Progressive Education Society's

**Modern College of Engineering, Pune**

**MCA Department**

**A.Y.2024-25**

**(410902) Web Technologies Lab**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Class: SY-MCA Shift / Div : A Batch: S3 Roll Number : 51062

Name: Laxman Shinde Assignment No : 2 Date of Implementation : 10/08/24

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Q.1. Create XML file to describe library book details and external DTD to it.**

**Code :**

**Q1.xml**

<?xml version="1.0"?>

<!DOCTYPE employees SYSTEM "employee.dtd">

<employees>

    <employee>

        <employee\_id>E001</employee\_id>

        <first\_name>John</first\_name>

        <last\_name>Doe</last\_name>

        <department>Engineering</department>

        <position>Software Engineer</position>

        <salary>75000</salary>

    </employee>

    <employee>

        <employee\_id>E002</employee\_id>

        <first\_name>Jane</first\_name>

        <last\_name>Smith</last\_name>

        <department>Marketing</department>

        <position>Marketing Manager</position>

        <salary>85000</salary>

    </employee>

</employees>

**Employee.dtd**

<!ELEMENT employees (employee+)>

<!ELEMENT employee (employee\_id, first\_name, last\_name, department, position, salary)>

<!ELEMENT employee\_id (#PCDATA)>

<!ELEMENT first\_name (#PCDATA)>

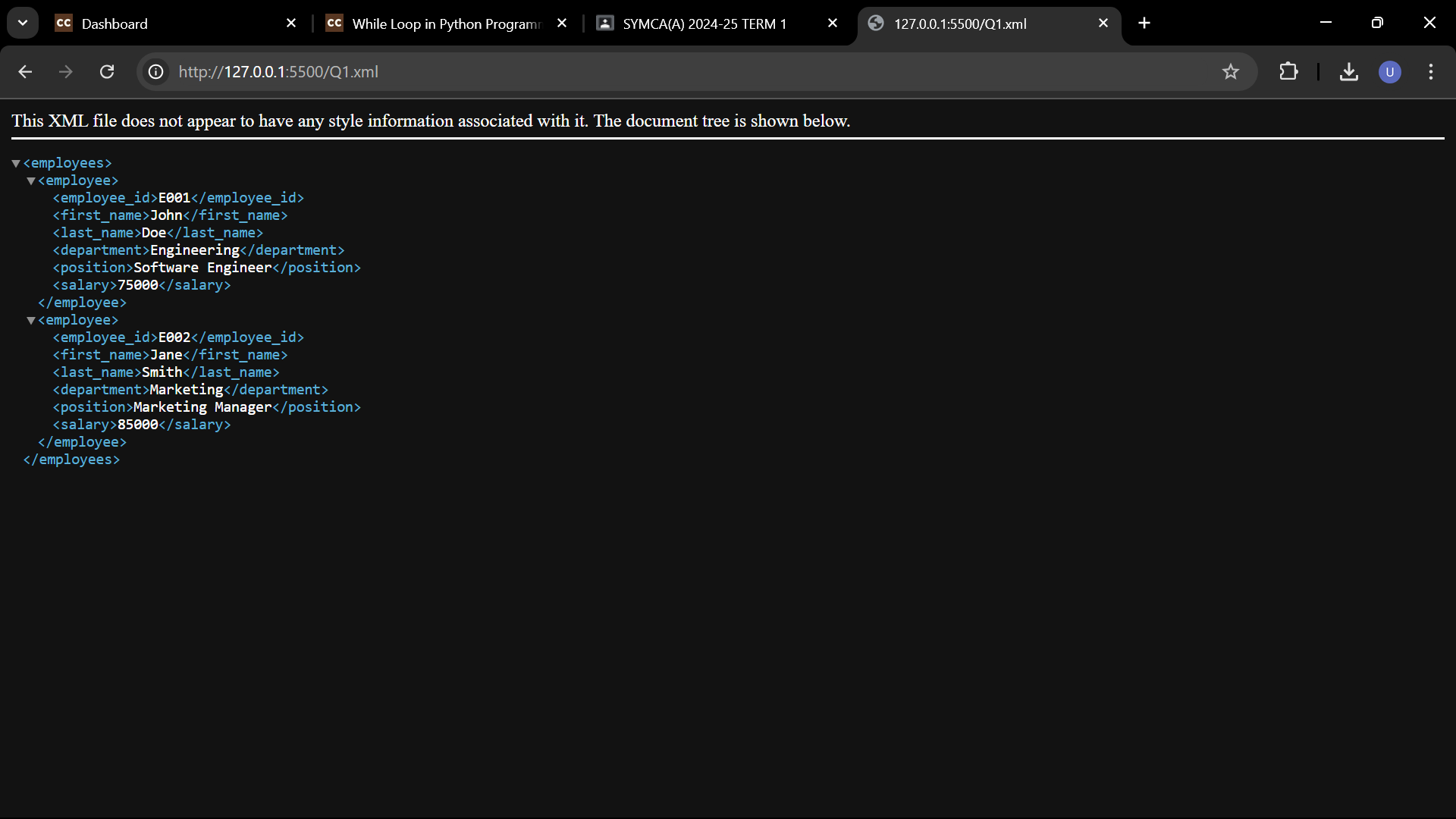
<!ELEMENT last\_name (#PCDATA)>

<!ELEMENT department (#PCDATA)>

<!ELEMENT position (#PCDATA)>

<!ELEMENT salary (#PCDATA)>

**Output :**

****

Q.2. Design a DTD for an XML database of restaurants. This will check that the XML file is well-formed and that it validates against the DTD you have given. If the document is valid, you get no response; if invalid, you get an error message.

Code :

