Course:	**INFO5101**
Professor:	**Jim Cooper**
Project:	**Project #1 – MLB Scoreboard**
Due Dates:	**Wednesday, March 1, 2017**
Submitting:	**Please see the last page for instructions.**

Student Name: ___________________Student Number: __________

How will my project be marked?

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

Marks Available	What are the Marks Awarded For?	Mark Assigned
2	Good coding style including proper indentation and use of variable and class naming conventions and suitable comments	
2	De-serialize JSON input to C# classes	
2	Add in memory MLB data from multiple online queries	
2	Save in memory generic type as a single JSON string for a particular year	
2	Switch between years and automatically synch if necessary	
2	Selection of date from valid date list to display scoreboard property set	
4	Creative but functional UI display of scoreboard properties	
2	App runs with proper error management	
2	Proper submission	
20	Total	

Project Description

Overview:

One of the trends in modern web (and non-web) application development is the ability to make use of “cloud” data sources that will return JSON formatted data for virtually countless applications.

As you’ve already experienced in other parts of your training at the college, web browsers contain sophisticated tools for converting JSON into JavaScript objects and subsequently, JavaScript objects back into JSON. With this project, we want to work with the data in C# directly. As such, we’ll need to find ways to convert between JSON and C# classes and ultimately generic collections in order to work with the data effectively.

The data that we're going to use for this project is an online repository of Major League Baseball information. For the remainder of this spec we'll call this MLB data. There are several specific goals that we're trying to achieve in working with the MLB data:

1. We need to be able to cache the MLB data locally in the form of a single JSON document and we also want to be sure that we are not missing any data for the currently selected year.

2. When your app starts up, it should check to see if a current year value was previously assigned and if so, it will load the single MLB JSON cache from disk into an in-memory generic collection.

3. Once the generic collection has been created, you will then need to determine if the cache needs to be updated.

4. The steps necessary to update the cache will consist of:

    a. Checking the current cache for any missing dates from the beginning of the year to the current date
    b. Retrieving all scoreboard JSON docs for the missing dates
    c. Integrating the new docs (if any) into the in-memory generic collection
    d. Serializing the in-memory data structure back out to disk as a single JSON doc.

5. While the MLB site contains different kinds of data, we're only interested in the "scoreboard" data. See the demo apps that accompany this project spec.

6. For our purposes, the data is stored uniquely based on date and there is typically a JSON document returned for every day of the year whether there are games played or not. Part of the challenge in doing this project will be to distinguish the JSON that contains scoreboard data from the JSON that doesn't. **<u>It's important that you only allow users to select from dates that actually have scoreboard data</u>**.

7. You will need to create a user interface using either Windows Forms or XAML (WPF) that will allow users to select a year as the default for the current database and then once done, retrieve and store all scoreboard data for that year up to the current date. Among other things this will involve making use of some sort of calendar control.

8. Ideally, you should consider using two calendar controls, one to select the current year and one to select the current date. **<u>It is not required that you use calendar controls to select years and dates</u>**.

9. You will also need to design a screen segment which will be used to display the properties for the currently selected date (scoreboard). Which UI controls you use for this purpose is up to you however it's important to preserve the hierarchical relationships of the data. Note as well that we have a grading component based on how well this is done. As well, you should use the property names as they exist within the JSON data to label your UI property data. **It's not a requirement of this project to convert JSON property names to meaningful display names**.

10. The currently selected year can be saved as a single value in a text file which your app will look for at start up.

11. In addition, a user may choose to change the current year to a different one and this will involve checking to see if the incoming year scoreboard cache already exists and if so just load it up and do a synch to update the new MLB cache (if necessary).

12. Once your app has completed initialization and synched up the local MLB cache (for the currently selected year), the user should be able to select a date from the calendar control to grab a set of scoreboard properties and display the results of all properties and resolvable links.

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 including the "Paste Special" option.

2. In addition, you can use JSON.Net from NewtonSoft to perform all serialization and de-serialization between JSON and C#. See the MLB Demo2 app posted on FOL.

3. The basic strategy for implementing this project is to create a C# class set that maps to the MLB scoreboard JSON content. As we've seen only a small part of this needs to be done manually.

Enhanced Solution:

The basic solution only requires displaying properties including resolvable links as immediate values however, any students wanting to enhance the capabilities around the resolvable links should look at making these links clickable and then depending on the media type, find a way to display the image, video, audio etc.

Resources:

Also, have a look at the following coded examples on FOL and other links:

MLB Demo1
MLB Demo2

http://jsonlint.com/

http://json.codeplex.com/

How should I submit my project?

Electronic Submission:

Submit your program files to the *Info5101 “Project 1”* electronic dropbox in *FanshaweOnline*. These files should be submitted as a single “zip” file containing your web application’s complete website.

Submit your project on time!

Project submissions must be made on time! Late projects will be subject to divisional policy on missed test and late projects. In accordance with this policy, no late projects will be accepted without prior notification being received by the instructor from the student.

Submit your own work!

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.