**FlickFix Defense Documentation**

**CSCI Senior Project**

**November 27, 2023**

**Matthew T. Weigel – Bachelor of Science in Computer Science**

**Dr. Sean Hayes – Project Advisor**

**Charleston Southern University**

**Statement of Purpose**

The purpose of this project was to create a free video editing desktop application that primarily focuses on simplicity and usability. This project is highly motivated by my personal experiences with the issues described in my problem statement. As a result, I have a strong interest in using this application myself. Additionally, this project presents a significant opportunity for me to expand my knowledge and experience in the field of computer science. By utilizing new frameworks to build my most complex program yet, I will gain valuable skills and insights that will benefit me both personally and professionally.

**Problem Statement**

Ideally, a video editing program is both user-friendly and intuitive, perfect for anyone who wants to create video projects. Unfortunately, many existing video editing programs are both cumbersome and unintuitive, making it difficult for casual users to make simple edits to their video clips. This is a problem for people such as educators or content-creators, who need to edit video recordings as part of their job. The proposed solution to this problem is to develop a simplified video editing software program with a user-friendly interface, streamlined features, and easy accessibility for casual users. This solution benefits the people who are not video editing enthusiasts, but who want to make easy edits to their video clips without having to spend time learning complex software. In summary, the problem of complex and cumbersome video editing software can be solved by developing software that is more accessible to a wider range of users by simplifying the video editing process, making it less time-consuming and more enjoyable for everyone.

**Research & Background**

Entering the initial phase of my senior project, I had limited knowledge about the Qt Creator integrated development environment (IDE), except for its capability to develop desktop applications. However, I was familiar with its primary use of C++, a language I had become comfortable with, making it a top contender among frameworks for my project. Another option I considered was JavaFX, known for using Java in desktop application development. After evaluating both, I ultimately chose Qt Creator for its user-friendly interface and integrated UI designer, which perfectly aligned with my needs. To fully grasp Qt Creator's potential, I spent the summer delving into research and watching numerous tutorials on YouTube. This helped me learn how to utilize Qt Creator's extensive tools and features effectively, enabling me to transform my application into a fully functional video editor.

**Project Language(s), Software, and Hardware**

This project was developed in C++ using the Qt Framework within Qt Creator 5.15.2. Additional necessary software includes the FFmpeg multimedia processing library and the K-Lite Basic Codec Pack. The only necessary hardware is a computer running Windows 10.

**Project Requirements**

* Requirements Document: <https://github.com/mtweigel/CSU-Senior-Project/blob/master/docs/FlickFix%20Requirements.md>

**Project Implementation Description & Explanation**

**Description**: FlickFix is a free video editing desktop application that emphasizes simplicity and ease-of-use. FlickFix aims to offer an intuitive and user-friendly video editing experience by streamlining the platform to include only the essential features. By adopting this approach, even novice editors can effortlessly edit videos without feeling intimidated or overwhelmed. Users will have access to several common features such as video trimming, intro and outro creation, and audio dubbing, all contained in a user-friendly GUI.

* Source code repository: <https://github.com/mtweigel/CSU-Senior-Project>

**Implementation Explanation:**

Once FlickFix has been opened, users are greeted with a simple, yet clean user interface (UI). See Fig 1.

A screenshot of a computer

Description automatically generated

Fig 1. The main user interface of FlickFix.

**Importing a Video**

To import a new video you have several different options. You can import from the menu bar by clicking ‘File’->’Import’, by clicking on the center video player, by dragging and dropping the video file into the video player, or by clicking either the ‘Append Video’ or ‘Prepend Video’ buttons.

Once you have initiated an import, the standard select video file window pops up which allows you to select which file you want to import. After a video file has been chosen, the program loads the video into the player. See Fig 2.

A screenshot of a video

Description automatically generated

Fig 2. After a video has been imported.

The application is careful to ensure that only valid files can be imported – only mp4, avi, mkv, mov, and wmv video files are accepted by FlickFix. If you attempt to import an invalid file type, you will receive a warning and the file is not imported. See Fig 3.

A screenshot of a computer

Description automatically generated

Fig 3. Warning after trying to import invalid file.

In addition to FlickFix being able to import videos of different file types, it can import videos of different resolutions. It can do both because when it imports a video, it automatically converts it to mp4, 1080p for editing within the program. This keeps things consistent and makes the job easier for FFmpeg which handles the editing processes.

**Video Player Controls**

FlickFix utilizes standard video controls for its video player which helps to add familiarity to the user interface. See Fig 4. For precise editing, the Seek Forward and Backward buttons are designed to advance or reverse the video by 0.1 seconds with each click. If these buttons are held down, they continuously progress or rewind the video. The video duration slider enables quick adjustments to the video's position, while the volume slider offers control over the audio playback volume. Additionally, the mute button is available to toggle the sound on or off during video playback.



Fig 4. Video player controls.

**Adding New Videos**

Adding new video to a project in FlickFix is very simple. To add a new video after the existing video, you would press the ‘Append Video’ button and choose a video to add. See Fig 5. To add a new video before, you would press ‘Prepend Video’ and do the same. The adding video process adheres to the same file validity checks as when importing a video.

A screenshot of a computer

Description automatically generated

Fig 5. Window for Importing or Adding new video.

