CSE 445/598 Project 4 (Assignment 5) (50 Points)
Fall 2023

Project Due Date: Saturday, Oct. 28, 2023, by 11:59pm (Arizona Time), plus a one-day grace period.

Introduction

The aim of this project is to make sure that you understand and are familiar with the concepts covered in the lectures, including XML elements, attributes, statements, XML schema, XML validation, XML transformation (XSL), and the related library classes. By the end of the project, you should have applied these concepts and techniques in creating an XML file, its schema, its style sheet, and have written Web services and an SOA application to process these files.

This is an **individual project**. Each student must complete and submit independent work. No cooperation is allowed, even among the team members for project 3. Do not use WebStrar (which is shared among your team members) to host your files, applications, or services in project 4. You can use your ASU personal Web site space to host your XML, XSD, and XSL files (see Practice Exercises question 4) and use localhost to host the Web applications and services.

Practice Exercises (No submission required)

No submission is required for this part of exercises. However, doing these exercises can help you better understand the concepts and thus help you in quizzes or exams.

1. Reading: Textbook Chapter 4.

2. Answer the multiple choice questions 1.1 through 1.16 of the Text Section 4.8. Study the material covered in these questions can help you to prepare for the class exercises, quizzes, and the exam.

3. Study for the questions 2 through 8 in Text Section 4.8. Make sure that you understand these questions and can briefly answer these questions. Studying the material covered in these questions can help you to prepare for the exam and understand the assignments.

4. If you have not activated your file service and personal Web hosting site at ASU, you can activate them at: www.asu.edu/selfsub. Choose "Subscribe to additional computing access/services", and then chose "Personal Webpage Hosting". To deploy files to your personal Web site, you can go to

   MyASU --> View More --> My Files (AFS)

   You need to upload your XML/XSD files into the folder www, in order for you to access the files from web. Then, you can use the address: http://www.public.asu.edu/~YourASURITEID/myA4.xml to access your myA4.xml file.

   You can also search for "Uploading Your Personal Web page" within the ASU search page to find the steps for uploading your files. Notice that the ASU Personal Web site space hosts files only. You

can use it to host your .xml, .xsd, and .xsl files if needed for assignment or for your own exercises. The personal Web page does not host programs such as Web services and Web applications, because IIS are not installed. In order for your files to be accessible from the Web, you must put the files into the WWW directory, or it its sub-directories.

5 You can access the sample XML and XSD files at:

http://venus.sod.asu.edu/WSRepository/xml/Courses.xml
http://venus.sod.asu.edu/WSRepository/xml/Courses.xsd

Assignment Questions (Submission Required: 50 points)

1 In this question, you will create a directory of parks with a list of information related each park, which can be found through Google search. For example, Ground Canyon National Park information can be found at: https://en.wikipedia.org/wiki/Grand_Canyon_National_Park or https://www.nps.gov/grca. The information will be represented as an XML tree. The diagram below shows the required structure of the directory of Parks in XML tree notation that you will create in this assignment. All the "Park" elements have the same structure. Notice that different shapes and colors of boxes in the tree have different meanings. They represent elements, attributes, and text contents/values, respectively. The structure of elements and attributes given in the diagram in Figure 1 must be implemented as described, while the given text contents/values in the diagram are example's instance values and can be different in your files. The solid arrows show parent-child element relations, and the dotted arrows show the element-content or attribute-value relations. The optional attribute means that the XML instance can have the attribute or not have the attribute, without causing a verification error against its schema. However, it is still required to define the optional attribute in the XML Scheme file. In the instances, you must provide the attribute for some parks, but not for all parks. For the required attribute, you must provide this attribute for all such elements.

