# 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.

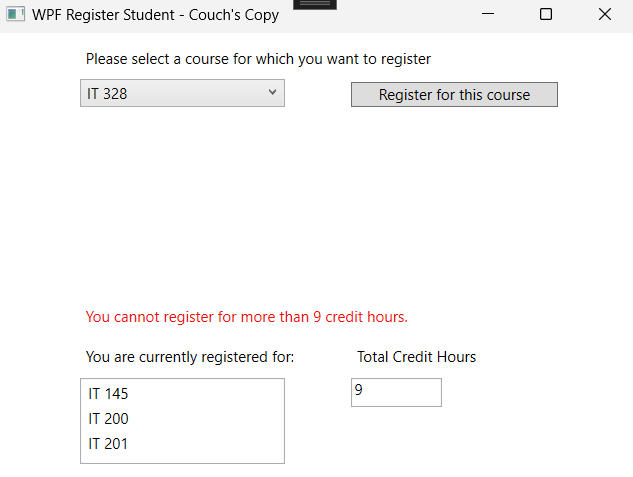
Document your work in the coding activity by completing each of the following items:

1. Provide a screenshot

A screenshot of a computer screen

Description automatically generated

The screenshot shows the application after successfully registering for three courses, each carrying 3 credit hours. The "Total Credit Hours" textbox displays "9", and the list of registered courses includes "IT 145", "IT 200", and "IT 201".



The screenshot shows the application attempting to register for a fourth course, which would exceed the maximum allowed 9 credit hours. An error message is displayed, stating "You cannot register for more than 9 credit hours."

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.

Edited MainWindow.xaml 🡪 <Window x:Class="WPFRegisterStudent.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:WPFRegisterStudent"

mc:Ignorable="d"

Title="WPF Register Student - Couch's Copy" Height="395.902" Width="525" Loaded="Window\_Loaded">

<Grid>

<ComboBox x:Name="comboBox" HorizontalAlignment="Left" Margin="66,37,0,0" VerticalAlignment="Top" Width="164"/>

<Button x:Name="button" Content="Register for this course" HorizontalAlignment="Left" Margin="283,39,0,0" VerticalAlignment="Top" Width="166" Click="button\_Click"/>

<ListBox x:Name="listBox" HorizontalAlignment="Left" Height="69" Margin="66,276,0,0" VerticalAlignment="Top" Width="164"/>

<TextBox x:Name="textBox" HorizontalAlignment="Left" Height="23" Margin="283,276,0,0" TextWrapping="Wrap" Text="0" VerticalAlignment="Top" Width="73" IsReadOnly="True"/>

<Label x:Name="label" Content="Please select a course for which you want to register" HorizontalAlignment="Left" Margin="66,7,0,0" VerticalAlignment="Top" Width="383"/>

<Label x:Name="label1" Content="You are currently registered for:" HorizontalAlignment="Left" Margin="66,245,0,0" VerticalAlignment="Top" Width="176"/>

<Label x:Name="label2" Content="Total Credit Hours" HorizontalAlignment="Left" Margin="283,245,0,0" VerticalAlignment="Top" Width="106"/>

<Label x:Name="label3" Content="" HorizontalAlignment="Left" Margin="66,213,0,0" VerticalAlignment="Top" Width="383" Foreground="#FFEE0E0E"/>

</Grid>

</Window>

MainWindow.xaml.cs 🡪

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

{

public partial class MainWindow : Window

{

private Course choice;

private List<Course> courses;

private int totalCredits = 0;

private const int MaxCredits = 9;

public MainWindow()

{

InitializeComponent();

}

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

{

courses = new List<Course>

{

new Course("IT 145"),

new Course("IT 200"),

new Course("IT 201"),

new Course("IT 270"),

new Course("IT 315"),

new Course("IT 328"),

new Course("IT 330")

};

foreach (var course in courses)

{

this.comboBox.Items.Add(course);

}

this.textBox.Text = "0";

}

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

{

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

if (choice == null)

{

this.label3.Content = "Please select a course.";

return;

}

if (choice.IsRegisteredAlready())

{

this.label3.Content = "You have already registered for this course.";

return;

}

if (totalCredits + 3 > MaxCredits)

{

this.label3.Content = "You cannot register for more than 9 credit hours.";

return;

}

choice.SetToRegistered();

totalCredits += 3;

this.textBox.Text = totalCredits.ToString();

this.label3.Content = $"Successfully registered for {choice.getName()}.";

UpdateRegisteredCoursesList();

this.comboBox.Items.Remove(choice);

}

private void UpdateRegisteredCoursesList()

{

this.listBox.Items.Clear();

foreach (var course in courses)

{

if (course.IsRegisteredAlready())

{

this.listBox.Items.Add(course.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 that enables students to register for courses, each carrying 3 credit hours, with a maximum of 9 credit hours. On loading the window, the available courses are initialized and added to a ComboBox. When a user selects a course and clicks the "Register for this course" button, the program validates the selection by ensuring a course is chosen, the course is not already registered, and the total credit hours do not exceed 9. If validation passes, the course is marked as registered, the total credit hours are updated, and the course is removed from the ComboBox. The ListBox is then updated to display the registered courses, and the total credit hours are shown in the TextBox. This design ensures dynamic interaction and state maintenance in the application.

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

Completing this activity reinforced my understanding of WPF application development and C# programming concepts, such as event handling, data binding, and validation logic. I learned how to create a user interface with dynamic elements that respond to user input and maintain state across interactions. The experience of debugging and resolving issues also improved my problem-solving skills and attention to detail, ensuring the application met the specified requirements.