# IT 230 Coding Activity Submission Template

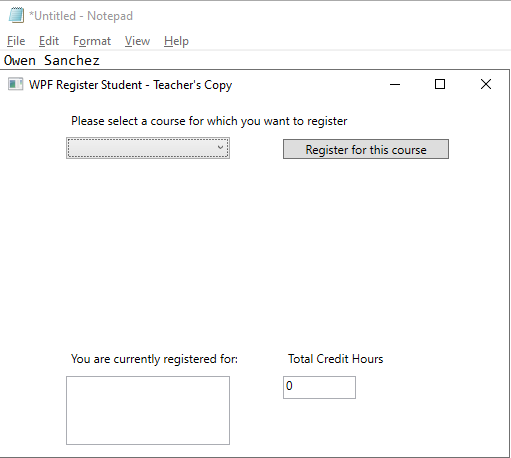
Submit your work on the coding activities for Modules One, Two, Three, Four, and Six in this document. In addition to this document, you should submit a ZIP file containing all your Visual Studio project files and source code that can be run in Visual Studio on a different computer.

For each coding activity, complete the following steps:

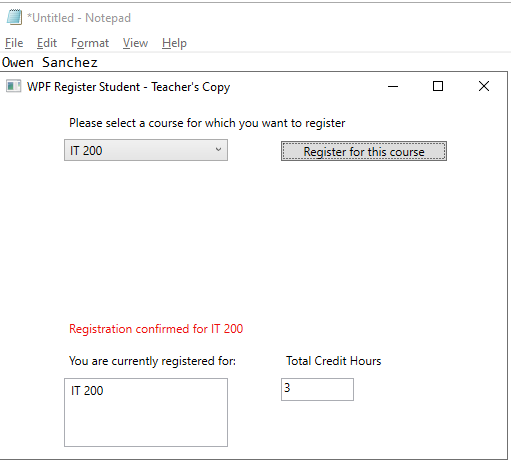
* Download and rename this document to meet the file naming conventions requested in the assignment instructions.
* Fill in the required information below by replacing the bracketed text with the relevant information.
* Submit this document and your ZIP file for grading and feedback. Your ZIP file should follow the same naming conventions.

Document your work in the coding activity by completing each of the following items:

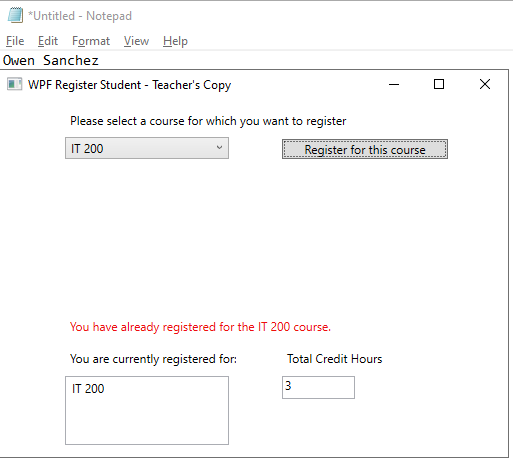
1. Provide a screenshot of the output that resulted from running your program successfully in Visual Studio. See the coding assignment instructions for an example of what should be included in the screenshot. Your screenshot must include the following elements:
   1. Your last name as the first printed text on the screen
   2. Verification that the program is fully functioning and data results are accurate for the given problem



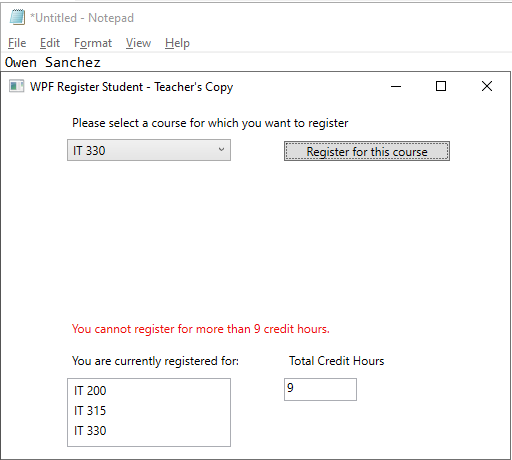
The main menu of the application before any courses have been registered.



Registering for a single course, with the course list and credit hours updated and a confirmation message displayed.



Attempting to register for the same course again results in an error message.



Attempting to register for more than 9 credit hours results in an error message.

1. Copy and paste the source code text you wrote for this assignment from the \*.cs file into the space below. Only providing the \*.cs files or a screenshot does not meet the requirements for this part of the assignment. Code should be logically organized. It should also follow proper syntax and conventions noted in the Coding Activity Guidelines and Rubric.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace WPFRegisterStudent

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

Course choice;

public MainWindow()

{

InitializeComponent();

}

// Returns the total number of credits for which the student is registered

private int GetTotalCredits()

{

int sum = 0;

// Add 3 credits per registered course

foreach (Course c in this.listBox.Items)

{

sum += 3;

}

return sum;

}

// Updates the credit count in the UI

private void UpdateCreditCount()

{

this.textBox.Text = Convert.ToString(GetTotalCredits());

}

private void Window\_Loaded(object sender, RoutedEventArgs e)

{

Course course1 = new Course("IT 145");

Course course2 = new Course("IT 200");

Course course3 = new Course("IT 201");

Course course4 = new Course("IT 270");

Course course5 = new Course("IT 315");

Course course6 = new Course("IT 328");

Course course7 = new Course("IT 330");

this.comboBox.Items.Add(course1);

this.comboBox.Items.Add(course2);

this.comboBox.Items.Add(course3);

this.comboBox.Items.Add(course4);

this.comboBox.Items.Add(course5);

this.comboBox.Items.Add(course6);

this.comboBox.Items.Add(course7);

// Update the credit count to its initial value

UpdateCreditCount();

}

// Sets the status text for when the user attempts to register for a course

private void SetStatusMessage(string text)

{

this.label3.Content = text;

}

private void button\_Click(object sender, RoutedEventArgs e)

{

// Get the user's choice

choice = (Course)(this.comboBox.SelectedItem);

// Check that the user selected a course

if (choice == null)

{

SetStatusMessage("Please select a course first.");

return;

}

// Prevent the user from registering for too many courses

if (this.listBox.Items.Count >= 3)

{

SetStatusMessage("You cannot register for more than 9 credit hours.");

return;

}

// Prevent the user from registering for the same course more than once

if (choice.IsRegisteredAlready())

{

SetStatusMessage("You have already registered for the " + choice.getName() + " course.");

return;

}

// Mark the course as registered

choice.SetToRegistered();

// Add the course to the registration list

this.listBox.Items.Add(choice);

// Update the credit count

UpdateCreditCount();

// Display a confirmation message

SetStatusMessage("Registration confirmed for " + choice.getName());

}

}

}

1. Show that you understand the task by explaining the design of your program in the space below. Include the process and steps you took to write your code. Explain how you arrived at the solution to the problem and completed the activity.

The program is a WPF application with a GUI that allows the user to select and register for courses. Upon selecting a course and clicking the register button, a method runs, which checks the validity of the selection. It checks whether a course was selected, whether the course registration limit has been reached, and whether the course has already been registered, and it displays an appropriate status message for all cases. I solved the problem by examining the software requirements documentation, determining how each feature could be implemented, and finally implementing all the requirements.

1. Reflect on your learning experience and what you learned from completing the activity.

I learned more about working with WPF in this activity, especially how list-like GUI elements work. I feel more confident in my ability to create WPF applications in the future, and I also gained familiarity with error handling and method creation in C#.