpage is displayed
Result	Successful

Table 3: Displaying the XML document through a browser

Screenshot

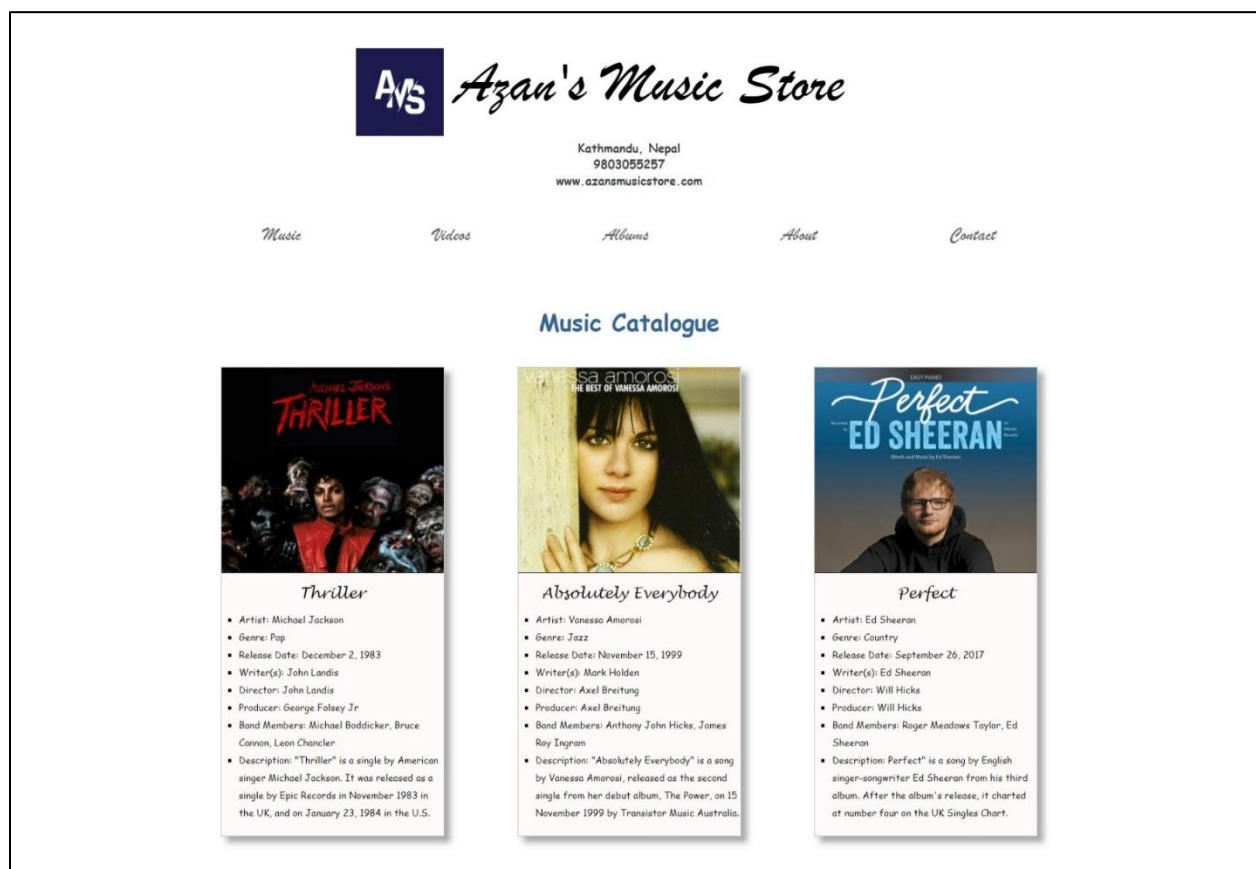


Figure 6: Displaying the XML file through a browser

Test 4	Displaying Hover Effect
Action	Hover the mouse cursor over the song box
Expected Output	The box should zoom in
Actual Output	The box zooms in
Result	Successful

Table 4: Displaying Hover Effect

Screenshot



Figure 7: Before Hover



Figure 8: After Hover

Test 5	Testing Optional elements
Action	Remove optional element from XML and validate it in xmlvalidation.com
Expected Output	No errors should be found
Actual Output	No errors were found
Result	Successful

