



Module Code & Module Title

CS5004NT Emerging Programming Platforms and Technologies

Assessment Weightage & Type

30% Individual Coursework

Year and Semester

2020-21 spring

Student Name: Prem Kumar Mehta

College ID: NP05CP4S210153

London-Met ID: 20048472

Assignment Due Date: 2022/05/05

Assignment Submission Date: 2022/05/04

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

Acknowledgement

First and foremost, I'd want to express our deep and sincere gratitude to Mr. Pratik Panta Sir, our module leader, for his enthusiasm and support during our Emerging Programming Platforms and Technologies and Module study and supervision. He helped us with our studies and conducted research on a variety of issues related to our projects.

They have taught us how to do research and how to grasp the work without causing any damage. Working and studying under his direction and guidance was a wonderful honor and privilege. We're appreciative for everything they've done for us.

Finally, I want to express our gratitude to the institution and its resources, as well as the professors who have assisted us in grappling with all of the challenges and doubts that we have encountered throughout the course of our work.

Abstract

This is a 30 percent (%) of the overall grade group and individual coursework for the module Emerging Programming Platforms and Technologies.

This coursework is all about to create and Gift Card Store that sell gift cards or vouchers. So to carry out the development Visual Studio Code was used. Using the Visual Studio Code, an XML file, XSD or schema file and DTD file was developed.

Table of Contents

1. Introduction.....	1
2. Tree Diagram.....	2
3. Development	4
3.1. XML.....	4
3.2. Schema Content	7
3.3. Document Type Definition (DTD).....	12
3.4. Cascading Style Sheets (CSS).....	14
4. Testing	19
5. Difference between Schemas and DTDs	39
6. How the Coursework was developed?	40
7. Critical Analysis and Conclusion.....	44
8. References	46

List of Figures

Figure 1: Tree Diagram.....	3
Figure 2: Evidence of Test Case 1 that shows the XML file rendered in the Browser	20
Figure 3: Inserting the XML file for the validation	22
Figure 4: Entering the Schema file for the validation	23
Figure 5: No error was found for the test case 2	24
Figure 6: Inserting the XML file for the validation	25
Figure 7: Inserting the DTD file for the validation	26
Figure 8: no errors was found for the test case 3	27
Figure 9: XML file render with the CSS in the browser	28
Figure 10: Inserting invalid value excpet numerated value	29
Figure 11: Evidence for the Test Case 5 which shows error	30
Figure 12: Inserting invalid value expect pattern value	31
Figure 13: Evidence for test case 3 which shows no errors.....	31
Figure 14: Removing discount element from a GiftCard element.....	33
Figure 15: Evidence of test case 7 with no errors.....	34
Figure 16: Inserting invalid (negative integer) value in quantity attribute	35
Figure 17: Evidence for test case 8 with errors	35
Figure 18: Inserting the Same CardID twice	37
Figure 19: Evidence for Test Case 9 with errors	38
Figure 20: Draw.io.....	41
Figure 21: Ms-word	41
Figure 22: Visual Studio Code.....	42
Figure 23: XML Validation.....	43

List of Tables

Table 1: Test Case 1	19
Table 2: Test Case 2	21
Table 3: Test Case 3	24
Table 4: Test Case 4	28
Table 5: Test Case 5	29
Table 6: Test Case 6	30
Table 7: Test Case 7	32
Table 8: Test Case 8	34
Table 9: Test Case 9	36

1. Introduction

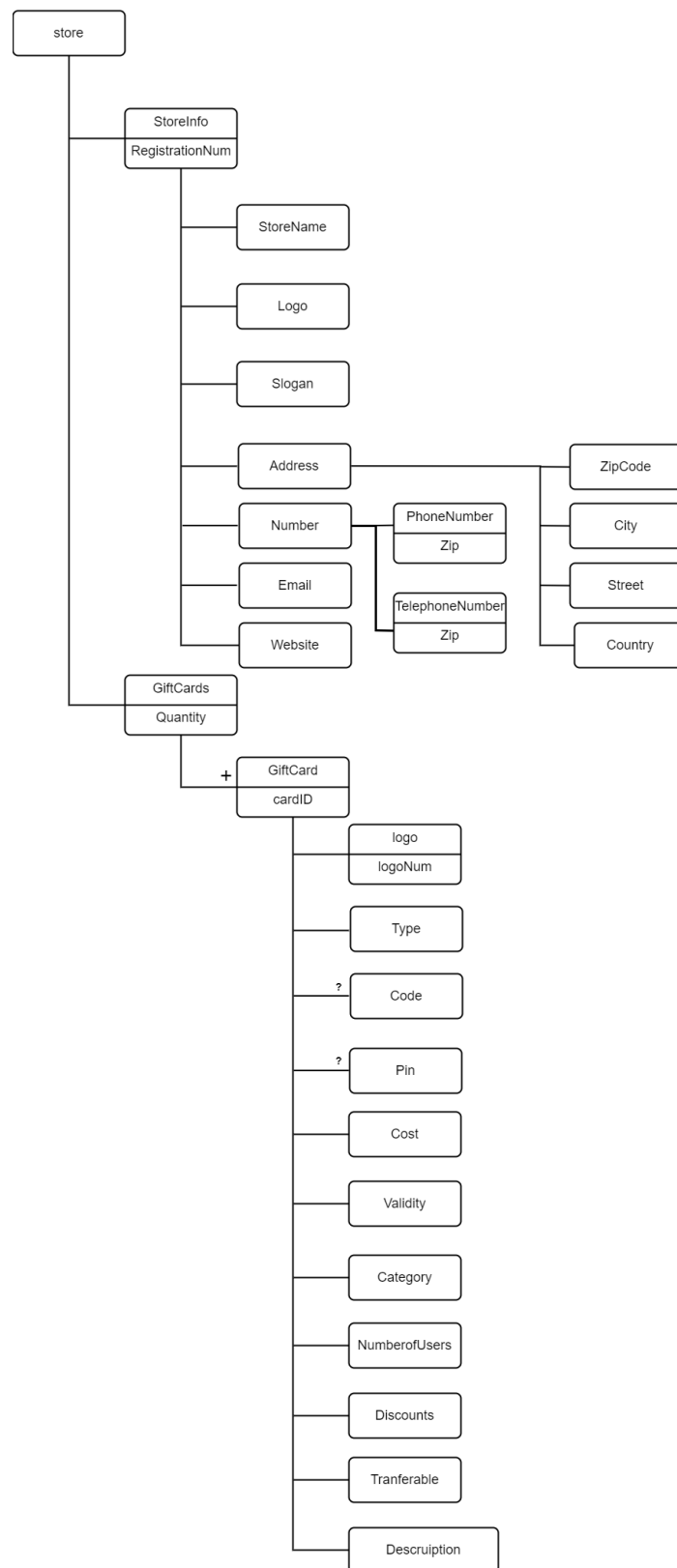
This project is the second individual coursework of the module Emerging Programming and Platforms and Technologies. As an XML developer, we were instructed to model a system for a gift card store that sells digital gift cards or gift vouchers which is a form of payment that can be used to make purchases at retail stores, gas stations, restaurants and other location. The project was created using an XML document, an XSD or schema and a DTD for validation, as well as CSS to style the instance document

This Coursework is all about designing or creating a model system for Gift Card Store who wants to sell their gift cards and vouchers where all the information about different gift cards or vouchers will be available with just a few clicks. The business is a selling gift cards platform and the store manager provided the essential information for building the system model as well as the scenario. Each item, the gift cards are properly labeled with its description, type, cost, validity and so on. XML together with its schema, DTD, and CSS were utilized to validate the data or information in the instance document, as well as to offer good decoration and styling to the instance documentation. Aside from the information provided by the store manager, some information was added by myself to have a positive impact on the gift cards.

In general, the coursework concentrates around the building of a well-based gift card store using an XML document that is located in London. There is a wide range of gift cards based on card id number. All of those data was rendered to the web browser through the production of an XML document, which aids in the methodical and structured arrangement of all the data from numerous cards. The CSS was a really useful technique that provided a proper style for the user to enjoy while browsing the page. At last, there was schema and DTD to verify all the data that were going to give a logical error with respect to its types or nature of the element.

2. Tree Diagram

A Tree Diagram is a visual representation of hierarchy in the form of a tree. It reflects structural relationships in a data set and enables for faster insertion, deletion, and searching operations than an array or linked list. It offers a versatile means of storing and moving data. A Tree Diagram's structure typically comprises of elements such as a root node, a member with no superior/parent, and a child node (Yin, 2012).

