How Will my project will be marked?

This project counts for 20% of your final mark and will be graded using the following grid:

Task	Mark
Good coding style including proper indentation and use of variable and class naming conventions and suitable comments	2
Parse JSON, XML, and SCV input to C# Dictionary generic class (The DataModeler class)	5
Create the CityInfo class for holding cities information	2
Create the Statistics class for manipulating Dictionary data	6
Creative but functional UI display the results out the Statistic class	2
The app runs with proper error management	2
Proper submission	1
Total	20 marks

Project Description

One of the current trends in modern software applications is to query different data sources with various data formats such as JSON, XML, and CSV. These formats are widely used in several business domains, including distributed systems and data analytics, to foster data retrieval and analysis.

In this project, we want to work on several data formats directly in C#. As such, we'll need to find ways to convert between JSON, XML, and CSV and C# generics classes in order to work with the data effectively.

The data that we're going to use in this project are extracted from the World Cities Database (https://simplemaps.com/data/world-cities). There are several specific goals that we're trying to achieve in working with the Cities data:

1. We need to be able to parse JSON, XML, and CSV files to a Dictionary generic type based on the user choice as of when the app starts, it should check the passing file type. You can write customized code for the parsing process, or you can use the available libraries in the Net framework, such as JSON.NET.
2. Build a Dictionary generic type in which its key equals the City name, and the value is an object of the city information. You need to handle the duplicate of the city name if there is any!
3. Create a statistics class that would enable the user to retrieve all information about the stored cities in the Dictionary generic type.

4. Based on your previous experience on developing web or form applications, we need to enable the user to access these statistics through a UI that allows him/her to select the city and present its all available information.
5. Statistics are meaningless unless we compare the cities. The application UI should enable the user to compare the cities or provinces based on population and location (using latitude and longitude). Also, the application should allow the user to know the total number of cities as well as the overall number of populations in any province, for instance.

Functional Requirements:

1. Create a non-generic class called CityInfo that will hold information about the city. It will have the followings:
    a. Prosperities: CityID, CityName, CityAscii, Population, Province, latitude, longitude.
    b. Methods: *GetProvince, GetPopulation, GetLocation*. The GetProvince returns the Province of the city. The GetPopulation returns the number of populations in the city. The GetLocation returns the latitude and longitude of a city.
    c. Add the necessary constructors according to your business and technical logic.
2. Create a non-generic class called DataModeler that will parse the data files. It will contain the followings:
    a. One parsing method for each file. They will be called: *ParseXML, ParseJSON, and ParseCSV*. All of these three methods must have the same signature. Each parsing method accepts only one input parameter (the file name) and returns void.
    b. A customized delegate matches the signature of the three previous parsing methods.
    c. Another method called ParseFile that will run one of the parsing methods in 2.a using the declared delegate in 2.b, according to the type of the passing file. The method will accept the file name and type as input parameters. This method will also return the value of the generic type dictionary. The key of that dictionary parameter is the city name, where its value will be the information of the city.
3. Create a class called Statistics that would include the followings:
    a. Property: a variable of the Dictionary generic type called CityCatalogue that holds the information of cities returned from the DataModeler class. The variable key is the city name itself, and the value is an object of the CityInfo class.

b. Constructor, (file name, file type). The user must specify the file name, which in this case will be “Canadacities” and then determine the file type or extension either to be JSON, XML, or CSV. You may get the value of the CityCatalogue here in this constructor by calling the DataModeler.Parse method.
c. City’s methods:
    i. *DisplayCityInformation*: This method will take a CityName parameter and return all the city stored information in the CityCatalogue dictionary variable.
    ii. *DisplayLargestPopulationCity*: It will return the largest population city in a province.
    iii. *DisplaySmallestPopulationCity*: It will return the smallest population city in a province.
    iv. *CompareCitiesPopluation*: This method will take two parameters; each represents one city. It will return the city with a larger population and the population number of each city.
    v. *ShowCityOnMap*: Use the name of the city and province to mark a city on the map. You may come up with a solution similar to what shown in this link (https://www.latlong.net/). **This method is a bonus (one mark)**
    vi. *CalculateDistanceBetweenCities*: This method calculates the distance between any two cities using the latitude and longitude of the input cities stored in the CityCatalogue dictionary variable. In order to calculate the distance, you can use one of the **free** Google services such as Google GeoCoding API. You may try other alternative APIs (if Google API fails) such as https://msdn.microsoft.com/en-us/library/ff701705.aspx
d. Province’s methods
    i. *DisplayProvincePopulation*: This method will take a province name parameter and return the sum of all populations of its cities saved in the CityCatalogue dictionary variable.
    ii. *DisplayProvinceCities*: This method will take the province name parameter and return a list of all cities of that province from the CityCatalogue dictionary variable.
    iii. RankProvincesByPopulation: It will rank and show all provinces by population. The order has to be ascending.
    iv. RankProvincesByCities: it will rank and show all provinces by the number of cities in each. The order has to be ascending.

Other Requirements and Notes:

The following is a list of requirements and constraints for your application:

1. Visual Studio and .NET have tools that can be used with this project.
2. Include detailed comments in your code describing the key aspects of your program.
3. Each class must be defined in a single separate cs file.
4. You can add what you think fits the application.
5. All input parameters to the Statistics methods have to be selected by the end-user of your application.
6. You are **not** allowed to edit the three source data files.
7. Create a "Data" folder in your project to store the three source data files.

How should I submit my project?

Electronic Submission:

Please submit the entire solution as a single zipped file to the submission folder Project 1 on FOL before the due date **Sunday, Feb 16, 11:59 PM**. Only one-day last submission is allowed and will be penalized with 20% of the whole project mark. Wrong submissions will not be marked! Please name the zipped file: **Project1_Gourp_X** (where X is the group number).
Any copied work will not be marked, and Fanshawe regulations in this situation will be applied.