Q2.xml

<?xml version="1.0"?>

<!DOCTYPE restaurantDatabase SYSTEM "restaurant.dtd">

<restaurantDatabase>

    <restaurant id="1" >

        <name>Restaurant One</name>

        <address>123 Main Street</address>

        <cuisine>Italian</cuisine>

        <rating>4.5</rating>

        <phone>(123) 456-7890</phone>

        <description>A cozy Italian restaurant serving pasta and pizza.</description>

    </restaurant>

    <restaurant id="2">

        <name>Restaurant Two</name>

        <address>456 Oak Avenue</address>

        <cuisine>Mexican</cuisine>

        <rating>3.8</rating>

        <phone>(987) 654-3210</phone>

    </restaurant>

</restaurantDatabase>

Restaurant.dtd

<!ELEMENT restaurantDatabase (restaurant+)>

<!ELEMENT restaurant (name, address, cuisine, rating, phone, description?)>

<!ELEMENT name (#PCDATA)>

<!ELEMENT address (#PCDATA)>

<!ELEMENT cuisine (#PCDATA)>

<!ELEMENT rating (#PCDATA)>

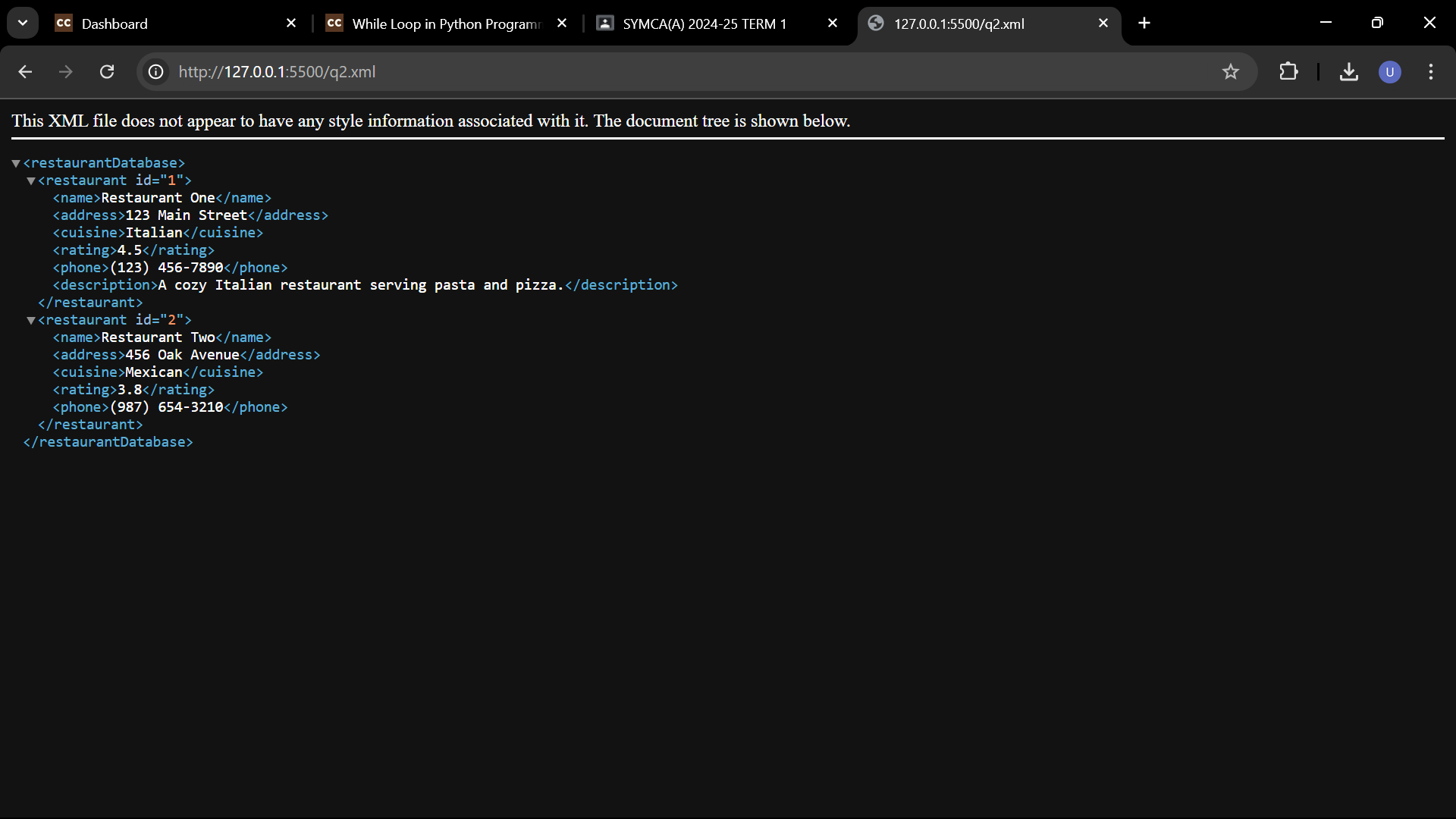
<!ELEMENT phone (#PCDATA)>

<!ELEMENT description (#PCDATA)>

<!ATTLIST restaurant id CDATA #REQUIRED>

<!ATTLIST restaurant website CDATA #IMPLIED>

Output :



**Q.3. Create XML file for a student database. Next create the document type**

**definition for the xml structure and finally create the schema document for the**

**xml document.**

**Code :**

Q3.xml

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

<students>

    <student>

        <id>1</id>

        <name>John Doe</name>

        <age>20</age>

        <major>Computer Science</major>

        <gpa>3.7</gpa>

    </student>

    <student>

        <id>2</id>

        <name>Jane Smith</name>

        <age>21</age>

        <major>Mathematics</major>

        <gpa>3.9</gpa>

    </student>

</students>

Student.dtd

<!ELEMENT students (student+)>

<!ELEMENT student (id, name, age, major, gpa)>

<!ELEMENT id (#PCDATA)>

<!ELEMENT name (#PCDATA)>

<!ELEMENT age (#PCDATA)>

<!ELEMENT major (#PCDATA)>

<!ELEMENT gpa (#PCDATA)>

Student.xsd

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

    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

        <xs:element name="students">

            <xs:complexType>

                <xs:sequence>

                    <xs:element name="student" type="studentType" minOccurs="1" maxOccurs="unbounded"/>