*Figure 1: Tree Diagram*

3. Development

The Development Part Consists of various Elements like:

- Tree diagram
- XML content
- Schema Content
- Document Type Definition (DTD)
- Cascading Style Sheet (CSS)

3.1. XML

XML is a markup language with a hierarchical structure. It stands for eXtensible Markup Language which defines data by using opening and closing tags and is much like HTML. It is a text based markup language derived from Standard Generalized Markup Language (SGML). It was designed to store and exchange data, and due to its immense versatility, it is utilized for everything from paperwork to images (Kenlon, 2021).

The XML document that I have created is provided below:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="catalog_20048472.css" type="text/css"?>
<!DOCTYPE store SYSTEM "catalog_20048472.dtd">
<store xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="catalog_20048472.xsd">
  <StoreInfo RegistrationNum="02345789">
    <StoreName>Meow Gift Store</StoreName>
    <Logo> </Logo>
    <Slogan>Gifts to express your feeling</Slogan>
    <Address>
      <ZipCode>5210</ZipCode>
      <City>269 Regent St</City>
      <Street>London W1B 2EX</Street>
      <Country>United Kingdom</Country>
    </Address>
    <Number>
      <PhoneNumber zip="+447">436400589</PhoneNumber>
      <TelephoneNumber zip="+44">2074954843</TelephoneNumber>
    </Number>
    <Email>info@Meow.com</Email>
```

```

    <Website>www.Meow.com</Website>
  </StoreInfo>

  <GiftCards Quantity="4">
    <GiftCard cardID="GC1">
      <logo logoNum="01"></logo>
      <Type>Physical</Type>
      <Pin>78046-34905</Pin>
      <Cost>£40.00</Cost>
      <Validity>2022-06-13</Validity>
      <Category>online payments</Category>
      <NumberOfUsers>4</NumberOfUsers>
      <Discounts>2</Discounts>
      <Transferable>Not_Transferable</Transferable>
      <Description>
        <![CDATA[
          Its Small but I hope this makes the next few weeks just a
little easier.
        ]]>
      </Description>
    </GiftCard>

    <GiftCard cardID="GC2">
      <logo logoNum="02"></logo>
      <Type>Digital</Type>
      <Code>JN893-8JKIE</Code>
      <Cost>£25.00</Cost>
      <Validity>2022-06-13</Validity>
      <Category>shopping mall</Category>
      <NumberOfUsers>5</NumberOfUsers>
      <Discounts>5</Discounts>
      <Transferable>Not_Transferable</Transferable>
      <Description>
        <![CDATA[
          Its Small but I hope this makes the next few weeks just a
little easier.
        ]]>
      </Description>
    </GiftCard>

    <GiftCard cardID="GC3">
      <logo logoNum="03"></logo>
      <Type>Physical</Type>
      <Pin>97543-77432</Pin>
      <Cost>£20.00</Cost>

```

```
<Validity>2022-06-13</Validity>
<Category>restaurants</Category>
<NumberOfUsers>3</NumberOfUsers>
<Discounts>4</Discounts>
<Transferable>Not_Transferable</Transferable>
<Description>
  <![CDATA[
    Its Small but I hope this makes the next few weeks just a
little easier.
  ]]>
</Description>
</GiftCard>

<GiftCard cardID="GC4">
  <logo logoNum="04"></logo>
  <Type>Digital</Type>
  <Code>U886R-MNUE5</Code>
  <Cost>£30.00</Cost>
  <Validity>2022-06-13</Validity>
  <Category>gift stores</Category>
  <NumberOfUsers>2</NumberOfUsers>
  <Discounts>3</Discounts>
  <Transferable>Transferable</Transferable>
  <Description>
    <![CDATA[
      Its Small but I hope this makes the next few weeks just a
little easier.
    ]]>
  </Description>
</GiftCard>
</GiftCards>
</store>
```

3.2. Schema Content

An XML schema known as XML Schema Definition (XSD), is a framework document used to describe and validate the structure and the content of XML data. An XSD formally describes the elements, attributes and data types in an XML document (Singh, 2021). It supports NameSpace and is similar to a database schema that describes the data in a database. An XML document with correct syntax is called “Well Formed” and an XML document validated against an XML Schema is both “Well Formed” and “Valid” (Lawton, 2021).

The Purpose of an XML Schema is to define the legal building blocks of an XML document:

- The elements and attributes that can be found in a document.
- The number and the sequence of child elements found in document.
- The different data types for Elements and Attributes.
- The value for Elements and Attributes are default and fixed (W3Schools, 2021).

The Schema Content which I have created is give below:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="store" type="storeType" />
  <xs:complexType name="storeType">
    <xs:sequence>
      <xs:element name="StoreInfo" type="StoreInfoType" />
      <xs:element name="GiftCards" type="GiftCardsType" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="StoreInfoType">
    <xs:group ref="StoreInfoChild"></xs:group>
    <xs:attribute type="xs:positiveInteger" name="RegistrationNum" />
  </xs:complexType>

  <xs:group name="StoreInfoChild">
    <xs:sequence>
      <xs:element name="StoreName" type="xs:string" minOccurs="1"
maxOccurs="1" />

```

```

        <xs:element name="Logo" type="xs:string" minOccurs="1"
maxOccurs="1" />
        <xs:element name="Slogan" type="xs:string" minOccurs="1"
maxOccurs="1" />
        <xs:element name="Address" type="AddressType" minOccurs="1"
maxOccurs="1" />
        <xs:element name="Number" type="NumberType" minOccurs="1"
maxOccurs="1" />
        <xs:element name="Email" type="EmailRestriction" minOccurs="1"
maxOccurs="1" />
        <xs:element name="Website" type="websiteRestriction" minOccurs="1"
maxOccurs="1" />
    </xs:sequence>
</xs:group>

<xs:complexType name="AddressType">
    <xs:sequence>
        <xs:element name="ZipCode" type="xs:positiveInteger" minOccurs="1"
maxOccurs="1" />
        <xs:element name="City" type="xs:string" minOccurs="1"
maxOccurs="1" />
        <xs:element name="Street" type="xs:string" minOccurs="1"
maxOccurs="1" />
        <xs:element name="Country" type="xs:string" minOccurs="1"
maxOccurs="1" />
    </xs:sequence>
</xs:complexType>

<xs:complexType name="NumberType">
    <xs:sequence>
        <xs:element name="PhoneNumber" type="PhoneNumberType" minOccurs="1"
maxOccurs="1" />
        <xs:element name="TelephoneNumber" type="TelephoneNumberType"
minOccurs="1" maxOccurs="1" />
    </xs:sequence>
</xs:complexType>

<xs:complexType name="PhoneNumberType">
    <xs:simpleContent>
        <xs:extension base="xs:positiveInteger">
            <xs:attribute type="xs:string" name="zip" use="required">
</xs:attribute>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>

```

```

<xs:complexType name="TelephoneNumberType">
  <xs:simpleContent>
    <xs:extension base="xs:positiveInteger">
      <xs:attribute type="xs:string" name="zip" use="required" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:simpleType name="EmailRestriction">
  <xs:restriction base="xs:string">
    <xs:pattern value="[A-Za-z]+@[A-Za-z]+\.[A-Za-z]+" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="websiteRestriction">
  <xs:restriction base="xs:string">
    <xs:pattern value="www.[A-Za-z]+\.[a-z]+" />
  </xs:restriction>
</xs:simpleType>

<xs:complexType name="GiftCardsType">
  <xs:sequence>
    <xs:element type="GiftCardType" name="GiftCard"
maxOccurs="unbounded" minOccurs="0" />
  </xs:sequence>
  <xs:attribute type="xs:positiveInteger" name="Quantity" />
</xs:complexType>

<xs:complexType name="GiftCardType">
  <xs:group ref="GiftCardChild"></xs:group>
  <xs:attribute type="xs:string" name="cardID" />
</xs:complexType>

<xs:group name="GiftCardChild">
  <xs:sequence>
    <xs:element name="logo" type="LogoType" minOccurs="1" maxOccurs="1"
/>
    <xs:element name="Type" type="Type" minOccurs="1" maxOccurs="1" />
    <xs:element name="Pin" type="PinRestriction" minOccurs="0"
maxOccurs="1" />
    <xs:element name="Code" type="CodeRestriction" minOccurs="0"
maxOccurs="1" />
    <xs:element name="Cost" type="xs:string" minOccurs="1"
maxOccurs="1" />