**Saving the Project**

To perform an initial save on a video project, you can either click ‘File’->’Save As’ or ‘File’->’Save’ in the menu bar. Doing so will pop up a save window where you can choose the location and name of your saved video. After saving to a location and making a new change, pressing ‘File’->’Save’ or ‘Ctrl + S’ will overwrite the save. If you forget to save and try to close the application, you receive a warning that you have unsaved changes. See Fig 5. Selecting ‘Cancel’ will cancel the exit, ‘Discard’ closes without saving, and ‘Save’ prompts the same save process as before. This same process also happens if you try to import a new video while a video has already been imported.

A screenshot of a computer error

Description automatically generated

Fig 6. Unsaved changes warning.

**Trimming the Video**

To begin trimming the video, you can press the ‘Trim Video’ button, which will bring up the trim controls. See Fig 7. To trim a portion of the video, choose a start point in the video and then press the ‘Start Point’ button. Then choose an end point in the video with the ‘End Point’ button. Then you have the option to either save or remove the selection. Pressing ‘Save’ will keep only the selection and remove the rest of the video, while pressing ‘Remove’ will remove the selection and keep the rest of the video.



Fig 7. Video trim controls.

**Loading Bar**

Any time you perform an edit such as adding a new video or trimming the video, a simple loading bar will pop up to let you know that the edit is processing. See Fig 8. Note that this loading bar is not directly connected to the progress of the process, but instead simply progresses while the process is running and either finishes when the process finishes or hangs at 99% if it reaches the end before the process finishes. In the future I will try to make the loading bar more accurate to the actual process.

A screenshot of a phone

Description automatically generated

Fig 8. Loading bar.

**Undoing and Redoing Changes**

Any time you make a change to your video that you want to revert you can either press ‘Edit’->’Undo’ or ‘Ctrl + Z’ to undo the edit. To redo an edit that you have undone, simply press ‘Edit’->’Redo’ or ‘Ctrl + Y’ to redo the edit. To keep track of previous edits to undo and redo, the program creates a new temp video file every time an edit is made and numbers them. This allows it to easily recall previous edits. Once the video has been discarded or the program closes, these temp files are deleted.

**Adding Audio**

The process of adding audio starts by pressing the ‘Add Audio’ button. This brings up the controls to add audio. See Fig 9. To choose the point where you want the audio to begin, press the ‘Start Point’ button. Then press the second ‘Add Audio’ button to bring up the window to choose an audio file. Only mp3 files are accepted by this window. In the future I plan to add the ability to accept more audio file formats than mp3 and an audio end point button since currently it adds in the entire audio file or until the video ends.



Fig 9. Add audio controls.

**Add Intro/Outro**

You have the option to add a simple intro or outro to your video. To do so press either the ‘Add Intro’ or ‘Add Outro’ button, the only difference being that it adds it to the beginning or end of the video. Pressing one of these buttons will bring up an intro/outro creator. See Fig 10. This creator allows you to change the text, font style and size, text color, background color, and duration in seconds. Any changes that you make are shown in real time in a preview on the right.

A screenshot of a computer

Description automatically generated

Fig 10. Intro/Outro creator.

**Taking a Screenshot**

Another feature that FlickFix provides is the ability to take a screenshot of a video. To do so, press the ‘Take Screenshot' button and choose a name and location for the screenshot. Pressing ‘Save’ saves a PNG of the still image from the video.

**Dark/Light Mode**

FlickFix has one theme customization option, the ability to change the user interface to Dark mode or back to Light mode. To do so, press ‘Options’->’Theme’->’Dark mode’ or ‘Light mode’. See Fig 11. This change updates all parts of the UI such as the loading bar and Intro/Outro creator and is persistent even after closing the application.

A screenshot of a video player

Description automatically generated

Fig 11. FlickFix after Dark mode is selected.

**The Tutorial**

There is a FlickFix tutorial available for people who would like to know how to use each feature of the application. To view it press ‘Help’->’Tutorial’. This brings up a separate tutorial window with a side menu bar of each feature. See Fig 12. Pressing the various features in the side menu brings up the tutorial for each one.

A screenshot of a video player

Description automatically generated

Fig 12. The FlickFix tutorial.

**Test Plan**

* <https://github.com/mtweigel/CSU-Senior-Project/blob/master/tests/FlickFix%20Test%20Plan.docx>

**Test Results**

* Test Cases: <https://docs.google.com/spreadsheets/d/1AKGEy2gAnmjuKqQM1qJmJLxnYHLELVpD3hXlkFgVaD0/edit?usp=sharing>
* Usability Testing Results:

<https://github.com/mtweigel/CSU-Senior-Project/blob/master/tests/FlickFix%20Usability%20Testing%20Results.docx>

**Challenges Overcome**

A lot of challenges were overcome in the creation of this project, most of which were during the development process. Learning how to use FFmpeg within my program to make video edits was a challenge to learn but was very rewarding once I figured it out since it is a very powerful tool for multimedia processing. Other challenges that I had overcome include figuring out how to add undo redo functionality and certain bugs like videos not loading or appending correctly.

**Future Enhancements**

* Let Seek Forward and Backward buttons increase in speed after being held down for a couple seconds.
* Add buttons to skip to beginning/end of video.
* Add ability to add a video between videos using a time stamp.
* Have intro/outro creator only edit selected text, not all.
* Add undo/redo buttons to intro/outro creator.
* Add audio end point button.
* Have lines mark start and end points.
* Add ability to mute portions or all of a video.
* Speed up dealing with larger video files.
* Fix bug in found in test case 23.

**Defense Presentation Slides**

* <https://docs.google.com/presentation/d/186tkq9QgpeAMfGjel42JKUZ0i1FOpHTc_bHdZwmv-48/edit?usp=sharing>