# C# WPF Media Player
for UTM CSCI 352

Zachary Rose, Logan Walsh

Abstract

Our project is a custom video player as a C# WPF application. It will allow you to browse to a file, load up a video, and control the playback as needed. The target is those who want more control over their video playback beyond what Windows comes with.

1. Introduction

For this project, we plan to make a media player program that offers some features that are unavailable in Windows Media Player. It will include a file browser to select a media file, and it will include the controls for playback such as: play/pause, fast forward/slow down, and skip to different parts of the video, as well as volume controls.

This project is intended for those who want some finer control over the playback of their video files. Windows Media Player does have the basic features for controlling playback of media, but it leaves much to be desired in customization and more niche functionality.

1.1. Background

Some useful terminology:

- media: in this case, a file containing video and/or sound
- media player: a program that can playback different media, and it usually allows for some control over different aspects of the playback
- playback: media is in playback when it is currently being shown to the user
- file browser: a user interface to make selecting a file from the system convenient
- Windows Media Player: a media player that comes with Windows OS by default. Very bare-bones.

Why a media player? We thought it would be interesting to make a tool that we might actually use, rather than some project for the sake of displaying some unusual programming concept. It will be good practice towards developing a user interface for a user-oriented application.

1.2. Impacts

Our hope is that this project will become a useful little tool for those who wish to have a light-weight and customizable media player. VLC media player can sometimes feel daunting and over bloated for an average use case, so our goal is to design something simple and functional.

1.3. Challenges

By far the hardest thing will likely be designing an interface to dynamically change key bindings. It will need a file to store them, the ability to parse and edit said file, and an interface with which the user will change the bindings from within the app.

2. Scope

Basic goals for getting started: You can load up a video, play and pause it using buttons, and it can be replayed.

The project will be done once the program runs without any issues and has enough features to differentiate it from Windows Media Player.

We think that this is a worthwhile project because it has practical use, and it will allow us to make further use of the WPF framework. It will require an attractive interface and different form elements working together behind the scenes.

2.1. Requirements

The functional requirements for a media player are straight forward for anyone who has used one before. We considered what we want to be able to do when watching a video, as well as what is absolutely necessary to do so. Loading a file is the absolute base requirement, but we've come to expect things like pausing and fast forwarding.

2.1.1. Functional.

- Design a functional interface
- Can play video files
- Has a video timeline
- Full screen button
- Pause/play button
- Fast forward and slower playback speeds
- Volume control

Stretch Goals:

- Support for MP3 files
- Customization for key bindings
- Combining multiple files into a playlist with associated skip forward/backwards buttons
- UI scales with the window

2.1.2. Non-Functional (Incomplete from here).

- Security – user credentials must be encrypted on disk, users should be able to reset their passwords if forgotten
- you'll typically have fewer non-functional than functional requirements

2.2. Use Cases

This subsection is arguably part of how you define your project scope (why it is in the Scope section...). In a traditional Waterfall approach, as part of your requirements gathering phase (what does the product actually *need* to do?), you will typically sit down with a user to develop use cases.

You should have a table listing all use cases discussed in the document, the ID is just the order it is listed in, the name should be indicative of what should happen, the primary actor is typically most important in an application where you may have different levels of users (think admin vs normal user), complexity is a best-guess on your part as to how hard it should be. A lower number in priority indicates that it needs to happen sooner rather than later. A sample table, or Use Case Index can be seen in Table **??**.

Use Case Number:	1
Use Case Name:	Add item to cart
Description:	A shopper on our site has identified an item they wish to buy. They will click on a "Add to Cart" button. This will kick off a process to add one instance of the item to their cart.

You will then go on to (minimally) discuss a basic flow for the process:

1) User navigates to page listing desired item
2) User left-clicks on "Add to Cart" button.
3) User cart is updated to reflect the new item, this also updates the current total.

Termination Outcome: The user now has a single instance of the item in their cart.

You may need to also add in any alternative flows:

Alternative: Item already exists in the cart

1) User navigates to page listing desired item
2) User left-clicks on "Add to Cart" button.
3) User cart is updated to reflect the new item, showing that one more instance of the existing item has been added. This also updates the current total.

Termination Outcome: The user now has multiple instances of the item in their cart.

You will often also need to include pictures or diagrams. It is quite common to see use-case diagrams in such write-ups. To properly reference an image, you will need to use the `figure` environment and will need to reference it in your text (via the `ref` command) (see Figure 1). NOTE: this is not a use case diagram, but a kitten.

After fully describing a use case, it is time to move on to the next use case:

Use Case Number: 2

Use Case Name:	Checkout
Description:	A shopper on our site has finished shopping. They will click on a “Checkout” button. This will kick off a process to calculate cart total, any taxes, shipping rates, and collect payment from the shopper.

You will then need to continue to flesh out all use cases you have identified for your project.

Figure 1. First picture, this is a kitten, not a use case diagram

2.3. Interface Mockups

At first, this will largely be completely made up, as you get further along in your project, and closer to a final product, this will typically become simple screenshots of your running application.

In this subsection, you will be showing what the screen should look like as the user moves through various use cases (make sure to tie the interface mockups back to the specific use cases they illustrate).

3. Project Timeline

Go back to your notes and look up a typical project development life cycle for the Waterfall approach. How will you follow this life cycle over the remainder of this semester? This will usually involve a chart showing your proposed timeline, with specific milestones plotted out. Make sure you have deliverable dates from the course schedule listed, with a plan to meet them (NOTE: these are generally optimistic deadlines).

4. Project Structure

At first, this will be a little empty (it will need to be filled in by the time you turn in your final report). This is your chance to discuss all of your design decisions (consider this the README’s big brother).

4.1. UML Outline

Show the full structure of your program. Make sure to keep on updating this section as your project evolves (you often start out with one plan, but end up modifying things as you move along). As a note, while Dia fails miserably at generating pdfs (probably my fault), I have had much success with png files. Make sure to wrap your images in a `figure` environment, and to reference with the `ref` command. For example, see Figure 2.

Figure 2. Your figures should be in the *figure* environment, and have captions. Should also be of diagrams pertaining to your project, not random internet kittens

4.2. Design Patterns Used

Make sure to actually use at least 2 design patterns from this class. This is not normally part of such documentation, but largely just specific to this class – I want to see you use the patterns!

5. Results

This section will start out a little vague, but it should grow as your project evolves. With each deliverable you hand in, give me a final summary of where your project stands. By the end, this should be a reflective section discussing how many of your original goals you managed to attain/how many desired use cases you implemented/how many extra features you added.

5.1. Future Work

Where are you going next with your project? For early deliverables, what are your next steps? (HINT: you will typically want to look back at your timeline and evaluate: did you meet your expected goals? Are you ahead of schedule? Did you decide to shift gears and implement a new feature?) By the end, what do you plan on doing with this project? Will you try to sell it? Set it on fire? Link to it on your resume and forget it exists?

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.