Table 5: Testing Optional elements

Screenshot

```

<song songId="S01" genreId="G01">
  <songOne></songOne>
  <songName>Thriller</songName>
  <songArtist>Artist: Michael Jackson</songArtist>
  <songGenre type="Popular Music" gId="G01">Genre: Pop</songGenre>
  <songReleaseDate>Release Date: December 2, 1983</songReleaseDate>
  <songWriter>Writer(s): John Landis</songWriter>
  <songDirector>Director: John Landis</songDirector>
  <songProducer>Producer: George Folsey Jr</songProducer>
  <songBandMembers bandId="B01">Band Members: Michael Boddicker, Bruce (
  <songDescription descriptionId="D01">
    <![CDATA[
      Description: "Thriller" is a single by American singer Michael Ja
      1984 in the U.S.
    ]]>
  </songDescription>
</song>

```

Figure 9: Before removing optional element

```


<song songId="S01" genreId="G01">
  <songOne></songOne>
  <songName>Thriller</songName>
  <songArtist>Artist: Michael Jackson</songArtist>
  <songGenre type="Popular Music" gId="G01">Genre: Pop</songGenre>
  <songReleaseDate>Release Date: December 2, 1983</songReleaseDate>
  <songWriter>Writer(s): John Landis</songWriter>
  <songDirector>Director: John Landis</songDirector>
  <songProducer>Producer: George Folsey Jr</songProducer>
</song>


```


Figure 10: After removing optional element

No errors were found

The following files have been uploaded so far:

XML document: 

catalog_19030779.xsd 

catalog_19030779.dtd 

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

Figure 11: Validating after removing optional elements

Difference between Schema and DTD

Document Type Definitions (DTD)

DTD stands for Document Type Definition and it is a document which defines the structure of an XML document (GeeksforGeeks, 2020). It is used to describe the attributes of XML language precisely (GeeksforGeeks, 2020). It can be classified into two types namely internal DTD and external DTD (GeeksforGeeks, 2020). It can be specified inside a document or outside a document (GeeksforGeeks, 2020). DTD mainly checks the grammar and validity of a XML document. It checks that a XML document has a valid structure or not (GeeksforGeeks, 2020).

XML Schema Definition (XSD)

XSD stands for XML Schema Definition and it is a way to describe the structure of a XML document (GeeksforGeeks, 2020). It defines the rules for all the attributes and elements in a XML document (GeeksforGeeks, 2020). It can also be used to generate the XML documents. It also checks the vocabulary of the document (GeeksforGeeks, 2020). It doesn't require processing by a parser. XSD checks for the correctness of the structure of the XML file. XSD was first published in 2001 and after that it was published in 2004 (GeeksforGeeks, 2020).

Difference between XML Schema and DTD

The critical difference between DTDs and XML Schema is that XML Schema utilize an XML-based syntax, whereas DTDs have a unique syntax held over from SGML DTDs (Informit, 2001). Although DTDs are often criticized because of this need to learn a new syntax, the syntax itself is quite terse (Informit, 2001). The opposite is true for XML

Schema, which are verbose, but also make use of tags and XML so that authors of XML should find the syntax of XML Schema less intimidating (Informit, 2001).

In summary, some of the major differences between XML Schema and DTD are:

XSD (Schema)	DTD
XSD stands for XML Schema Definition (Javatpoint, 2018).	DTD stands for Document Type Definition (Javatpoint, 2018).
XSD supports namespace (Javatpoint, 2018).	DTD doesn't support namespace (Javatpoint, 2018).
XSDs are written in XML (Javatpoint, 2018).	DTDs are derived from SGML syntax (Javatpoint, 2018).
XSD supports datatypes for elements and attributes (Javatpoint, 2018).	DTD doesn't support datatypes.
XSD provides more control on XML structure (Javatpoint, 2018).	DTD provides less control on XML structure (Javatpoint, 2018).
XSD defines order for child elements (Javatpoint, 2018).	DTD doesn't define order for child elements (Javatpoint, 2018).
XSD is simple to learn as you don't have to learn a new language (Javatpoint, 2018).	DTD is not simple to learn (Javatpoint, 2018).
XSD is extensible (Javatpoint, 2018).	DTD is not extensible (Javatpoint, 2018).
It gives us more control on structure of XML document (GeeksforGeeks, 2020).	It doesn't give us much control on structure of XML document (GeeksforGeeks, 2020).
XSD is strongly typed (Joan, 2017).	DTD is not strongly typed (Joan, 2017).
It supports numeric, Boolean and String data types (Career Ride, 2016).	It supports two types of data. PCDATA and CDATA (Career Ride, 2016).

