Program:	**CPA2-5**
Course:	**INFO-5060**
Professor:	**Tony Haworth**
Project:	***Multiplayer Game***
Due Date:	**Friday, March 31, 2023 by 11:59 p.m.**
Last Update:	**Feb 28, 2023**

Note: This project must be competed in a group of three or four students.

Description

Design and develop a distributed multi-player game application that incorporates multiple assemblies and WCF.

Purpose

On completing this project you should be able to:

1. Design a distributed application that reasonably conforms to the principles of distributed programming
2. Implement, host, configure and consume a WCF service in a LAN computing environment
3. Set-up and use callbacks on a client application from a WCF service

Requirements

1. Choose a game that you will implement in C#. Note that *Scrabble* is not an option because of project 1. Furthermore, if you implement a game involving regular playing cards you may not use the *CardsLibrary* code from our in-class examples. Instead you should develop your own unique implementation. Experience suggests that you choose a simple game such as *Go Fish* or *Concentration* or even an adapted version of a two-player game like *tic-tac-toe.* Here are the characteristics of a suitable game:
    a. The game must be a multi-user program that implements a simple multi-player game. The game should not require "chatty" game service objects. Note that for full marks you are required to implement a game that can be played by a *variable* number of players (for example: 2 to 8 players, but you can choose the range).
    b. There must be some kind of shared game state (i.e. shared by all users/clients) that should be reported in the client's user interface and this state should change periodically for all instances of the client as the game progresses. In other words, all players must be made aware of the status of other players as appropriate to the game rules (for example: if one player/client scores a point then all other players/clients should probably see this).

Console App (.NET Framework)!!!

2. At a minimum, the application should incorporate the following *.NET Framework* assemblies that must be coded by you:
    a. **A service host assembly** that hosts the class (or classes) used to implement the game logic; this will be a self-hosting WCF service. You can use a Windows exe console or GUI assembly.
    b. **A class library assembly** that defines the class (or classes) that implements the WCF service (i.e. the game logic). This assembly should also define all the WCF contracts used. You must use at least one of each of the following contract types:
        i. Service contract
        ii. Data Contract
        iii. Callback Contract
    c. **A Windows GUI or console client** assembly that includes a user interface that adequately depicts the current game state and permits the user to play the game while preventing play that does not conform to the rules. The user interface should keep all players informed of which player is currently playing, how they are playing, and what the current score is as appropriate to the rules of your selected game.

3. **You MUST demonstrate your program** via a recorded video (see *Demonstrations & Submissions* below)

Tips

1. Start with a simple game concept and get that working first. Make sure you can use the basic service simultaneously using multiple instances of your client before adding client callbacks.
2. Some other client-server software can interfere with WCF. For example, if you have Skype running this may prevent your WCF application from working because they both use port 80 by default. The easiest solution is to shutdown Skype when testing your code.

Grading Scheme

1. You may lose marks if your code does not exhibit good coding style (proper comments, conforming to naming conventions, proper code layout, proper variable declarations and usages).

2. Here is a detailed marking scheme (next page). Note that many of your marks will be based on your project demonstration (see *Demonstrations & Submissions* below). If you don't demonstrate your project then you won't get those marks!

Item	*Marks*
Video Demo submitted addressing all required elements (See the next section *Demonstrations & Submissions* for information).	5
Application uses a WCF *service* contract	5
Application uses at least one WCF *data* contract	5
Application uses at least one WCF *callback* contract	5
Callback(s) work properly to update client status	5
Game can be played with a variable number of players	5
Service adequately manages game initialization (i.e. proper startup) for a variable number of players	5
Service maintains/manages the game state and enforces game rules including sequence of play (as applicable to the game implemented)	5
Clean and reasonably usable user-interface for supporting game play and prevents the user from playing in a way that violates the game rules	5
Game executes without unhandled exceptions	5
Deduction if significant game rules are coded into the client assembly rather than the service library	-10 (max)
Total	***50***

Demonstrations & Submissions

Project Demonstration (by recorded video):

You are required to demonstrate your game via a recorded video that you will submit with your project code. Your video demonstration must show the following elements and you should include narration to highlight each of these elements as you demonstrate them:

1. Loading the service application (service's user interface should be visible)
2. Loading at least two clients/players (both client user interfaces should be visible)
3. Starting and playing a game, showing game-play by multiple players
4. Clearly demonstrating that the game rules are enforced by the service (for example, a player can't play when it's not their turn or after the game is over)
5. Clearly demonstrating client callbacks (one player does something and other players immediately see something different in their user interfaces)
6. Ending a game including the status of each player at the end (i.e. player scores and/or who won) – *You can pause the video recording temporarily to get closer to the end of the game before continuing the recording since this could make the video very long otherwise!*
7. Starting another game using a <u>different</u> number of players.

<u>Please keep your video under 5 minutes in length.</u>

<u>Please read the next section on *submission* for instructions on submitting your video.</u>

Submission:

NOTE: Your source code files should identify all the students involved in the group. Only one student in a group needs to make the submission, but all members will be penalized if no submission is received.

Submit via the *Project 2* drop box in *FanshaweOnline*, an archive (zip or rar file) containing:

1. All of your source code including your (cleaned) Visual Studio solution and project files
2. Any application configuration files, if required for your program to run
3. Any special instructions that may be required to build[1], install and run your program *([1] note that your project must be "buildable" without any non-standard Visual Studio libraries or resources unless you include these with your submission)*
4. *A link to your video demonstration*

Late Submissions:

A penalty of 20% will be deducted if your submission is received < 24 hours late.
No marks will be awarded for projects that are submitted more than 24 hours after the submission deadline.

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.