```

```

        <xs:element name="Validity" type="ValidityRestriction"
minOccurs="1" maxOccurs="1" />
        <xs:element name="Category" type="xs:string" minOccurs="1"
maxOccurs="1" />
        <xs:element name="NumberOfUsers" type="NumberOfUsersRestriction"
minOccurs="1" maxOccurs="1" />
        <xs:element name="Discounts" type="DiscountRestriction"
minOccurs="0" maxOccurs="1" />
        <xs:element name="Transferable" type="TransferableType"
minOccurs="0" maxOccurs="1" />
        <xs:element name="Description" type="xs:string" minOccurs="1"
maxOccurs="1" />
    </xs:sequence>
</xs:group>

<xs:complexType name="LogoType">
    <xs:simpleContent>
        <xs:extension base="xs:string">
            <xs:attribute type="xs:positiveInteger" name="logoNum"
use="required" />
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>

<xs:simpleType name="Type">
    <xs:restriction base="xs:string">
        <xs:enumeration value="Digital" />
        <xs:enumeration value="Physical" />
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="PinRestriction">
    <xs:restriction base="xs:string">
        <xs:pattern value="[0-9]{5}-[0-9]{5}"></xs:pattern>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="CodeRestriction">
    <xs:restriction base="xs:string">
        <xs:pattern value="[A-Z0-9]{5}-[A-Z0-9]{5}"></xs:pattern>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="ValidityRestriction">
    <xs:restriction base="xs:string">

```



```
        <xs:pattern value="[0-9]{4}-[0-9]{2}-[0-9]{2}"/>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="NumberOfUsersRestriction">
    <xs:restriction base="xs:positiveInteger">
        <xs:minInclusive value="1"/>
        <xs:maxInclusive value="6"/>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="DiscountRestriction">
    <xs:restriction base="xs:positiveInteger">
        <xs:minInclusive value="2"/>
        <xs:maxInclusive value="6"/>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="TransferableType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="Transferable"/>
        <xs:enumeration value="Not_Transferable"/>
    </xs:restriction>
</xs:simpleType>

</xs:schema>
```

3.3. Document Type Definition (DTD)

A DTD, which stands for Document Type Definition, is a technique to properly describe the XML language, defining the structure as well as the legal elements and attributes of an XML document. It is a set of markup assertions that define a type of document for the SGML family, which includes GML, SGML, HTML, and XML. It can be stated as an inline or external recommendation within an XML document. It specifies how many times a node should appear and how their child nodes should be arranged (Refsnes, 2021).

The Document Type Definition (DTD) content that I have created is given below:

```
<?xml version="1.0" encoding="UTF-8"?>

<!ELEMENT store (StoreInfo,GiftCards)>
<!ATTLIST store xmlns:xsi CDATA #FIXED 'http://www.w3.org/2001/XMLSchema-
instance' xsi:noNamespaceSchemaLocation CDATA #REQUIRED>

<!ELEMENT StoreInfo (StoreName,Logo,Slogan,Address,Number,Email,Website)>
<!ATTLIST StoreInfo RegistrationNum CDATA #REQUIRED>
<!ELEMENT StoreName (#PCDATA)>
<!ELEMENT Logo (#PCDATA)>
<!ELEMENT Slogan (#PCDATA)>
<!ELEMENT Address (ZipCode,City,Street,Country)>
<!ELEMENT ZipCode (#PCDATA)>
<!ELEMENT City (#PCDATA)>
<!ELEMENT Street (#PCDATA)>
<!ELEMENT Country (#PCDATA)>
<!ELEMENT Number (PhoneNumber,TelephoneNumber)>
<!ELEMENT PhoneNumber (#PCDATA)>
<!ATTLIST PhoneNumber zip CDATA #REQUIRED>
<!ELEMENT TelephoneNumber (#PCDATA)>
<!ATTLIST TelephoneNumber zip CDATA #REQUIRED>
<!ELEMENT Email (#PCDATA)>
<!ELEMENT Website (#PCDATA)>

<!ELEMENT GiftCards (GiftCard+)>
<!ATTLIST GiftCards Quantity CDATA #REQUIRED>
<!ELEMENT GiftCard (logo,Type, Pin?, Code?,
Cost,Validity,Category,NumberOfUsers,Discounts,Transferable,Description)>
<!ATTLIST GiftCard cardID ID #REQUIRED>
<!ELEMENT logo EMPTY>
<!ATTLIST logo logoNum CDATA #REQUIRED>
```

```
<!ELEMENT Type (#PCDATA)>  
<!ELEMENT Pin (#PCDATA)>  
<!ELEMENT Code (#PCDATA)>  
<!ELEMENT Cost (#PCDATA)>  
<!ELEMENT Validity (#PCDATA)>  
<!ELEMENT Category (#PCDATA)>  
<!ELEMENT NumberofUsers (#PCDATA)>  
<!ELEMENT Discounts (#PCDATA)>  
<!ELEMENT Transferable (#PCDATA)>  
<!ELEMENT Description (#PCDATA)>
```

3.4. Cascading Style Sheets (CSS)

CSS is an abbreviation for Cascading Style Sheets. It is the language used to describe the presentation of Web pages, such as colors, layout, and fonts, which allows us to offer our web pages to users. CSS is most commonly used to style web pages written in markup languages. It defines how HTML or XML items should appear on a screen, on paper, or in other media. It saves you a lot of time and allows you to alter the layout of multiple web pages at once (Parvez, 2021).

```
@import url('https://fonts.googleapis.com/css?family=Baloo+Thambi');
store{
    font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen,
    Ubuntu, Cantarell, 'Open Sans', 'Helvetica Neue', sans-serif;
    background-color: #EDC2D8FF;
    padding: 2em 0;
}
StoreName{
    display: flex;
    justify-content: center;
    align-items: center;
    color: #FC766AFF;
    font-family:'Baloo Thambi', cursive;
    font-size: 50px;
    animation: leftright 2s linear infinite;
    animation-direction: alternate;
    border: 1px solid black;
    border-radius: 30px;
    background-color: #B0B8B4FF;
    width: 500px;
    height: 65px;
    -webkit-border-radius:30px;
    -ms-border-radius: 30px;
    -o-border-radius:30px;
    text-transform: uppercase;
    margin-left: 350px;
    margin-top: 30px;
}
@keyframes leftright{
    from{padding-left: 40px;}
    to{padding-right: 40px;}
}
Slogan {
    margin-bottom: -50px;
```

```
    display: flex;
    color: #FC766AFF;
    font-family: 'Baloo Thambi', cursive;
    font-size: 50px;
}

Slogan {
    font-family: Georgia, 'Times New Roman', Times, serif;
    margin-bottom: -5em;
    margin-top: 30px;
    margin-left: 470px;
    font-size: 20px;
}

StoreInfo{
    display: grid;
    margin-bottom: 5em;
}

StoreInfo{
    margin-right: auto;
    margin-left: auto;
    flex-wrap: wrap;
    font-weight: bold;
    border-bottom: 2px solid rgb(0, 0, 0);
    border-style: double;
    border-color: brown;
    padding-bottom: 30px;
    padding-top: 30px
}

Logo {
    position: absolute;
    background: url(cat.gif) no-repeat;
    background-size: 200px;
    width: 200px;
    height: 200px;
    margin-left: 100px;
    border-radius: 100px;
    margin-top: -15px;
}

Address, Number, Email, Website{
    text-align: right;
    padding-right: 2em;
    font-family: 'Times New Romans';
```

```

    font-size:30;
    color:#184A45FF;
}

GiftCards{
    font-size: 25px;
    margin-right: -5em;
}
GiftCard{
    display: grid;
    margin-bottom: 10px;
    padding: 50px;
}
logo[logoNum="01"], logo[logoNum="02"], logo[logoNum="03"],logo[logoNum="04"]
{
    width: 1000px;
    height: 500px;
}
logo[logoNum="01"]{
    background: url(/40.png) no-repeat;
    background-size: 450px;
}
logo[logoNum="02"]{
    background: url(/25.png) no-repeat;
    background-size: 450px;
}
logo[logoNum="03"]{
    background: url(/20.png) no-repeat;
    background-size: 450px;
}
logo[logoNum="04"]{
    background: url(/30.png) no-repeat;
    background-size: 450px;
}

logo,Type,Pin,Code,Cost,Validity,Category,NumberofUsers,Discounts,Usuability,Tr
ansferable,
Description{
    font-family: papyrus;
    display: list-item;
    list-style: none;
    width: 25%;
    text-align: left;
    line-height: 30px;
    position: relative;