                </xs:sequence>

            </xs:complexType>

        </xs:element>

        <xs:complexType name="studentType">

            <xs:sequence>

                <xs:element name="id" type="xs:int"/>

                <xs:element name="name" type="xs:string"/>

                <xs:element name="age" type="xs:int"/>

                <xs:element name="major" type="xs:string"/>

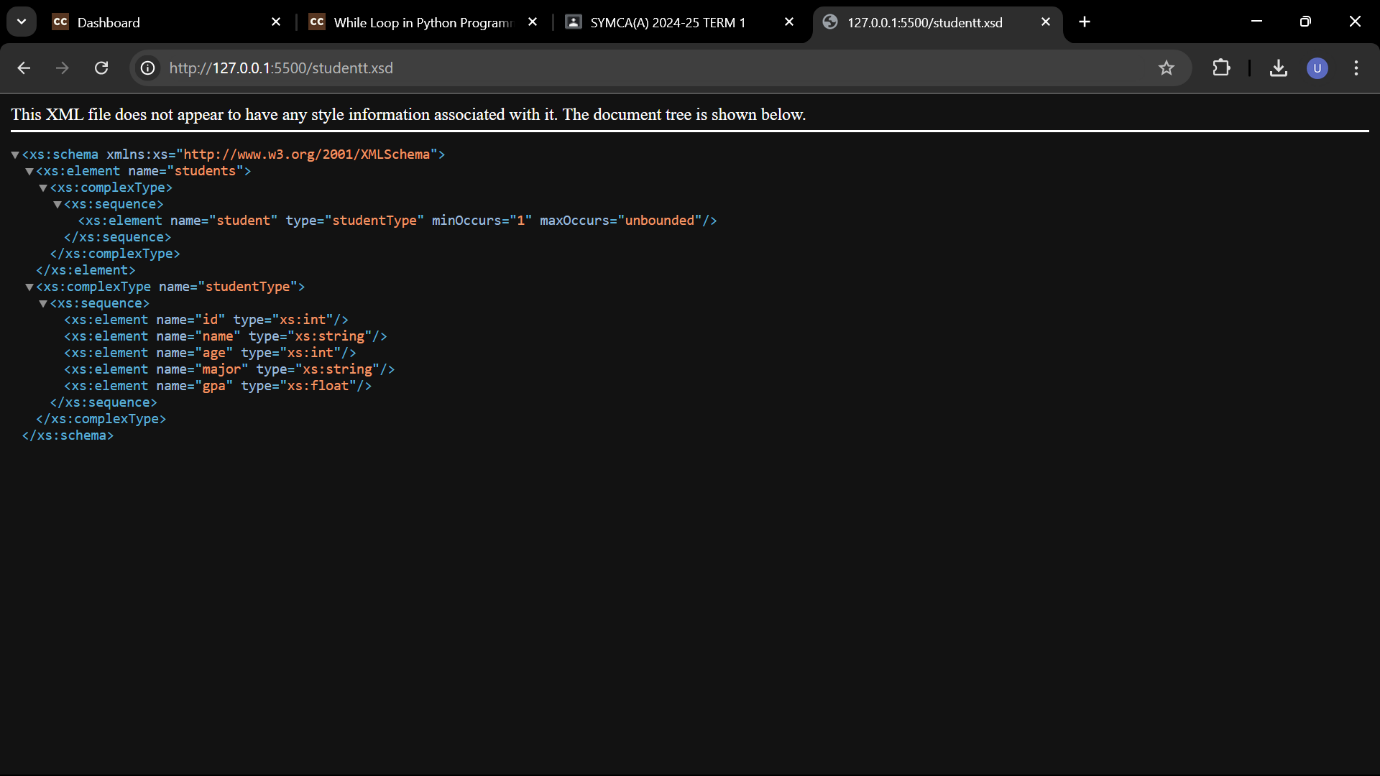
                <xs:element name="gpa" type="xs:float"/>

