Program:	**IWD1-4**
Course:	**INFO-5064**
Professor:	**Tony Haworth and Stacey Kazmir**
Project:	**#2 – Schools Application**
Due Date:	**Thursday, April 9, 2020 by 11:59 pm**
Last Update:	**Tuesday, March 17th, 2020**
To be completed individually by each student!	

Description

Create a web application that uses an XML document as a data source and uses XSLT and XPath to generate formatted (html) reports based on the XML data.

Purpose

To apply the following XML technologies within a web application:

1. Transforming/formatting XML data using XSLT
2. Retrieving specific subsets of an XML document using XPath

Overview

For this project you are provided with the XML document ***schools.xml*** which contains data about the schools in London, Ontario. On successfully completing this project, you will have an interactive webpage that will allow the user to view information about any type of school they select such as “elementary” or “secondary”. The user will also be able to search for a school by name.

Requirements

Create the following application components:

1. An parameterized XSLT style sheet is designed to format/transform the given XML file with the following requirements:
    a. Uses an input parameter to determine which type of school (elementary, secondary, etc.) the report should be for
    b. Uses a second input parameter to select one or more schools (of the type selected) using the name, or partial name, of a school
    c. Output is a report in HTML format and includes:
        i. A line of text reporting the number of schools in the report
        ii. A table with column labels to display the school information for each school in the report in alphabetical order based on each school’s *short-name* attribute and including the following fields/columns:
            1. The full *Name* of the school with a hyperlink to the school’s *Website*
            2. The school’s *Board*
            3. The school’s *Address*
2. A website (web page) that does all of the following:
    a. Displays an appropriate title

b. Prompts the user to choose a type of school using a select field that is populated from the file *schools.xml*. The school types are the names of all elements that are children of the root element "London-Schools". These values can be obtained with some JavaScript using either XPath or XSLT. However, the select field should also include the value "All Schools" which should result in schools of any type being included in the output.
c. Uses JavaScript code to execute the XSLT style sheet from part 1 to generate and display the output of the style sheet based on the following inputs:
    iii. The XML data file (*schools.xml* provided)
    iv. Your XSLT style sheet
    v. The school type (or "All Schools") selected by the user
    vi. A school name (or partial name) entered by the user*

    ** This last input is optional for the user. If nothing is entered then this input can be ignored.*

Note that you are not required to publish your website for evaluation. You are only required to submit the files required for the website.

3. You should also conform to these additional requirements:
    a. Do not modify the XML document *schools.xml*.
    b. Your application should work with the following web browsers: Edge, Chrome, and Firefox

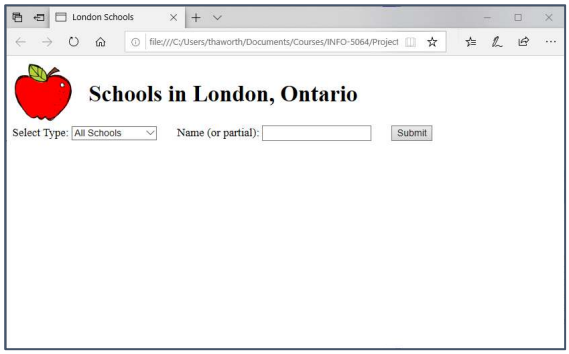
Tips & Hints

1. Here's a suggested series of steps for working towards a full solution:
    a. To start with, create an XSLT style sheet that will return data for all schools in the required format. You can test this without having the website and JavaScript code completed.
    b. Now modify the style sheet to incorporate style sheet parameters for the selected type and name. You can assign default values to these parameters, like "All Schools" and "Fanshawe", so that it should work even if no values are otherwise assigned to the parameters when the style sheet is executed.
    c. Once you have the above version working, try creating a webpage that will allow the user to select a school type (you can temporarily hardcode the types into the select field in the web page) and a school name. Add a submit button and some JavaScript code to execute the style sheet and display the style sheet output. The webpage should assign the school type and school name selected by the user to the style sheet's parameters when executing the style sheet. You will need some conditional logic in your style sheet to retrieve the school elements differently depending on whether the type selected is "All Schools" or another value and also depending on whether the user entered a school name.
    d. Finally, add a JavaScript function to your webpage that will execute when the web page loads to populate the "school types" select field with all the school types in *schools.xml* (as well as the value "All Schools"). Again, this may be done using either XPath or XSLT. The following screen shot shows the school types in the xml file.

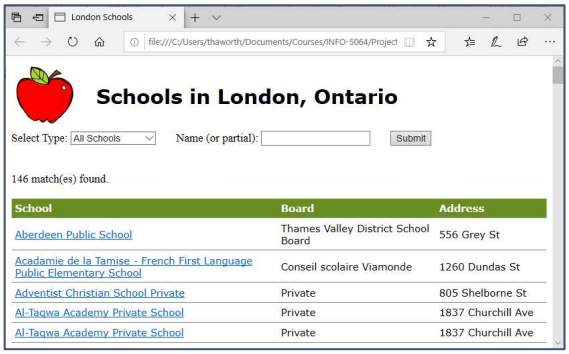
```
<London-Schools>
   <Adult-Education>...</Adult-Education>
   <Elementary>...</Elementary>
   <Post-Secondary>...</Post-Secondary>
   <Pre-School>...</Pre-School>
   <Secondary>...</Secondary>
</London-Schools>
```

Sample Output

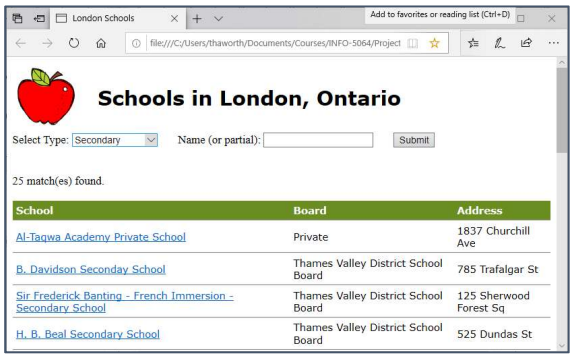
The sample output shown is only a guideline. Style yours as you wish. The apple graphic shown is not required. However, your page should function as illustrated here.