Table 6: Difference between Schema and DTD

Development of the coursework

In order to develop this coursework, an XML, XSD, DTD and CSS file had to be created. A tree diagram has also been created to show the root elements and its succeeding branches.

The tree diagram has been created using draw.io (a very useful tool for designing diagram using simple drag and drop method).

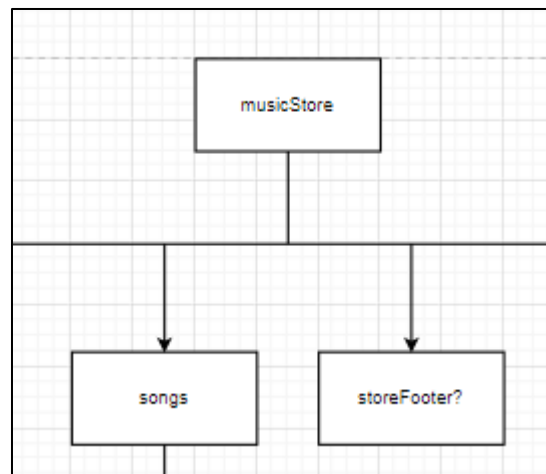


Figure 12: Creating Tree diagram with draw.io

The XML, XSD, DTD and CSS files were created using a text editor called Sublime Text (a sophisticated text editor for code, mark-up and prose). First the XML document was created, all of the required elements and attributes were created and tested through a browser for errors.

```

<?xml version="1.0" encoding="UTF-8"?>
<!--Linking XML to CSS-->
<?xml-stylesheet type="text/css" href="catalog_19030779.css"?>
<!--Referencing external dtd-->
<!DOCTYPE musicStore SYSTEM "catalog_19030779.dtd">
<!--Declaring Root element musicStore and Referencing external schema-->
<musicStore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <!--Describing the store title along with a logo for the store-->
  <storeTitle>
    <storeLogo></storeLogo>
    <storeName>Azan's Music Store</storeName>
  </storeTitle>
  <!--Describing information about the store like its address,
  <storeInfo infoType="About">
    <storeAddress>Kathmandu, Nepal</storeAddress>
    <storeContact>9803055257</storeContact>
    <storeWebsite type="url">www.azansmusicstore.com</storeWebsite>
  </storeInfo>
  <!--Describing the store's location-->
  <storeLocation>
    <storeAddress>Kathmandu, Nepal</storeAddress>
    <storeContact>9803055257</storeContact>
    <storeWebsite type="url">www.azansmusicstore.com</storeWebsite>
  </storeLocation>
  <!--Describing the store's products-->
  <storeProducts>
    <storeProduct>
      <storeName>Azan's Music Store</storeName>
      <storeAddress>Kathmandu, Nepal</storeAddress>
      <storeContact>9803055257</storeContact>
      <storeWebsite type="url">www.azansmusicstore.com</storeWebsite>
    </storeProduct>
  </storeProducts>
  <!--Describing the store's services-->
  <storeServices>
    <storeService>
      <storeName>Azan's Music Store</storeName>
      <storeAddress>Kathmandu, Nepal</storeAddress>
      <storeContact>9803055257</storeContact>
      <storeWebsite type="url">www.azansmusicstore.com</storeWebsite>
    </storeService>
  </storeServices>
  <!--Describing the store's contact information-->
  <storeContactInfo>
    <storeName>Azan's Music Store</storeName>
    <storeAddress>Kathmandu, Nepal</storeAddress>
    <storeContact>9803055257</storeContact>
    <storeWebsite type="url">www.azansmusicstore.com</storeWebsite>
  </storeContactInfo>
  <!--Describing the store's footer-->
  <storeFooter>
    <storeName>Azan's Music Store</storeName>
    <storeAddress>Kathmandu, Nepal</storeAddress>
    <storeContact>9803055257</storeContact>
    <storeWebsite type="url">www.azansmusicstore.com</storeWebsite>
  </storeFooter>
  </musicStore>
  