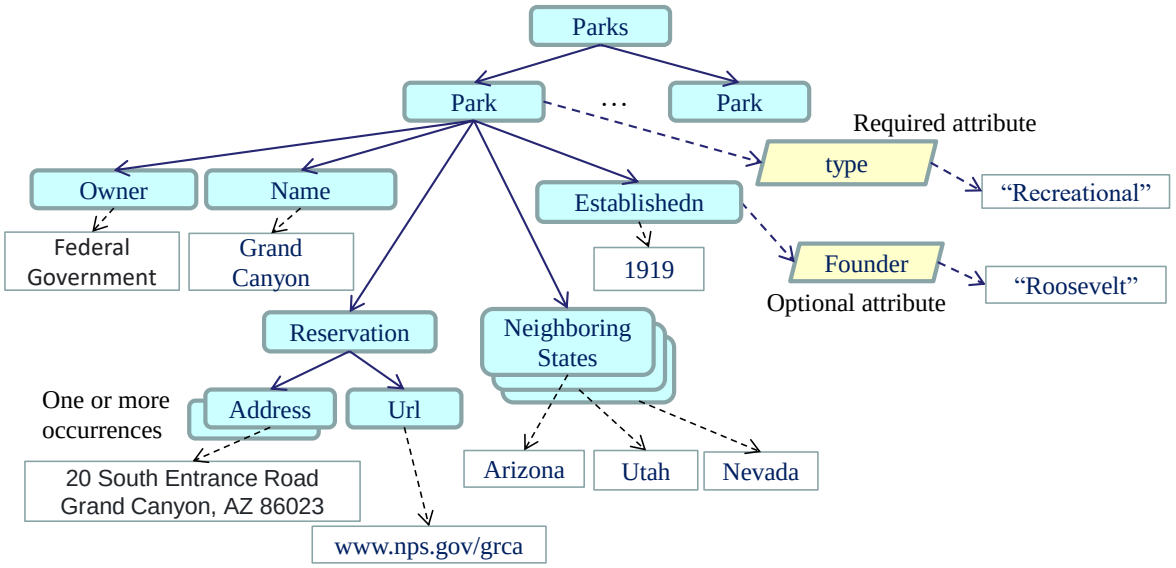

Figure 1. Required XML structure

1.1 Write the Parks.xsd file that defines the XML schema allowing the structure shown in Figure 1. You can use any tool to create/edit the file. [10 points]

1.2 Create an XML file Parks.xml as an instance of your schema file. Enter the information of at least ten (10) real parks into the Parks.xml file. You can use any tool to edit the file. You must include all the elements given in the XML tree in Figure 1. If an element has a Required attribute, you must provide the attribute for each of the elements. If an element has an Implied (Optional) attribute, you will provide this attribute for some elements, but not for all elements. [10 points]

2. Develop a Web service (.svc) containing **two** of the Web operations listed below. The node mentioned in the sub questions below includes every component (element, content, and attribute) shown in the XML tree in question 1. You choose two (and only two) following operations to implement. If you implement more than two operations, we will grade the first two operations only. If you have implemented and submitted a service listed below in your Project 3 as an elective service, you cannot choose that service here. [20 points]

2.1 Web operation "verification" takes the URL of an XML (.xml) file and the URL of the corresponding XMLS (.xsd) file as input and validates the XML file against the corresponding XMLS (XSD) file. The Web method returns "No Error" or an error message showing the available information at the error point. You must use the files that you created in this assignment, with and without fault injection, as the test cases. However, your service operation should work for other test cases (other XML file and its corresponding XSD file) too.

2.2 Web operation "search" takes the URL of an XML (.xml) file and a key (e.g., the element name Headquarter) as input. It returns the node's content information related to the search key, for example: park name, number, departure port, etc. Your program must also read any attributes. Attributes should be searched too. If there are multiple occurrences, you must return all of them. In this question, you can use DOM or SAX model. In the GUI (Question 3), you can display them all or once at a time through a "Display Next" button.

2.3 Web operation "XPathSearch" takes the URL of an XML (.xml) file and a path expression as input. It returns the path expression value of the given path. It could be a list of nodes, the content value, etc., depending on the path given.

2.4 Web operation "addPark" modifies the XML file by adding a new park element into the XML file. You can use multiple parameters or one parameter that contains all the information required for adding a new park into the tree. The method must work for any tree that conforms with your XML schema file. The original XML file must be read from a website. After the modification, the tree must be saved back to the file in App_Data in your working directory. In the TryIt page (Question 3), the added contents must be demonstrated.

Notice that, for all the questions above, do not place the XSD file as a namespace in your XML file. It may cause an exception in some library classes. The absence of the XSD namespace will not cause a problem with the schema validation and with XSL transformation, as your method will take the XSD file as a parameter if needed.

3. Create a Web application (ASPX) and add the project into the same "solution" that hosts your web services. The Web site application must provide a GUI (TryIt Page), which allows entering the required inputs, such as URLs and keyword, path, or contents, based on the questions that you select in Question 2. The GUI must have the buttons required to invoke the service operations, depending on the operations that you implement in the previous question, for example,

- The button that validates the XML file against the schema file;
- The button that searches by keyword or by path in the XML file.
- The button that adds a new park.

