PROG2121 – WINDOWS PROGRAMMING

A02 – WPF APPLICATION

OVERVIEW

WPF is a rich platform for applications. It enables a Model-View-ViewModel Design Pattern for C#. The descriptive approach to the development of the UI is similar to the way most mobile application development is done today.

This assignment will give you an opportunity to work with many common aspects of a WPF application. You will be implementing a text editor, similar to Notepad.

OBJECTIVES

- Create Windows applications using the .NET Framework
- Demonstrate the MVVM pattern in a Windows application

ACADEMIC INTEGRITY AND LATE PENALTIES

- Please refer to the SET Policies document regarding Academic Integrity Information
- Please refer to the SET Policies document regarding Late Policy

EVALUATION

- Please see rubric on eConestoga

PREPARATION

- Review the material in the following:
  Layout - WPF .NET Framework | Microsoft Docs
  Focus on the behaviours of the Panels.

SAMPLE EXECUTABLE / SAMPLE INPUT AND OUTPUT

The Assignment is based on Microsoft's Notepad. Although not fully implemented in this assignment, the parts you are required to implement should behave like they do in Notepad.

REQUIREMENTS

Individual Submission or Group of 2 submission (No other options!)

1. The program must be written in C# using the WPF platform. As a reminder, use the .NET Framework, NOT .NET Core.
2. You must use GitHub with the link provided in class/lab. As you develop your solution, please commit regularly to the repository. If you are a group of 2, only 1 repository is to be used. Make sure you identify the repository in the comments for your eConestoga submission.

3. The newest version of Notepad supports multiple open files at one time using Tabs. You DO NOT need to implement this. Implement support only for one file.
4. The overall user interface should have a menu at the top, a status bar at the bottom, and the remaining area in between as the work area for editing.
5. Implement a menu that has “File   Edit Help” as the top level item.
6. Implement “New”, “Open”, “Save”, “Save As” and “Exit” as the menu items in “File”. For each of these, examine how Notepad works and mimic the behaviour.
    - “New” allows you to start a new file for editing. If there is unsaved text in the work area, you must give the user the chance to save the file. Use the Save File Dialog if a save is requested.
    - “Open” must display the Open File Dialog and allow you to choose a text file to Open and load into the work area of your application. If there is unsaved text in the work area, you must give the user the chance to save the file. Use the Save File Dialog if a save is requested.
    - “Save” must save the date in the currently open file. If a file has not yet been Opened or Saved (To), you must use the Save File Dialog
    - “Save As” must display the Save File Dialog and allow you to save a text file with the content from your work area.
    - “Exit” closes the application. If the current text in the work area has not been saved, you must give the user the opportunity to save before completely closing the application. In fact, if there is unsaved text, when choosing Exit, the user should be allowed to either Save and Exit, Exit without Saving, or Cancel the Exit operation. The “Exit” is identical in action to the “X” in the top right corner of the window.
7. The title on the top of the form must include the name of the file as you would find with most Windows applications. If there is no file yet in use (or saved to), the name is “Untitled”.
8. Implement “About” as the menu item in “Help”
    - About should bring up a modal About Box with the standard information about your application. It should behave like most common about boxes in Windows applications. You may create one from scratch or use something that might be available in .NET. Do not use a MessageBox for this.
9. Implement a Status Bar at the bottom of the window that displays the current count of characters in the work area. As you type, or delete, you should have this number updated. You do not have to remove any white space.
10. Using WPF Commanding, implement Cut, Copy and Paste for the “Edit” menu.
11. Follow SET coding standards.

FILE NAMING REQUIREMENTS

- There are no specific file naming requirements. Please keep the name short, but meaningful.
- The submission must be a .zip file (or fully compatible)

SUBMISSION REQUIREMENTS

- The full C# solution should be submitted in *one* zip folder. Only *.zip* (or fully compatible) format is acceptable.
- Several GitHub commits to show development progress. If a group submission, commits are expected from both members to the same repository.