```

Figure 13: Writing code for the XML document

After that, the XSD and DTD files were created and structure of the elements and attributes in the XML document.

```
<!--schema start-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <!--musicStore is Root Element-->
  <xs:element name="musicStore">
    <xs:complexType>
      <xs:sequence>
        <!--Element storeTitle contains the logo and Name of the store-->
        <xs:element name="storeTitle">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="storeLogo" type="xs:string"/>
              <xs:element name="storeName" type="xs:string"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Figure 14: Creating the XSD file

```
<!--musicStore is the root element, it stores 6 child elements data-->
<!ELEMENT musicStore (storeTitle, storeInfo, storeMenu?, storeCatalogue?)>
<!--Element storeTitle stores two child elements which are of type 'EMPTY'-->
<!ELEMENT storeTitle (storeLogo, storeName)>
<!ELEMENT storeLogo EMPTY>
<!ELEMENT storeName (#PCDATA)>
<!--element storeInfo describes the store information, it stores three child elements-->
<!ELEMENT storeInfo (storeAddress?,storeContact+,storeWebsite)>
<!ELEMENT storeAddress (#PCDATA)>
<!ELEMENT storeContact (#PCDATA)>
<!ELEMENT storeWebsite (#PCDATA)>
<!--element storeMenu describes the menu, it stores six child elements-->
<!ELEMENT storeMenu (menuMusic, menuVideo, menuAlbum, menuAbout, menuContact, menuFeedback)>
<!ELEMENT menuMusic (#PCDATA)>
<!ELEMENT menuVideo (#PCDATA)>
<!ELEMENT menuAlbum (#PCDATA)>
<!ELEMENT menuAbout (#PCDATA)>
<!ELEMENT menuContact (#PCDATA)>
<!ELEMENT menuFeedback (#PCDATA)>
```

Figure 15: Creating the DTD file

Once the XML, XSD and DTD files were created, the XML document is validated against the XSD Schema and DTD using an online validator called xmlvalidation.com. No errors were found during the validation process.

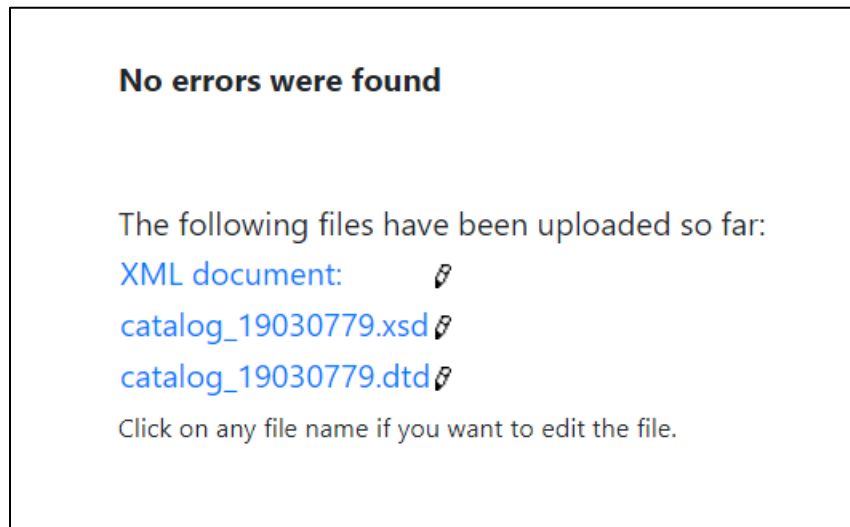


Figure 16: Validating the XML document

The XML, XSD and DTD files have been successfully created and tested. The next step is to format the layout of the webpage, make it more appealing and present the document to the user in a web browser. This can be done using CSS (Cascading Style Sheet). Using CSS, the contents of the XML document can be displayed in a clear manner.