```

```

    color: #184A45FF;
    font-weight: 1000;
    padding-top: 7x;
  }

  Type,Pin,Code,Cost,Validity,Category,NumberOfUsers,Discounts,Usuability,Trans
ferable,
Description {
    margin-left: 700px;
}
logo {
    margin-bottom: -500px;
    margin-left: 200px;
}

Description{
    font-size: 20px;
    text-align: left;
    position: relative;
    margin-top: 50px;
    text-align: justify;
    color: #184A45FF;
    font-size: 30;
    margin-top: 30px;
    font-family:'Baloo Thambi', cursive;
}

Type::before{
    content: "Type: ";
}
Pin::before {
    content: "Pin: ";
}
Code::before {
    content: "Code: ";
}
Cost::before{
    content: "Cost: ";
}
Validity::before{
    content: "Validity: ";
}
Category::before{
    content: "Category: ";
}

```

```
}  
NumberOfUsers::before{  
    content: "NumberOfUsers: ";  
}  
Discounts::before{  
    content: "Discounts: ";  
}  
Transferable::before{  
    content: "Transferable: ";  
}  
Discounts::after{  
    content: "%";  
}
```


4. Testing

Testing is the process of examining a program or its components with the goal of determining if they meet the criteria. Testing entails running a system to look for faults, errors and requirements that they aren't present in the actual requirements.

4.1 Test 1: Checking if the XML Document is well-formed or not.

Table 1: Test Case 1

Objective	To check the XML file is well formed or not.
Action	XML file was opened in the browser without attaching the CSS to it.
Expected Result	XML document will be rendered in the browser.
Actual Result	XML document was render in the browser.
Conclusion	Test was successful.

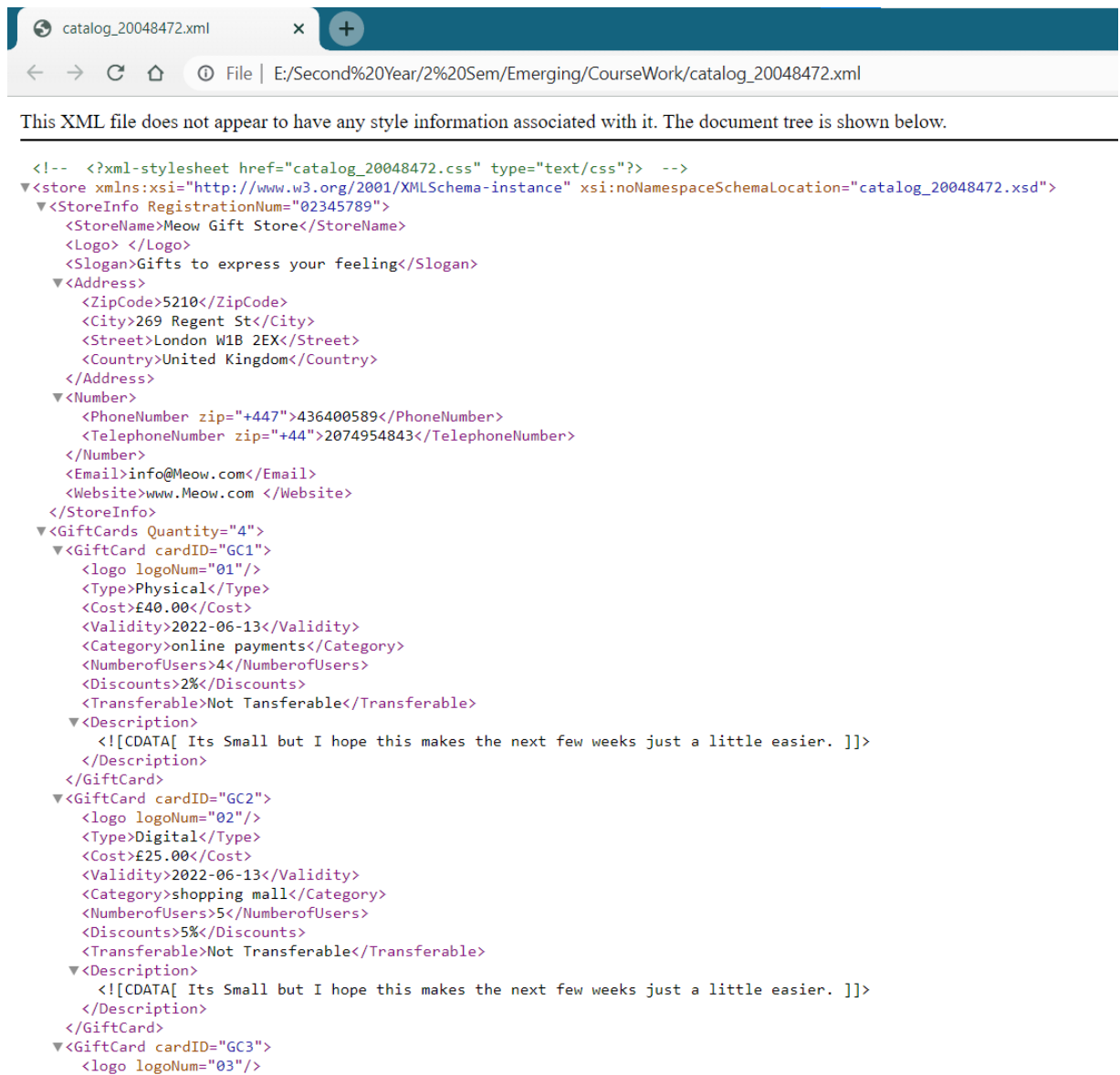


Figure 2: Evidence of Test Case 1 that shows the XML file rendered in the Browser

4.2 Test 2: Checking Validity of XML file including Schema

Table 2: Test Case 2

Objective	To check whether the XML file is valid with schema or not.
Action	The XML file and the Schema file was validate in the link (www.xmlvalidation.com)
Expected Result	XML and schema will be valid and should not contain any errors.
Actual Result	It was valid with no errors.
Conclusion	Test was successful.

Validate an XML file

Read [here how to validate your XML files](#) (including referenced DTDs) online with just a few mouse clicks.

Please copy your XML document in here:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="catalog_20048472.css" type="text/css"?>
<store xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="catalog_20048472.xsd">
  <StoreInfo RegistrationNum="02345789">
    <StoreName>Meow Gift Store</StoreName>
    <Logo> </Logo>
    <Slogan>Gifts to express your feeling</Slogan>
    <Address>
      <ZipCode> 5210</ZipCode>
```

Figure 3: Inserting the XML file for the validation

Validate an XML file

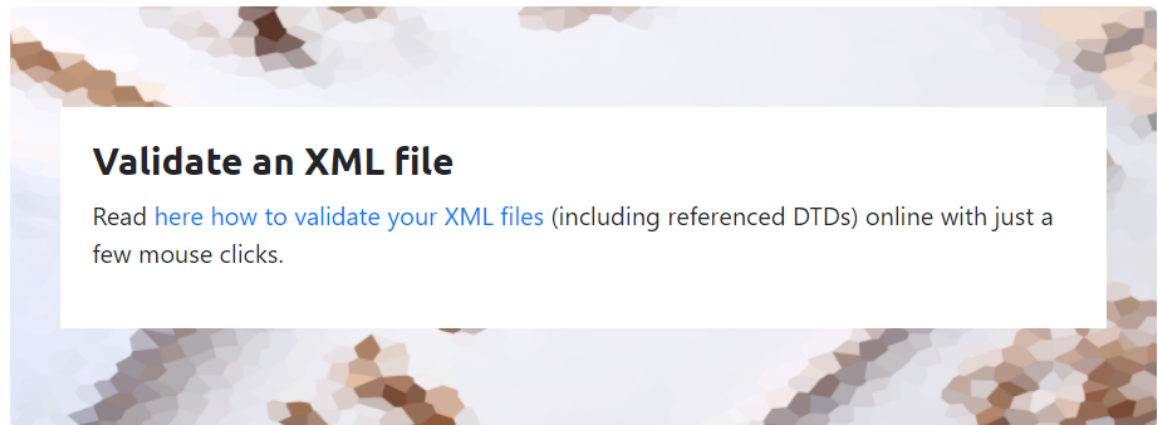
Read [here how to validate your XML files](#) (including referenced DTDs) online with just a few mouse clicks.

The file catalog_20048472.xsd is being referenced. Please copy it in here, so that the validation can continue:


```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="store" type="storeType" />

  <xs:complexType name="storeType">
    <xs:sequence>
      <xs:element name="StoreInfo" type="StoreInfoType" />
      <xs:element name="GiftCards" type="GiftCardsType" />
    </xs:sequence>
  </xs:complexType>
```

Figure 4: Entering the Schema file for the validation

**No errors were found**

The following files have been uploaded so far:

[XML document:](#) 

[catalog_20048472.xsd](#) 

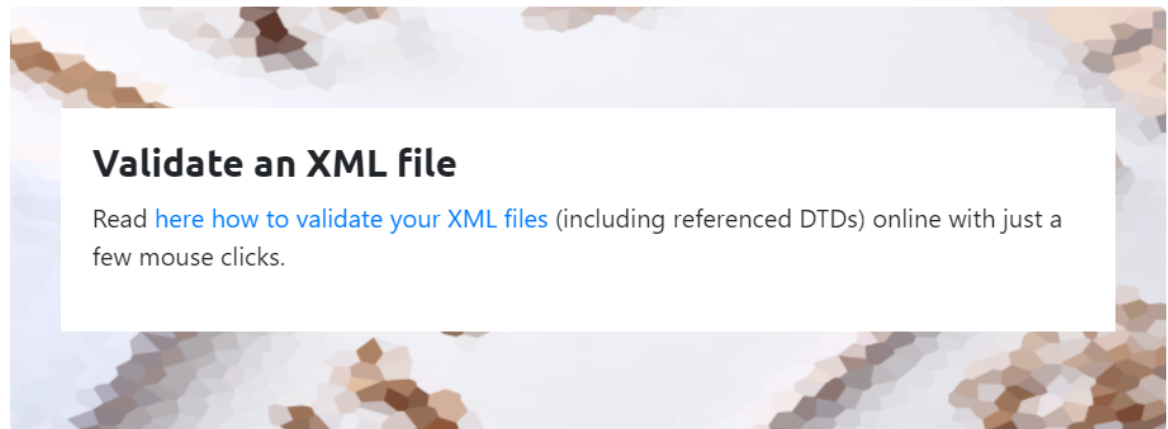
Click on any file name if you want to edit the file.

Figure 5: No error was found for the test case 2

4.3 Test 3: Check validity of XML file with DTD

Table 3: Test Case 3

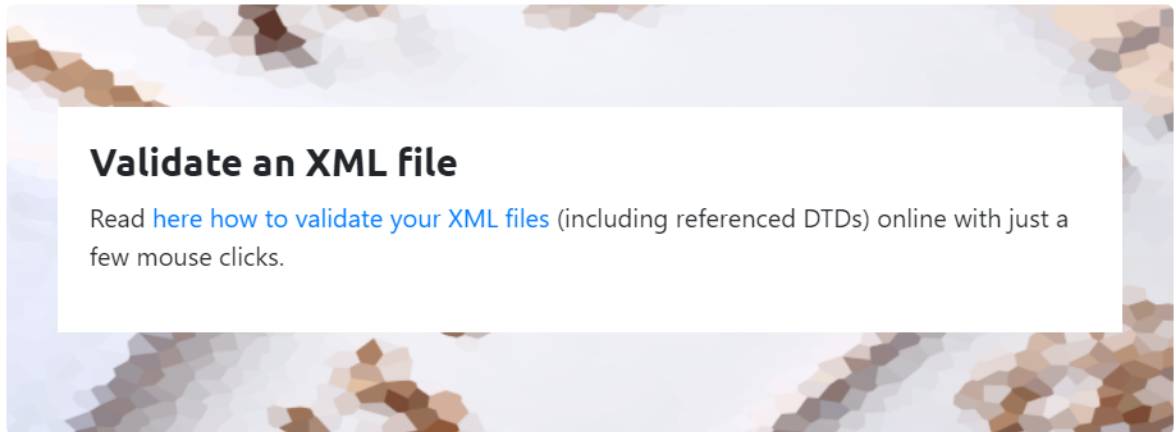
Objective	To check whether the XML file is valid with DTD or not.
Action	XML and DTD was tested in the link (www.xmlvalidation.com)
Expected Result	XML and DTD will be valid with no errors.
Actual Result	It was validated with no errors.
Conclusion	Test was successful.



Please copy your XML document in here:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="catalog_20048472.css" type="text/css"?>
<!DOCTYPE store SYSTEM "Catalog_20048472.dtd">
<store>
  <StoreInfo RegistrationNum="02345789">
    <StoreName>Meow Gift Store</StoreName>
    <Logo> </Logo>
    <Slogan>Gifts to express your feeling</Slogan>
    <Address>
      <ZipCode>5210</ZipCode>
```

Figure 6: Inserting the XML file for the validation



The file Catalog_20048472.dtd is being referenced. Please copy it in here, so that the validation can continue:

```
<?xml version="1.0" encoding="UTF-8"?>  
  
<!ELEMENT store (StoreInfo,GiftCards)>  
  
<!ELEMENT StoreInfo (StoreName,Logo,Slogan,Address,Number,Email,Website)>  
<!ATTLIST StoreInfo RegistrationNum CDATA #REQUIRED>  
<!ELEMENT StoreName (#PCDATA)>  
<!ELEMENT Logo (#PCDATA)>  
<!ELEMENT Slogan (#PCDATA)>
```

Figure 7: Inserting the DTD file for the validation

Validate an XML file

Read [here how to validate your XML files](#) (including referenced DTDs) online with just a few mouse clicks.

No errors were found

The following files have been uploaded so far:

[XML document:](#) 

[Catalog_20048472.dtd](#) 

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

Figure 8: no errors was found for the test case 3

4.4 Test 4: Running the XML file with CSS in the browser

Table 4: Test Case 4

Objective	To run the XML file attaching the CSS.
Action	The XML document was opened in the browser.
Expected Result	The XML document will render with CSS without any errors.
Actual Result	The XML documents was rendered successfully with CSS.
Conclusion	Test was Successful.