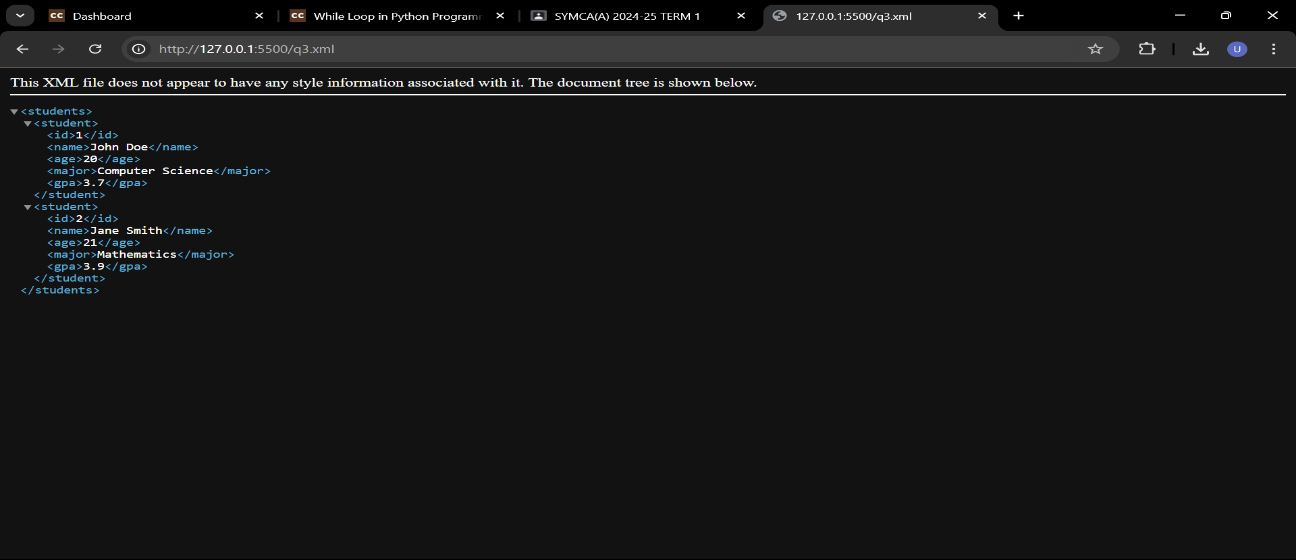
            </xs:sequence>

        </xs:complexType>

    </xs:schema>

Output :





Q4. Open the XML file “amazon-extended.xml”. The file contains 2 types of products: books and software. Create an XSL transformation document for presenting all the products in the XML file. Transform the XML using the XSLT you developed. Save the developed XSL transformation file as: “all-products.xsl”. View the resulting HTML file in a browser window.

Product.xml:

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

<products>

    <product>

        <type>Book</type>

        <title>Sample Book</title>

        <author>John Doe</author>

        <price>19.99</price>

    </product>

    <product>

        <type>Software</type>

        <title>Sample Software</title>

        <company>XYZ Inc.</company>

        <price>49.99</price>

    </product>

   </products>

Product.xsl

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

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

    <xsl:template match="/">

        <html>

            <head>

                <title>Product List</title>

            </head>

            <body>

                <h1>Product List</h1>

                <table border="1">

                    <tr>

                        <th>Product Type</th>

                        <th>Title</th>

                        <th>Author/Company</th>

                        <th>Price</th>

                    </tr>

                    <xsl:apply-templates select="//product"/>

                </table>

            </body>

        </html>

    </xsl:template>

    <xsl:template match="product">

        <tr>

            <td><xsl:value-of select="type"/></td>

            <td><xsl:value-of select="title"/></td>

            <td><xsl:value-of select="author|company"/></td>

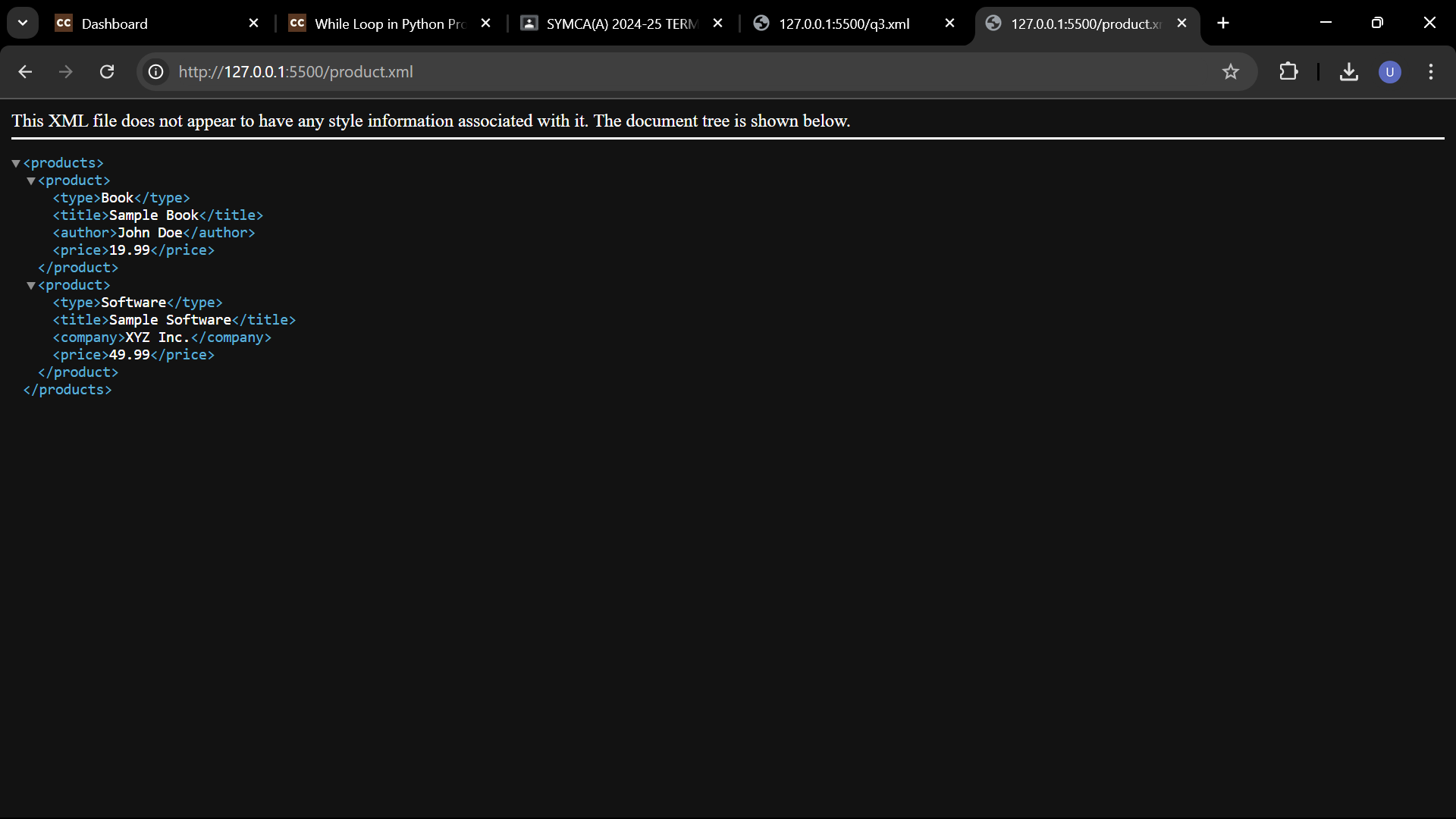
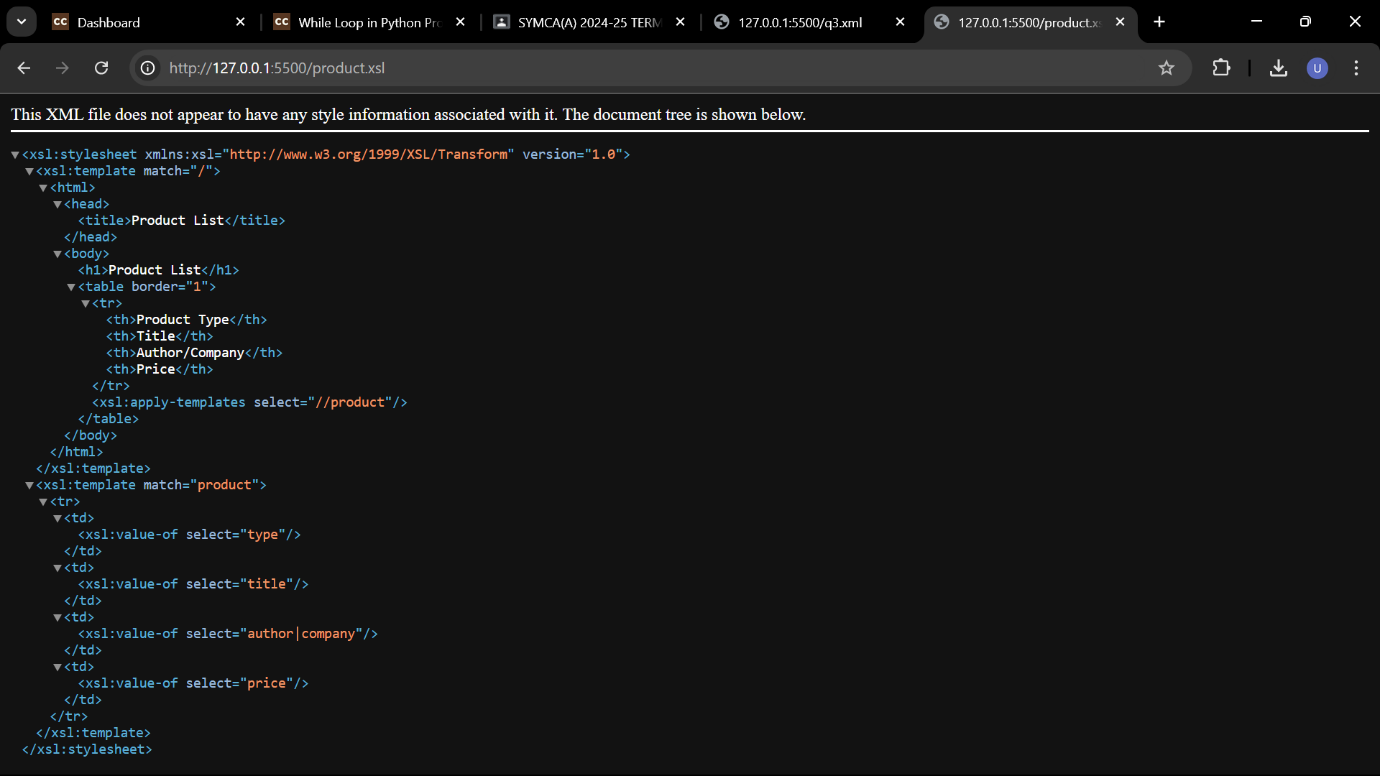
            <td><xsl:value-of select="price"/></td>