```
/* Set Background color of the page */
musicStore{
    background-color: #FFFFFF; /* Set background color to white */
}
/* Styling the logo */
storeLogo{
    background-image: url("ams.jpg"); /* Set store logo */
    background-repeat: no-repeat; /* Set background image to no repeat */
    background-size: cover; /* Set background image size to cover */
    width: 150px;
    height: 150px;
    margin-left: -100px;
    margin-right: 20px;
    cursor: pointer; /* Set cursor type to pointer */
}
/* Styling the storeTitle element */
storeTitle{
    display: flex; /* Set display to flex */
    text-align: center;
    justify-content: center;
    width: 800px;
```

Figure 17: Creating the CSS file

Finally, the CSS file was created and now it's time to open the XML document through a web browser (Microsoft Edge, Google Chrome, etc.) to check if the data in the XML document is being rendered properly.

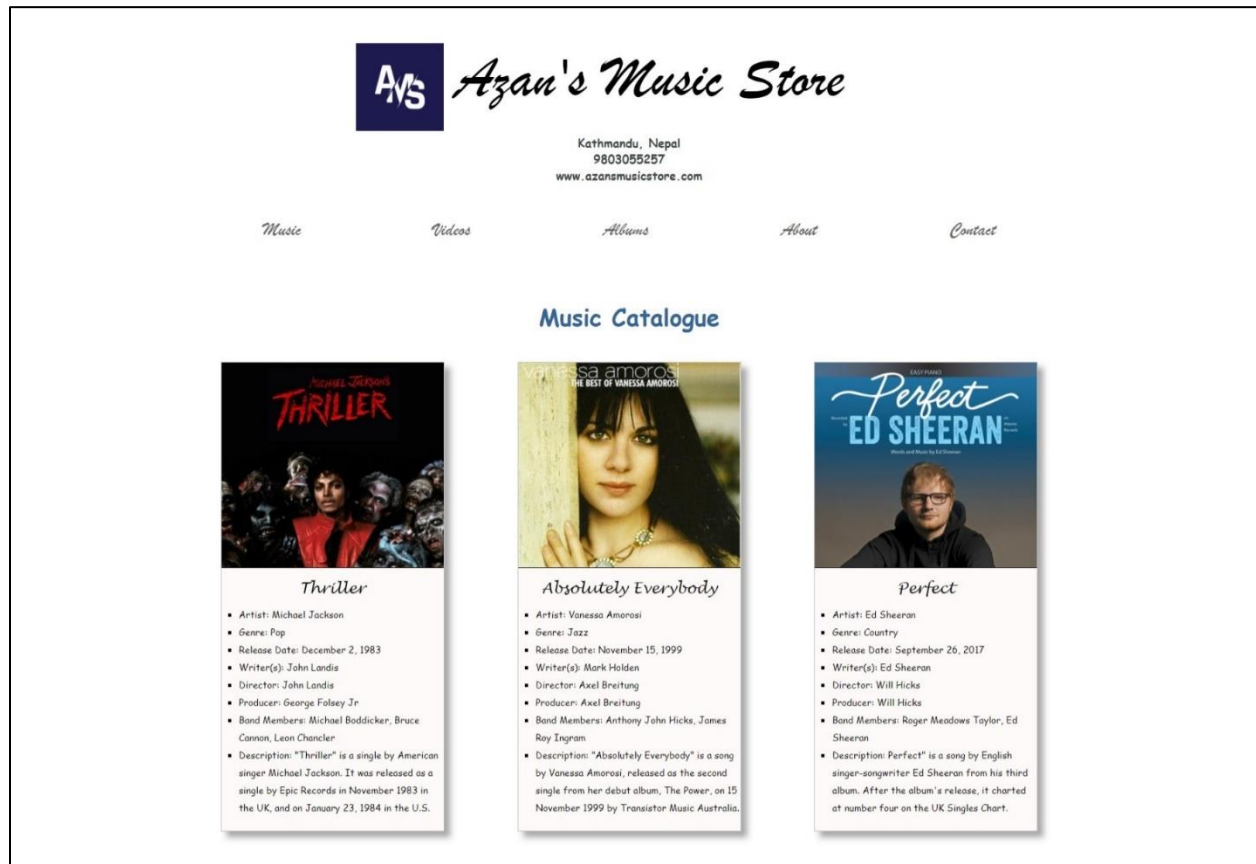


Figure 18: XML document after adding CSS