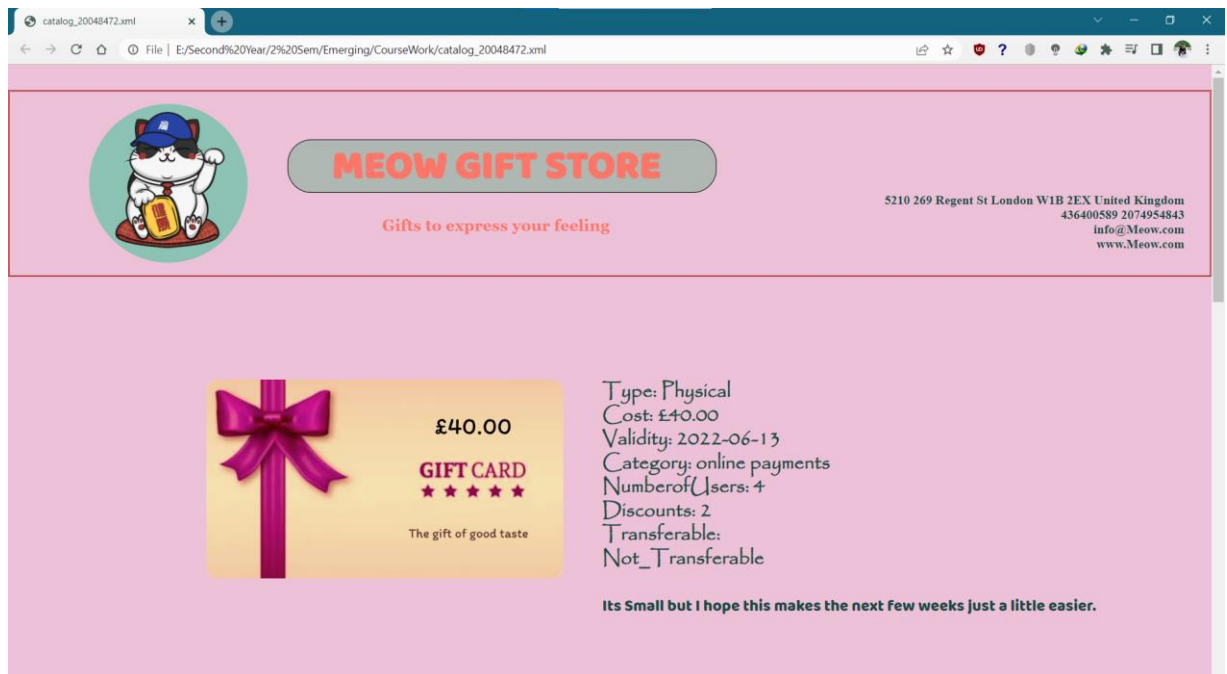


Figure 9: XML file render with the CSS in the browser

4.5 Test 5: Checking the working of enumeration value in type element.

Table 5: Test Case 5

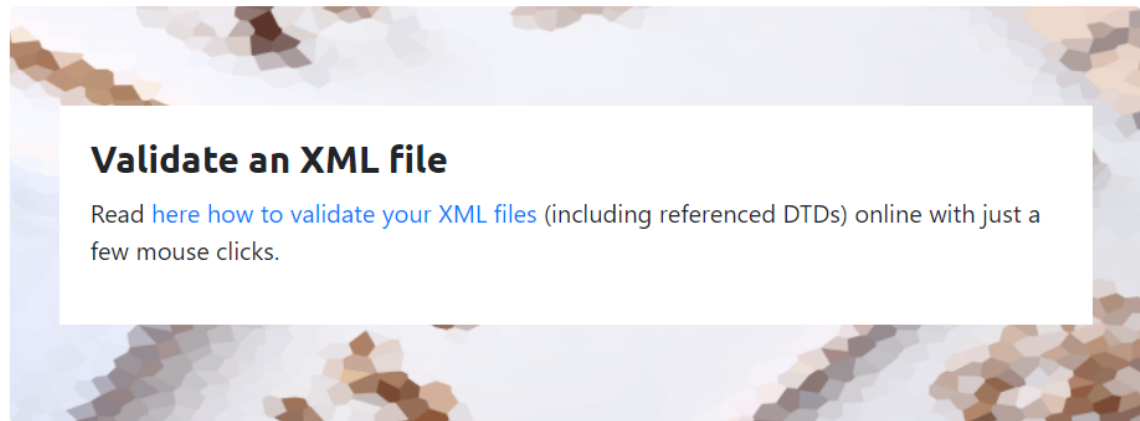
Objective	To check whether the values expect from the enumeration value is accepted in the type element or not.
Action	Except for the enumerated values, the values were entered in the type element, which was <Type>Double</Type>
Expected Result	The XML validator will be able to detect the error.
Actual Result	The XML validator detected the error while validating it.
Conclusion	Test was successful.

```

<GiftCards Quantity="4">
  <GiftCard cardID="GC1">
    <logo logoNum="01"></logo>
    <Type>Double</Type>
    <Cost>£40.00</Cost>
    <Validity>2022-06-13</Validity>
    <Category>online payments</Category>
    <NumberOfUsers>4</NumberOfUsers>
    <Discounts>2</Discounts>
    <Transferable>Not_Transferable</Transferable>
    <Description>
      <![CDATA[
        Its Small but I hope this makes the next few weeks just a little easier.
      ]]>
    </Description>
  </GiftCard>

```

Figure 10: Inserting invalid value except enumerated value



2 errors have been found!

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

Errors in the XML document:

- 26:32 cvc-enumeration-valid: Value 'Double' is not facet-valid with respect to enumeration '[Digital, Physical]'. It must be a value from the enumeration.
- 26:32 cvc-type.3.1.3: The value 'Double' of element 'Type' is not valid.

Figure 11: Evidence for the Test Case 5 which shows error

4.6 Test 6: To check the working of defined restriction in Validity element

Table 6: Test Case 6

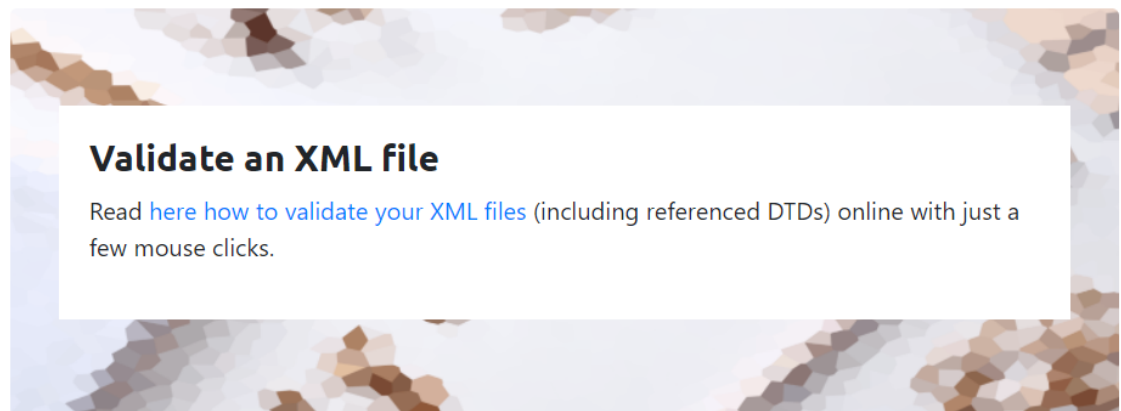
Objective	To see if a value other than the pattern value can be used in the Validity element.
Action	Except for the pattern values, all other values were inserted in the type element, which was
Expected Result	The XML validator will be able to detect the error.
Actual Result	The XML validator was able to detect the error while doing validation.
Conclusion	Test was successful.

```

<GiftCards Quantity="4">
  <GiftCard cardID="GC1">
    <logo logoNum="01"></logo>
    <Type>Physical</Type>
    <Cost>£40.00</Cost>
    <Validity>202-06-13</Validity>
    <Category>online payments</Category>
    <NumberOfUsers>4</NumberOfUsers>
    <Discounts>2</Discounts>
    <Transferable>Not_Transferable</Transferable>
    <Description>
      <![CDATA[
        Its Small but I hope this makes the next few weeks just a l
      ]]>
    </Description>
  </GiftCard>
</GiftCards>

```

Figure 12: Inserting invalid value expect pattern value



2 errors have been found!

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

Errors in the XML document:

- ✖ 28:43 cvc-pattern-valid: Value '202-06-13' is not facet-valid with respect to pattern '[0-9]{4}-[0-9]{2}-[0-9]{2}' for type 'ValidityRestriction'.
- ✖ 28:43 cvc-type.3.1.3: The value '202-06-13' of element 'Validity' is not valid.

Figure 13: Evidence for test case 3 which shows no errors

4.7 Test 7: To check the working of minOccurs of an element.

Table 7: Test Case 7

Objective	To check whether the elements can occur as per defined minOccurs value or not.
Action	Element Discounts is removed from one of the songs as its minOccurs value is zero
Expected Result	The XML, Schema and DTD will be valid and no error should be thrown in the process of validation.
Actual Result	The validator didn't detect any error on the process of validation and the validation was successfully.
Conclusion	Test was successful.

```
<GiftCards Quantity="4">
  <GiftCard cardID="GC1">
    <logo logoNum="01"></logo>
    <Type>Physical</Type>
    <Cost>£40.00</Cost>
    <Validity>2022-06-13</Validity>
    <Category>online payments</Category>
    <NumberOfUsers>4</NumberOfUsers>
    <Transferable>Not_Transferable</Transferable>
    <Description>
      <![CDATA[
        Its Small but I hope this makes the next few weeks just a
      ]]>
    </Description>
  </GiftCard>

  <GiftCard cardID="GC2">
    <logo logoNum="02"></logo>
    <Type>Digital</Type>
    <Cost>£25.00</Cost>
    <Validity>2022-06-13</Validity>
    <Category>shopping mall</Category>
    <NumberOfUsers>5</NumberOfUsers>
    <Discounts>5</Discounts>
    <Transferable>Not_Transferable</Transferable>
    <Description>
      <![CDATA[
        Its Small but I hope this makes the next few weeks just a
      ]]>
    </Description>
  </GiftCard>
</GiftCards>
```


