# PROG 2500 Windows PRogramming - Assignment 1

# WPF Applications

Assignment Value: 12.5% of overall course mark.

Due Dates: **See assigned due date on Assignment 1 dropbox – Marked by code review during scheduled class time**

Late submissions will receive the standard late submission penalty as stated in the course outline. (5% overall deduction per day late, and 0% after assignment handed back to the class.)

#### Assignment Instructions:

In Visual Studio, use XAML/C# to create a WPF application as described by the requirements of this assignment.

#### Submissions:

Submission of work will be received through your GitHub repository. Please ensure that the commit you wish to have marked is labelled with an appropriate Commit Comment in your repo, and a text file containing the Commit ID is uploaded to the Assignment 1 dropbox on Brightspace. Unless otherwise specified, your most recent commit prior to the assignment deadline will be the one chosen for evaluation.

#### Evaluation:

To insure the greatest chance of success on this assignment, be sure to check the marking rubrics at the end of this document or in Brightspace. The rubrics contain the criteria your instructor will be assessing when marking your assignment.

## Program – Media Player/MP3 Tagger

## Design and write a program allows users to select and play an MP3 audio file, as well as edit and save some of the MP3 file’s tagged metadata.

On application start, the user can open a file dialog, browse to an MP3 audio file, and open it. When an MP3 has been opened, the user can use either a menu option or toolbar buttons to control the music’s playback state (Play, Pause or Stop). After a song is selected, the primary song metadata (Title, Artist, Album, Year) shall be displayed in a “Now Playing” screen. The song’s current time progress should also be shown in a timer, as well as use a slider control to visually display the progress and allow the user to move to any point in the song.

Once a song has been selected, the user should be able to toggle the display between the “Now Playing” screen and a tag-editing screen. If the user makes changes and saves any tag data, the changes should be written back into the MP3 file’s tag metadata.

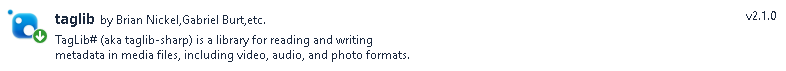
Technical Requirements

Your solution should be built to include the following technical specs:

* WPF application using XAML and C#.
* Use CommandBindings for the media and application controls
* Use at least one User Control (Suggested use: Now Playing and Tag Editor screens)
* Use at least three layout managers to create an intuitive and flexible user interface.
* At minimum, the app should be able to read and write the following tag data: Song Title, Artist, Album, Year. Other tags can be used as desired.
* Implement basic exception handling to avoid program crashes.

Technical Resources

For accessing and editing ID3 tag metadata from the Mp3, it is suggested that you install the TagLib-Sharp package shown below. This third-party package allows reading and writing of ID3 tag data in MP3 files. Documentation and examples can be found on their website: <http://taglib.org/api/>



For creating a media player, you can find an example program at the website listed below:

<http://www.wpf-tutorial.com/audio-video/how-to-creating-a-complete-audio-video-player/>

**Important Note:** If you refer to work or code from a website or other resources, whether you copy any code or not, it MUST be cited in your work. The references provided are not intended for you to just copy-paste… you are expected to cite anything from resource sites, and will be asked to explain how they work.

Sample Screenshots

|  |  |
| --- | --- |
| File Menu | Media Menu |
| Now Playing screen | Tag Editor screen |
| Open File dialog | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Application – Media Tagger** | |  |  |  |  |
| **Criteria** | **Insufficient (0 pts)** | **Good (1 pt)** | **Excellent (2 pts)** | **Mark** | **X** |
| **Functionality** | Multiple requirements or elements not working as expected, or missing. | At least one requirement or element does not work as expected, or is missing. | Application meets or exceeds all expected functional requirements, with no errors or omissions. |  | 3 |
| **Layout Managers** | Use of Layout Managers does not demonstrate a basic understanding of XAML layout principles and controls, or layout manager usage was insufficient/missing. | Use of Layout Managers demonstrates a basic understanding of XAML layout principles and controls, or three different types of panels/grids were not used, or not used appropriately. | Use of Layout Managers demonstrates a strong understanding of XAML layout principles and controls. At least three different types of panels/grids used appropriately in the interface layout. |  |  |
| **ommand Bindings** | Command Bindings functionality implemented, but contains multiple errors or omissions, or is not implemented. | Command Bindings functionality implemented, but contains at least one error or omission. | Implementation of media playback and application commands demonstrate strong comprehension of Command Bindings concepts. |  |  |
| **User Controls** | User Control(s) implemented, but contains multiple errors or omissions, or is not implemented. | User Control(s) implemented, but contains at least one error or omission. | Implementation of User Controls demonstrates strong understanding of concept. |  | 2 |
| **ID3 Tag Reading** | Reading/displaying of ID3 tag data missing or contains multiple errors/omissions. | Application can partially read and display ID3 tags stored in file metadata, but contains at least one error/omission. | Application can successfully read and display ID3 tags stored in file metadata. |  |  |
| **ID3 Tag Writing** | Saving of ID3 tag data missing or contains multiple errors/omissions. | Application can save ID3 tag data, but contains at least one error/omission. | Application can successfully save data changes to ID3 tags stored in file metadata. |  |  |
| **Slider/Timer Controls** | Slider and timer controls not implemented, or contain multiple errors/omissions. | Slider and timer controls implemented, but contain at least one error/omission. | Slider and timer controls implemented and working as expected. |  |  |
| **Interface Usability & Navigation** | Interface operability and navigation contains multiple errors/omissions, is non-very non-intuitive or difficult/confusing to use. | Interface operability and navigation contains at least one error or omission, is non-intuitive or difficult/confusing to use. | Interface operability and navigation is smooth and intuitive, with no errors/omissions. |  | 2 |
| **Exception Handling** | No try-catch blocks implemented or used appropriately. | At least one try-catch block used appropriately. | Try-catch blocks implemented correctly, and cover all reasonably foreseeable exceptions. |  |  |
| **Polish & Detail** | Application has little to no polish/usability. | Application has some polish, but has some areas of potential improvement. | Application is well-polished and presentable, with evidence of lots of attention to detail and thought towards presentation/usability. |  |  |
| **Structure & Efficiency** | Poor or missing. | Good, but some improvements can be made. | Code and solution architecture shows strong understanding of good developer practices in code commenting, method and class use, and general code efficiency. |  | 2 |
|  |  |  | **Total:** |  | **/32** |