The development section of the coursework has therefore been completed and all the requirements have been fulfilled.

Critical Analysis

While doing this coursework, I faced many challenges. I encountered various different types of error while creating the XML, XSD, DTD and CSS files.

The first major problem that I encountered was while validation the XML against both XSD and DTD files. I wanted to validate them together by linking the XSD and DTD to each other so that I would not have to comment out one of the DTD or XSD reference in the XML document while validating it but I kept getting error message.

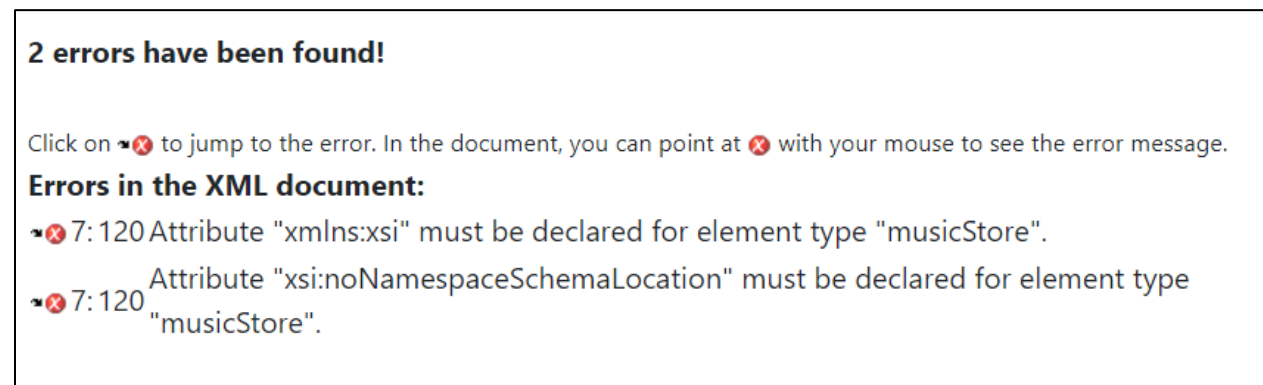


Figure 19: Error during validation

After doing some research, I found out that in order to validate XML document against the XSD and DTD file at the same time, the attribute of the schema XSD need to be added in the DTD file. Once I did this, I was able to get rid of the errors and was able to validate the XML document against the XSD and DTD files at the same time without any errors.