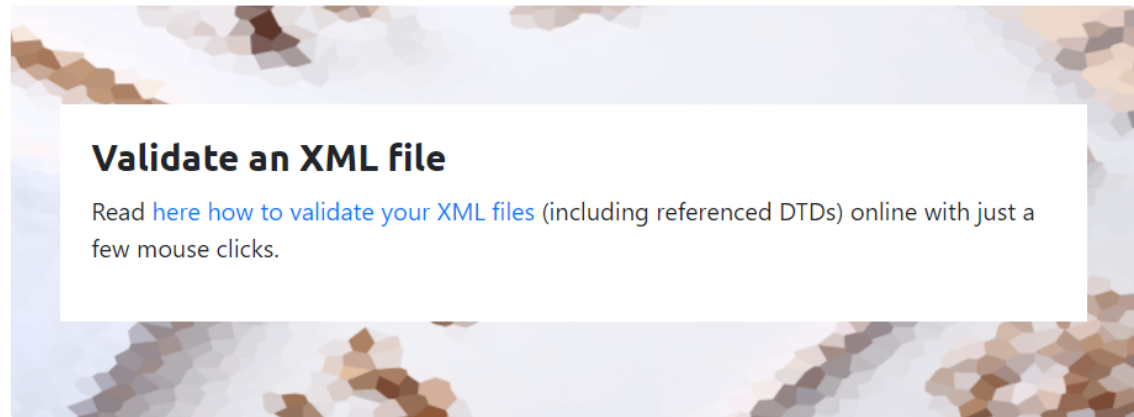



Figure 14: Removing discount element from a GiftCard element



No errors were found

The following files have been uploaded so far:

[XML document:](#) 

[catalog_20048472.xsd](#) 

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

Figure 15: Evidence of test case 7 with no errors

4.8 Test 8: Checking whether the working of positiveInteger data type in Quantity Attributes.

Table 8: Test Case 8

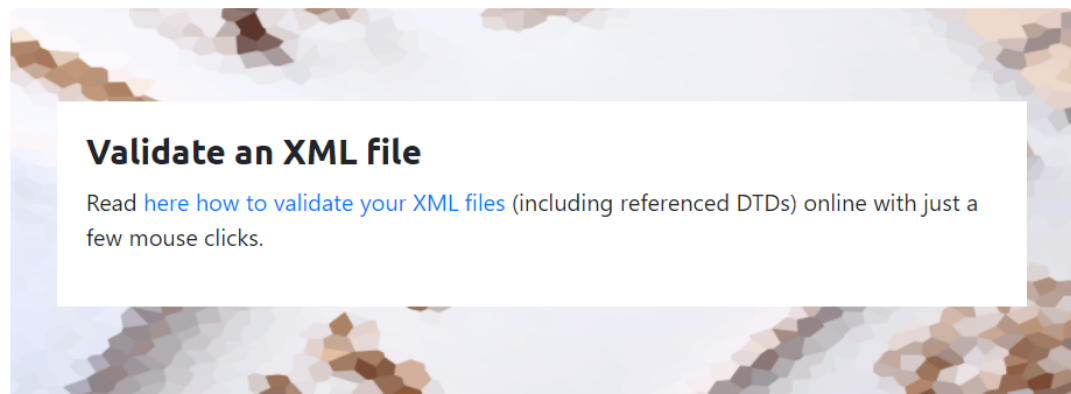
Objective	To check whether the declares positiveInteger data type in quantity attributes works or not.
Action	Negative integer value is entered in the attribute Quantity. Which is: Quantity= -4
Expected Result	The XML, Schema and DTD will not be valid and error will be shown in the process of validation.
Actual Result	The validator detected the error on the process of validation and the files was not valid.
Conclusion	Test was successful.


```

<GiftCards Quantity="-4">
  <GiftCard cardID="GC1">
    <logo logoNum="01"></logo>
    <Type>Physical</Type>
    <Cost>£40.00</Cost>
    <Validity>2022-06-13</Validity>
    <Category>online payments</Category>
    <NumberOfUsers>4</NumberOfUsers>
    <Transferable>Not_Transferable</Transferable>
    <Description>
      <![CDATA[
        Its Small but I hope this makes the next few weeks just a lit
      ]]>
    </Description>
  </GiftCard>

```

Figure 16: Inserting invalid (negative integer) value in quantity attribute



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:

- 22: 30 cvc-attribute.3: The value '-4' of attribute 'Quantity' on element 'GiftCards' is not valid with respect to its type, 'positiveInteger'.
- 22: 30 cvc-minInclusive-valid: Value '-4' is not facet-valid with respect to minInclusive '1' for type 'positiveInteger'.

Figure 17: Evidence for test case 8 with errors

4.9 Test 9: To check the working of the data type ID in the CardID attribute.

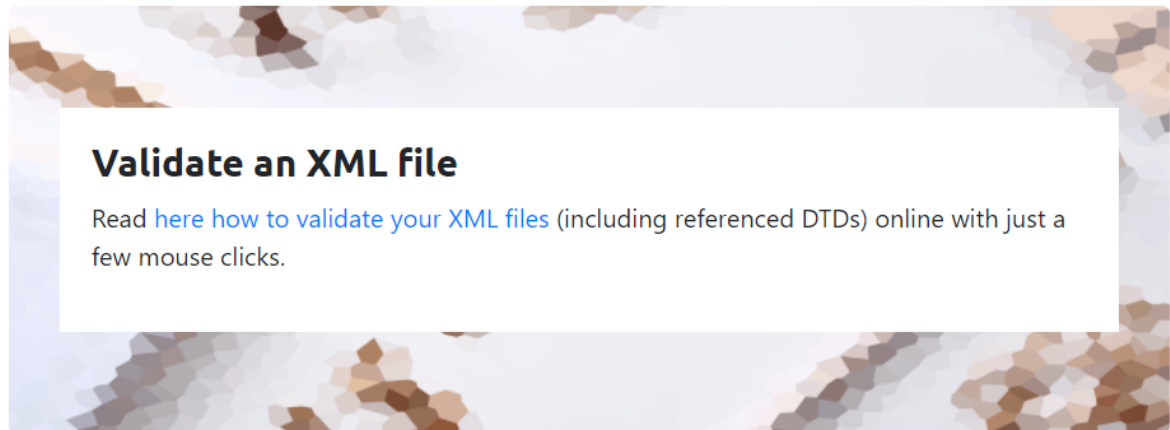
Table 9: Test Case 9



Objective	To see whether the CardID attribute's stated ID data type works or not.
Action	Same value of the CardID is provided twice in the attribute CardID which should be unique.
Expected Result	The XML, Schema and DTD won't be valid and error will be thrown in the process of validation.
Actual Result	The validator detected the error while validating the file and the files was not validated.
Conclusion	Test was successful.

```
<GiftCards Quantity="4">
  <GiftCard cardID="GC1">
    <logo logoNum="01"></logo>
    <Type>Physical</Type>
    <Cost>£40.00</Cost>
    <Validity>2022-06-13</Validity>
    <Category>online payments</Category>
    <NumberOfUsers>4</NumberOfUsers>
    <Transferable>Not_Transferable</Transferable>
    <Description>
      <![CDATA[
        Its Small but I hope this makes the next few weeks just a lit
      ]]>
    </Description>
  </GiftCard>

  <GiftCard cardID="GC1">
    <logo logoNum="02"></logo>
    <Type>Digital</Type>
    <Cost>£25.00</Cost>
    <Validity>2022-06-13</Validity>
    <Category>shopping mall</Category>
    <NumberOfUsers>5</NumberOfUsers>
    <Discounts>5</Discounts>
    <Transferable>Not_Transferable</Transferable>
    <Description>
      <![CDATA[
        Its Small but I hope this makes the next few weeks just a lit
      ]]>
    </Description>
  </GiftCard>
</GiftCards>
```

Figure 18: Inserting the Same CardID twice

**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:

- ✖ 37:20 The content of element type "GiftCard" must match "(logo,Type,Cost,Validity,Category,NumberOfUsers,Discounts,Transferable,Description)".
- ✖ 39:32 Attribute value "GC1" of type ID must be unique within the document.

Figure 19: Evidence for Test Case 9 with errors

5. Difference between Schemas and DTDs

An XML Schema is a document that outlines how an XML document must be formatted in order to comply with the structure required by a system or service that uses the XML Schema (Adams, 2021).

A Document Type Definition (DTD) provides a document's tree structure as well as some information about its data. It is a set of markup assertions that define a type of document for the SGML family, such as GML, SGML, HTML, and XML.