The Web application must use the Web services created in Question 2 to perform the required processing tasks and display the return message in the GUI. You can use a textbox, a list box, a label, etc., to display the results, display it in a separate page, or save then into an html file. This assignment will be implemented on localhost / IIS Express. You must make ensure that the application and the services are still linked when the assignment is graded on a different computer. Do not deploy your services and application into the WebStrar server. [10 points]

Note: The files, such as XML and XSD files, must be hosted in a website. Do not use WebStrar server to host your XML and XSD files. You can use your personal website offered by ASU at: http://www.public.asu.edu/~YourRITEID/ to host the files, follow the Question 4 in Practice Exercises section in this document. If the files must be modified, you must save the modified file into the local folder App_Data.

Testing and Submission

Testing: Based on your selection of the sub questions in question 2, provide your test inputs and test results. For example, if you chose questions 2.1, 2.2, or 2.3, you can place the files: Parks.xml and Parks.xsd into a Web site and use them to test your program on your IIS Express localhost. Inject (plant) an error into your XML file and make sure the validation service can detect the error. You must submit the test results (inputs and outputs) in screenshots, **for example**, including the followings:

(1) A screenshot of the GUI, including the output of the XML validation displayed in the GUI, with no error and with error messages;

(2) A screenshot of the GUI, including the input and output for keyword search;

These files above are not programs and can be deployed to any web location, such as the personal web space provided by ASU. See Question 4 in Practice Exercises section. Do not use the server for Project 3 in this project.

Canvas Submission list:

1) The complete solution folder with all project files for the services and the application that can be tested by the TA/Graders.

2) You also need to submit the two files Parks.xml and Parks.xsd. If you have selected the verification question, you must also include the XML file with error injected: ParksError.xml.

3) Put the screenshot of the testing results in a Word or PDF file. Zip all these files into a single zip file for submission.

General Submission Requirement

All submissions must be electronically submitted to the assignment folder where you downloaded the assignment paper. All files must be zipped into a single file.

Submission preparation notice: The assignment consists of multiple **distributed** projects and components. They may be stored in different locations on your computer when you create them. You must choose your own location to store the project when you create the project. Then, you can copy these projects into a single folder for the blackboard submission. To make sure that you have all the files included in the zip file and they work together, you must **test** them before submission. You must also download your own submission from the blackboard. Unzip the file on a different machine, and test your assignment and see if you can run the solution in a different machine, because the TA will test your application on a different machine.

If you submitted an empty project folder, or an incomplete project folder, we cannot grade your resubmission after the due date! We grade only what you submitted before the submission due date. Please read FAQ document in the course Web page for more details.

Late submission deduction policy:

- Grace period (Sunday): No penalty for late submissions that are received within 24 hours of the given due date.
- 1% grade deduction for every hour after the first 24 hours of the grace period (from Monday through Tuesday!
- No submission will be allowed after Tuesday midnight. The submission link will be disabled at 11:59pm on Tuesday. You must make sure that you complete the submission before 11:59pm. If your file is big, it may take more than an hour to complete the submission!

Grading and Rubrics

Each sub-question (programming tasks) has been assigned certain points. We will grade your programs following these steps:

(1) Compile the code. If it does not compile, 50% of the points given for the code under compilation will be deducted. Then, we will read the code and give points between 50% and 0, as shown in right part of the rubric table.

(2) If the code passes the compilation, we will execute and test the code using test cases. We will assign points based on the left part of the rubric table.

In both cases (passing compilation and failed compilation), we will read your program and give points based on the points allocated to each sub-question, the readability of your code (organization of the code and comments), logic, inclusion of the required functions, and correctness of the implementations of each function.

Please notice that we will not debug your program to figure out how big or how small the error is. You may lose 50% of your points for a small error such missing a comma or a space!

We will apply the following rubrics to **each sub-question** listed in the assignment. Assume that points assigned to a sub-question is *pts*:

Rubric Table

Major	Code passed compilation				Code failed compilation		
Points	pts * 100%	pts * 90%	pts * 80%	pts * 70% - 60%	pts * 50% - 40%	pts * 30% - 10%	0 or 1 point
For each sub-question	Meeting all requirements, well commented, and working correctly in all test cases	Working correctly in all test cases. Comments not provided to explain what each part of code does.	Working with minor problem, such as not writing comments, code not working in certain uncommon boundary conditions.	Working in most test cases, but with major problem, such as the code fail a common test case	Failed compilation or not working correctly, but showing serious effort in addressing the problem.	Failed to compile, showing some effort, but the code does not implement the required work.	0 points if no submission 1 point if submitted files that do not answer any questions required, for example, the files are empty, cannot be open, wrong files, etc.