        </tr>

    </xsl:template>

</xsl:stylesheet>

Output :



**Q.5. Create an XML document with the following sample real estate data**

**a. Root element real-estate will contain a sequence of sub-elements agencies, owners, properties and flats, all with an empty content**

**b. Ensure well-formedness**

**c. Create an internal DTD to ensure its validity and then try to break it.**

**d. Move the DTD to an external file and validate the XML document again**

**Modify the DTD schema: Using parameter entities and conditional sections, declare 2 modes for the content of owners: ‒ Verbose • Everything ‒ Standard • Everything except the description, Activate the standard mode and adjust the XML document appropriately.**

Internal.xml:

<?xml version="1.0"?>

<!DOCTYPE real-estate [

<!ELEMENT real-estate (agencies, owners, properties, flats)>

<!ELEMENT agencies EMPTY>

<!ELEMENT owners (#PCDATA)>

<!ELEMENT properties EMPTY>

<!ELEMENT flats EMPTY>

<!-- Define parameter entities for owner content modes -->

<!ENTITY % owner-content "INCLUDE">

<!ENTITY % standard-mode "IGNORE">

<!-- Conditional sections for owner content -->

<!ENTITY % owner-content-mode SYSTEM "owner-content-mode.dtd">

%owner-content-mode;

]>

<real-estate>

<agencies></agencies>

<owners>

<owner>

<name>John Doe</name>

<!-- In standard mode, description is excluded -->

<!-- <description>Real estate owner</description> -->

</owner>

<owner>

<name>Jane Smith</name>

<!-- <description>Another owner</description> -->

</owner>

</owners>

<properties></properties>

<flats></flats>

</real-estate>

Real-estate.dtd:

<!ELEMENT real-estate (agencies, owners, properties, flats)>

<!ELEMENT agencies EMPTY>

<!ELEMENT owners (#PCDATA)>

<!ELEMENT properties EMPTY>

<!ELEMENT flats EMPTY>

<!-- Define parameter entities for owner content modes -->

<!ENTITY % owner-content "INCLUDE">

<!ENTITY % standard-mode "IGNORE">

<!-- Conditional sections for owner content -->

<![%owner-content;[

  <!ELEMENT owner (name, description?)>

  <!ELEMENT name (#PCDATA)>

  <!ELEMENT description (#PCDATA)>

]]>

<![%standard-mode;[

  <!ELEMENT owner (name)>

  <!ELEMENT name (#PCDATA)>

]]>

Q5.xml:

<?xml version="1.0"?>

<!DOCTYPE real-estate SYSTEM "real-estate.dtd">

<real-estate>

  <agencies></agencies>

  <owners>

    <owner>

      <name>John Doe</name>

      <!-- In standard mode, description is excluded -->

      <!-- <description>Real estate owner</description> -->

    </owner>

    <owner>

      <name>Jane Smith</name>

      <!-- <description>Another owner</description> -->

    </owner>

  </owners>

  <properties></properties>

  <flats></flats>

</real-estate>

Output :