```
<!--element musicStore contains schema attributes, the attributes are of type 'CDATA' and the values are '#REQUIRED'-->
<!ATTLIST musicStore xmlns:xsi CDATA #REQUIRED xsi:noNamespaceSchemaLocation CDATA '#REQUIRED'>
<!--element storeInfo contains attribute infoType, the attribute is of type 'CDATA' and the value is '#FIXED'-->
<!ATTLIST storeInfo infoType CDATA #FIXED "About">
```

Figure 20: Adding the schema attributes to DTD

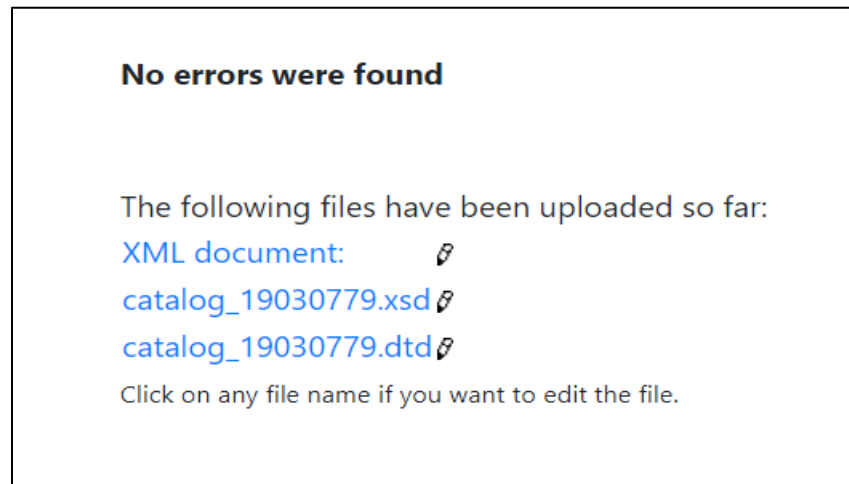


Figure 21: Validation successful after adding schema attributes to DTD

The second problem that I faced was that I could not get the content of the song to display in box type manner.

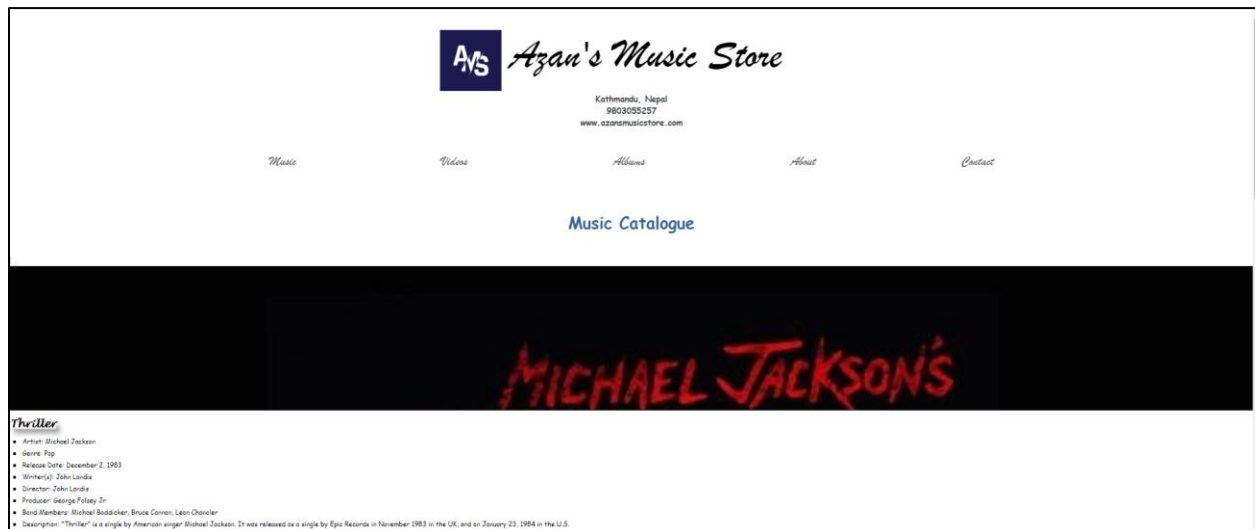


Figure 22: XML content not displaying as expected.

While looking up possible solutions to this problem, I learned about CSS flex-box, I learned that that by setting the display to flex in the CSS, I could organize the content in the way I wanted. Flex-box makes it easier to design the structure of a web page without having to use float.