The difference between them are:

Schemas	DTD
XML Schema helps in the creation of XML documents by defining the rules for attributes and elements in an XML document.	DTD describes the attributes and structure of an XML document and languages.
It supports variety of data types similar to programming language.	In DTD everything is treated as text.
There are 44 types of data of any object that are supported in Schemas.	There are only 10 type of data of any object that are supported in DTDs.
Creating relationship among elements is possible.	This is not possible in DTD without invalidating existing documents.
It is extensible.	It is not extensible.
XSD are written in XML.	They are derived from SGML syntax.
It is simple to learn because you don't need to learn new language.	It is not simple to learn.
It supports namespace.	It doesn't support namespace.
It defines order for child elements.	It doesn't define for child elements.
They need to be created from the scratch for an every new XML document.	DTD are established and examples of common objects defined in a DTD abound on the internet- freely for re-use.

6. How the Coursework was developed?

The question was first learned and thoroughly studied. Following an interview with the store manager, the job was to design a model of a system for a gift card store. In order to grasp the scope of the project, it was required to conduct research and add additional material that was more realistic and not included in the scenario. All of the requirements and information needed to complete the project were gathered, and the development was proceeded forward to include the production of an XML document, as well as a DTD and schema file to validate all of the data and information contained inside the XML documents. Things like defining the Occurrence of element, restricting the data in the element were considered and were investigated effectively with the help of the teaching module and were used in the Schema file. The tree diagram was built with the help of Draw.io after the XML file was successfully validated. Draw.io creates the XML documents as tree elements and plays a vital role in describing any XML document quickly. Following the production of the XML and Schema files, the DTD (Document Type Definition) for the XML document was also created. The structure, as well as the elements and attributes that must be valid in XML documents, were stated clearly in the DTD so that there would be no confusion about the various data kinds and information in the XML file. DTD is also used to validate and authenticate data coming from the outside world, as well as to double-check our own data. After finishing all of these files and diagrams, the CSS for the XML documents was created, which aids in providing a certain style or look to the gift shops. After all of the development work was completed, all of the documentation was completed.

The tools used for creating this project is given below:

- Draw.io

It is a flowchart solution which is considered to help developers, network admins, IT analysts and designers use drag and drop functionality in order to create and distribute diagrams. It enables professionals to add toggle layers with customizable URLs and align texts within the shapes. (ComputerHope, 2020)



Figure 20: Draw.io

- Microsoft Word

Microsoft Word is a word processing program that was first released in 1983 by Microsoft. It is the most widely used word processing program. It is used to create professional-looking papers, letters, reports, resumes, and other documents, as well as to edit or change existing ones. The.docx extension refers to a document saved in Microsoft Word (Rodrigo, 2021).



Figure 21: Ms-word

- Visual Studio Code

Microsoft's Visual Studio Code (often known as VS Code) is a free open source text editor. For Windows, Linux, and macOS, VS Code is available. VS Code includes numerous significant features that have made it one of the most popular development environment tools in recent years, despite its modest weight (Mustafeez, 2021).



Figure 22: Visual Studio Code

- XMLvalidation.com

XML validation is a method that checks an XML document for syntax errors in order to ensure that it is well-written and follows the standard rules using either DTD or schema. To check for validation document type definitions, two schemas are used, and it is examined for correctness, so the data and structure of the XML document are not taken into account. If the data is the location and first name, for example, validation works if the values are provided without being empty (Pedarkar, 2021).



Figure 23: XML Validation

7. Critical Analysis and Conclusion

For each of us, the endeavor was both enlightening and fascinating. The goal of the course was to design and build a model for a system that could be used as an XML developer on the web. Various things that were required during the development process were offered in the scenario, as well as the request to add more relevant information that would help the user grasp the document and make it more practical. To begin, an XML document was developed, followed by the creation of a schema, also known as an XSD, to check and authenticate the information in the XML file. The development of an XML file appears simple at first, but it was a challenging but enjoyable aspect of the training.

According to the scenario that was presented to us, the model system should contain a minimum of 15 data, 5 characteristics, and the mandatory usage of 5 optional fields for the element. It was difficult to conduct research that was not presented to us as part of the scenario. The XML file was simple to create, but the Schema file was more difficult to create, as several problems and errors were discovered early on while validating it. The development of the Schema file was a difficult but interesting topic to grasp, understand, and correct faults. All of these issues were effectively resolved with the assistance of our Module Leader, Mr. Pratik Panta Sir. Various studies were conducted on a variety of websites, publications, papers, research materials, and discussion boards. The tree diagram for the XML file was constructed after the successful production of the XML document, also known as an instance document. The DTD was also created to validate the XML document. It was also a new topic, similar to Schema, and it was fun to learn and create. The DTD was used to specify the XML Document's structure, as well as the legal components and attributes.

Following the successful development and validation of the XML documents, CSS was developed to provide the XML documents with a decent organization, layout, and style. We were comfortable with CSS when it came to HTML, but now we had to create CSS for an XML file, which proved to be pretty difficult. All of the challenges were overcome with a little amount of hard work, investigation, and, to a

large extent, the module teacher's help. It was quite perplexing and uncertain how to use CSS to style an XML document. As a matter of fact, those problems and troubles faced during the creation of this project was very fruitful and informative about various components, modules in XML, CSS, DTD as well as the Schema of the project. In fact, the problems and difficulties encountered during the development of this project were highly beneficial and educational regarding numerous components, modules in XML, CSS, DTD, and the project's Schema.

Conclusion

Following a discussion with the store manager, the coursework required preparing an XML document to model the Gift Card Store's system. The main goal of this project was to show and preserve thorough information and statistics about the numerous gift cards accessible in the gift card store. The XML, DTD, and Schema files were developed along with the CSS file to provide the XML file style and were used to test, validate, and format the XML document. The scenario was given to us, and we gathered a lot of information from the web, papers, and other sources to incorporate into the project to make it more valid and practical. The project already has valid Schema, DTDs, and CSS for the XML documents, as can be seen. This project has required a great deal of expertise and understanding of Schema, DTS, and CSS. The coursework has been really helpful in learning more about CSS, particularly with regard to the use of various selectors. All of the information and expertise we obtained from this project will be extremely useful in our future study of XML, DTD, Schema, CSS, and web design. Furthermore, all of the contents of those topics will be beneficial and useful in furthering one's career as an IT expert and web designer.

8. References

- Adams, C., 2021. *Modern Analyst*. [Online]
Available at: <https://www.modernanalyst.com/Careers/InterviewQuestions/tabid/128/ID/1363/What-is-an-XML-Schema-and-what-is-its-purpose.aspx>
[Accessed 02 May 2022].
- ComputerHope, 2020. *ComputerHope*. [Online]
Available at: <https://www.computerhope.com/jargon/d/drawio.htm>
[Accessed 04 May 2022].
- Kenlon, S., 2021. *Opensource.com*. [Online]
Available at: <https://opensource.com/article/21/7/what-xml>
[Accessed 27 April 2022].
- Lawton, G., 2021. *techtarget*. [Online]
Available at: <https://www.techtarget.com/whatis/definition/XSD-XML-Schema-Definition>
[Accessed 27 April 2022].
- Mustafeez, A. Z., 2021. *Educative*. [Online]
Available at: <https://www.educative.io/edpresso/what-is-visual-studio-code>
[Accessed 04 May 2022].
- Parvez, F., 2021. *Great Learning*. [Online]
Available at: <https://www.mygreatlearning.com/blog/css-tutorial/>
[Accessed 02 May 2022].
- Pedamkar, P., 2021. *EDUCBA*. [Online]
Available at: <https://www.educba.com/xml-validation/>
[Accessed 04 May 2022].

Refsnes, J. E., 2021. *Introduction to DTD*. [Online]
Available at: <https://www.xmlfiles.com/dtd/>
[Accessed 27 April 2022].

Rodrigo, A., 2021. *envato tuts+*. [Online]
Available at: <https://business.tutsplus.com/tutorials/what-is-microsoft-word-definition--cms-34990>
[Accessed 04 May 2022].

Singh, C., 2021. *Liquid Technologies*. [Online]
Available at: <https://beginnersbook.com/2018/11/xml-schema/>
[Accessed 27 April 2022].

W3Schools, 2021. *W3schools*. [Online]
Available at: https://www.w3schools.com/xml/schema_intro.asp
[Accessed 27 April 2022].

Yin, Y., 2012. Using Tree Diagrams as an Assessment Tool in Statistics Education. *Educational Assessment*, 17(1), pp. 22-49.