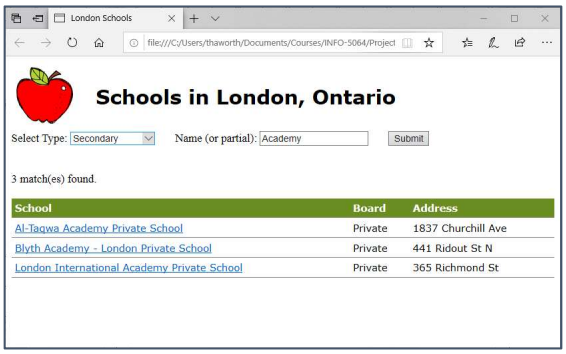
Initial page view



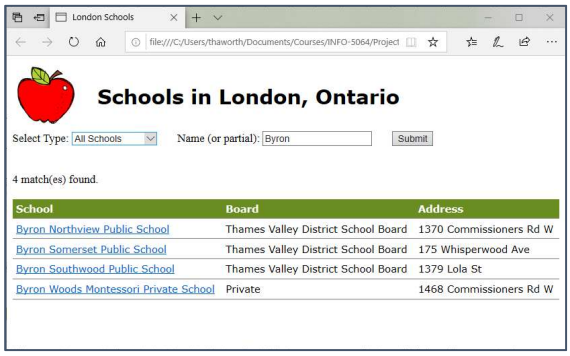
All schools (first few rows only)



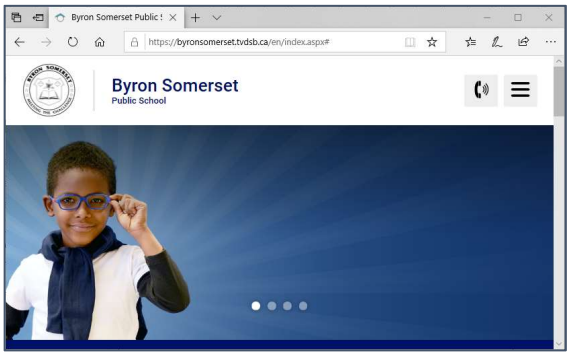
Secondary schools (first few rows only)



Secondary schools with “Academy” in the name



All schools with “Byron” in the name



After clicking on Byron Somerset Public School

Grading Scheme

1. Projects will be evaluated as follows:

Description	Marks
XSLT document submitted is well-formed and valid XSLT	5
The XSLT document creates html output from the XML data which:	
1. is a well-formed HTML-format document	5
2. displays information for schools matching the user-selected type and (optionally) a name or partial name	5
3. displays the *Name*, *Board* and *Address* values for each school displayed in a tabular format with column labels	5
4. incorporates a hyperlink to the school website on the *Name* field using the url provided in the *Website* field	5
5. displays the schools sorted in alphabetical order using the *short-name* field for each school	5
The website submitted:	
1. fills the *Select Type* select field with all school types in *schools.xml* using either XPath or XSLT, as well as the value "All Schools"	5
2. works without errors when provided with reasonable inputs, displays an appropriate title, and allows the user to select a type and (if desired) a school name or partial name	5
3. uses the XLT style sheet, the XML document, the *type* selected and any school name or partial name inputted by the user to generate HTML output containing the relevant school information and including a count of the number of schools returned by the style sheet	5
4. neatly displays the output generated by the style sheet the browser	5
Deduction if modifications were made to the XML file to create a working solution	-5
TOTAL	50

2. The following grade deductions will be applied for submitting a project after the submission deadline:

1 day late (< 24 hours late)	10% deduction
2 days late (< 48 hours late)	20% deduction
3 days late (< 72 hours late)	30% deduction
4 days late (< 96 hours late)	40% deduction
5 days late (< 120 hours late)	50% deduction
≥ 5 days late	Grade of 0%

Note that the FOL submission folders report "days late" differently. For example, it reports 1 day late when a submission is made <u>at least</u> 24 hours late but less than 48 hours late.

Submit

Via the link on the *Project 2* dropbox in FanshaweOnline an archive (ZIP or similar) file containing:

1. Your XSLT style sheet
2. Your web page
3. Any other files required to deploy your website

Submit your project on time!

Code submissions to the FOL drop box must be made on time! Grades for late projects will include deductions as explained above under Grading Scheme.

Submit your own work and *Keep it to yourself*!

It is considered cheating to submit work done by another student or from another source. Helping another student cheat by sharing your work with them is also not tolerated. Students are encouraged to share ideas and to work together on practice exercises, but any code or documentation prepared for a project must be done by the individual student. Penalties for cheating or helping another student cheat may include being assigned zero on the project with even more severe penalties if you are caught cheating more than once. Just submit your own work and benefit from having made the effort on your own.

Project Corrections

If any corrections or changes are necessary they will be posted to the course web site and you will be notified of any changes in class. It is your responsibility to check the site periodically for changes to the project. Additional resources relating to the project may also be posted.