```
songs{
  display: flex; /* Set display to flex */
  flex-flow: row wrap; /* wrap rows */
  justify-content: space-around;
}
```

Figure 23: Setting display to flex

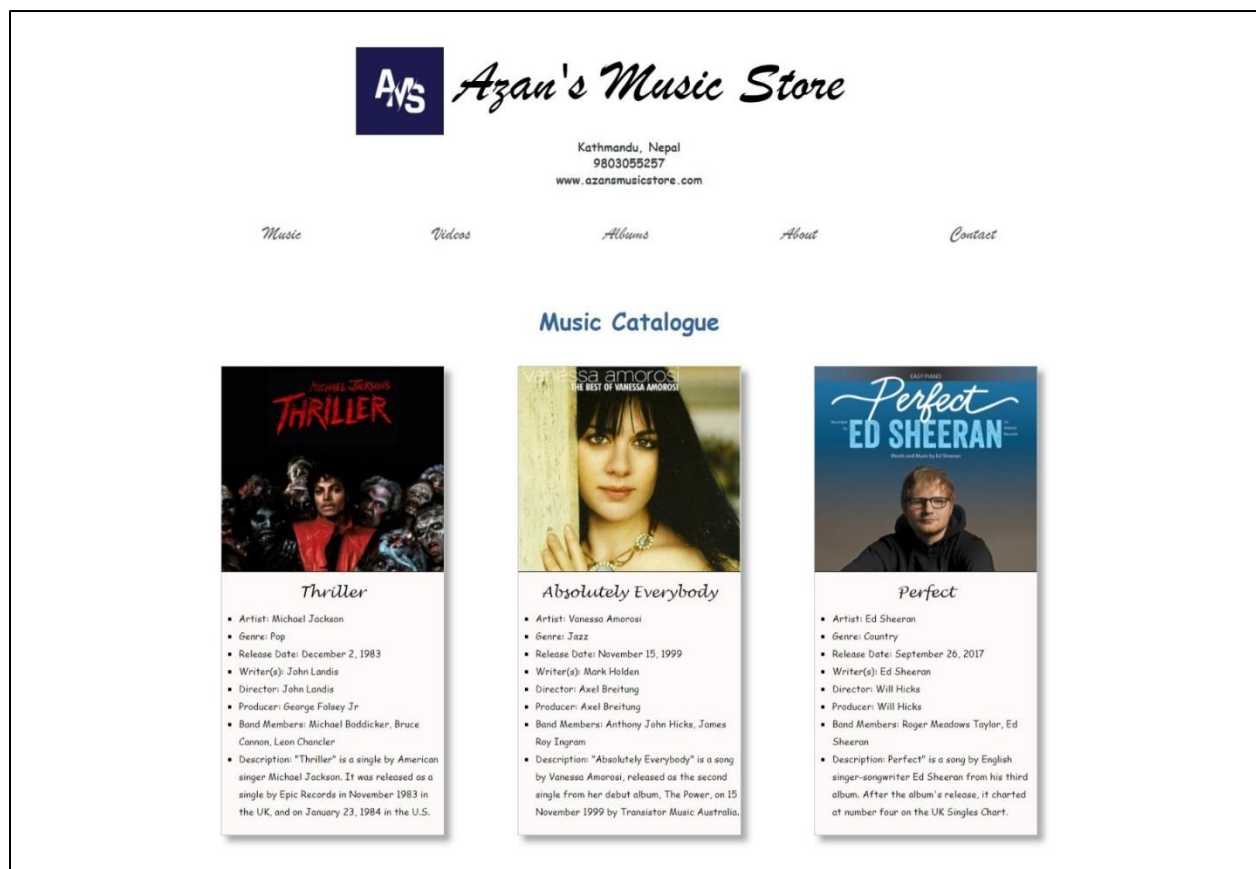


Figure 24: XML content after setting display to flex in CSS

In this way, by using flex-box, I was able to put the contents of the XML file in an organized manner and make it more visually pleasing to the user.

There were various other minor problems that I encountered while developing this coursework. I was able to rectify those problems by going through the study material provided by my tutors and doing my own research on the internet.

Conclusion

In conclusion, the coursework has been completed. A system for a music store has been successfully modelled. The XML, XSD, DTD and CSS files have been created using Sublime Text Editor. All the necessary testing has been done to make sure all the necessary requirement given in the coursework scenario has been completed.

Overall, after completing this coursework, I learned a lot about XML as it's a completely new language to me. I also learned how to create a tree diagram also how to validate the structure of an XML document using Schema and DTD and was also able to revise some CSS concept which I had previously learned thanks to this coursework. Lastly, I would like to thank my tutors for their guidance as it would not have been possible to complete this coursework without them.

References

Career Ride, 2016. *Career Ride*. [Online]

Available at: <https://www.careerride.com/xml-schema-vs-dtd.aspx>

[Accessed 6 May 2021].

GeeksforGeeks, 2020. *GeeksforGeeks*. [Online]

Available at: <https://www.geeksforgeeks.org/difference-between-document-type-definition-dtd-and-xml-schema-definition-xsd/>

[Accessed 6 May 2021].

Informit, 2001. *Informit*. [Online]

Available at: <https://www.informit.com/articles/article.aspx?p=24614&seqNum=3>

[Accessed 6 May 2021].

Javatpoint, 2018. *Javatpoint*. [Online]

Available at: <https://www.javatpoint.com/dtd-vs-xsd>

[Accessed 6 May 2021].

Javatpoint, 2018. *JavaTpoint*. [Online]

Available at: <https://www.javatpoint.com/xml-tree-structure#:~:text=An%20XML%20document%20has%20a,referred%20as%20an%20XML%20tree.&text=A%20tree%20structure%20contains%20root,nodes%20starting%20from%20the%20root.>

[Accessed 6 May 2021].

Joan, B., 2017. *Difference Between*. [Online]

Available at: <http://www.differencebetween.net/technology/difference-between-xml-schema-and-dtd/>

[Accessed 6 May 2021].

Liquid Technologies, 2021. *Liquid Technologies*. [Online]

Available at: <https://www.liquid-technologies.com/xml-schema-tutorial/xsd-elements-attributes>

[Accessed 6 May 2021].

Tutorials Point, 2021. *Tutorials Point*. [Online]

Available at: https://www.tutorialspoint.com/xml/xml_dtds.htm

[Accessed 6 May 2021].

Tutorials Point, 2021. *Tutorials Point*. [Online]

Available at: https://www.tutorialspoint.com/xml/xml_overview.htm

[Accessed 6 May 2021].

W3, 2016. *W3*. [Online]

Available at: <https://www.w3.org/standards/webdesign/htmlcss>

[Accessed 6 May 2021].