**Requirements Findings and Design for Accessible Video Editor**

# **Software Requirements Elicitation**

## Session Overview:

During our one-hour remote meeting on Teams, David Engebretson, a male individual with visual impairment and a key member involved in the development of dAVidE, provided valuable insights and feedback regarding usability, issue identification, design goals, and scenarios for wireframing.

## Organization of the Session:

The session followed the outlined agenda:

1. **Introduction** (5 minutes): Brief introduction to set context of the project.
2. **Usability Presentation** (45 minutes): A comprehensive discussion on usability issues and design goals.
3. **Identify Issues** (30 minutes): Discussion of existing issues with dAVidE, including limitations in editing capabilities, unreliable keyboard shortcuts, lack of undo functionality and overwriting of video files.
4. **Design Goals** (15 minutes): Exploration of desired improvements and features
5. **Scenarios & Wireframing** (15 minutes): Brainstorming and envisioning user scenarios to inform wireframe design.

## Questions and Answers:

* How can we make the wireframe more accessible for you?
  + David highlighted the importance of descriptive text next to each panel so he can properly understand the wireframe panel.
* How would you like the alerts to be changed?
  + David proposed a history log of user and system actions with hidden timestamps, and alerts specifically for mark 1 and mark 2 additions.
* Should the keyboard shortcuts option be on by default?
  + David recommended having keyboard shortcuts enabled by default but providing users with the option to disable them. He also mentions that the accessibility menu where the keyboard shortcuts checkbox is currently placed will be replaced with the accessibility options menu of Ashlar’s preheader.
* What changes do you want made in the video editor?
  + David emphasized the need for separate lines for the current time and mark 1 and 2 indicators, facilitating individual access to each piece of information.
* Would it be useful to have an alert when mark 1 is after mark 2? What about other errors?
  + David stated that he would like to have alerts for when there are errors but to keep them concise as the screen reader will read the alerts when they pop up which may interfere with what the user is currently doing.

## Tasks and Associated Scenarios:

1. Uploading a Video:

* David encountered audio clutter due to screen reader repetition and suggested minimizing redundant text.
* He emphasized the need for concise information display.

1. Trimming a Video:

* David navigated through the trimming process, providing feedback on ease of use and accessibility.

1. Looking at Alerts:
   * David explored the alerts feature, emphasizing the importance of clear presentation for easy navigation.
2. Playing a Selection:
   * Feedback was provided on the playback functionality, ensuring accessibility and user-friendliness.
3. Saving the Video:
   * David shared insights on the saving process, highlighting any accessibility challenges encountered.

## Additional Design Goals:

1. **Introduction Section:** Introduce an introductory section before accessing the main video editor interface. This section can provide brief instructions, tips, and accessibility features overview to orient users, especially those with visual impairments, before they begin editing.
2. **Separate Video Timeline Region:** Implement a dedicated and easily accessible region for the video timeline within the interface. This separate area allows users, including David, to focus specifically on editing the video timeline without distraction from other interface elements.
3. **Enhanced Download Feature:** Revise the download feature to prompt users for the destination location where the video should be downloaded. This customization option ensures flexibility and convenience for users, including David, who may have specific preferences or accessibility requirements for storing downloaded videos.
4. **Video Concatenation Capability:** Integrate video concatenation functionality to enable users to seamlessly merge multiple video files into a single cohesive video project. This feature enhances the editing versatility of AVidE, catering to the needs of users like David who may require combining various video segments for accessibility or content creation purposes.

# **Conclusion of the Requirements Phase**

In our requirements elicitation phase, we worked with David Engrebetson to outline key features and functionalities for the projects. In this section we will consider their priority levels and discuss the implementation timeline.

**Requirements:**

1. **Fixing Existing Bugs:** Addressing existing bugs and issues is crucial for maintaining the stability and usability of the system. This requirement is of high priority and will be continuously addressed throughout the development process, with specific focus in R1 and R2 to ensure a reliable user experience.
2. **Introduction Section:** This introductory section is crucial for user orientation, especially for those with visual impairments. It provides brief instructions, tips, and an overview of accessibility features. Given its importance for user guidance, it holds a high priority and will be implemented in R1.
3. **Log of announcements/alerts:** This feature holds a high priority as it directly impacts user experience by enabling users to track past announcements and alerts efficiently. Given its significance, it is scheduled for implementation in R1.
4. **Option to save alerts:** This relates to the alerts log and also allows users to archive and analyze past alerts. This will also be included in R1.
5. **Styling the web app using Ashlar:** Making the web app visually appealing is important for user engagement. Matching the styling of the app with the rest of Western’s pages will allow seamless integration with the rest of Western’s web pages. This will be implemented in R1.
6. **Keyboard shortcuts:** Recognizing the importance of accessibility and user efficiency, this feature is deemed high priority. Its implementation in R1 aligns with our commitment to usability and ease of use.
7. **Video Concatenation Capability:** Integrating video concatenation functionality enriches the editing versatility of AVidE, catering to users' needs, including David's. While significant, it's rated medium priority due to its complexity and is scheduled for implementation in R2.
8. **Enhanced Download Feature:** Revising the download feature to allow users to specify the destination location enhances flexibility and convenience, especially for users with specific accessibility requirements. This customization option is of medium priority and will be included in R2.

We aim to deliver a robust system that meets user expectations while maintaining a manageable development pace. Our decisions are informed by insights gathered from PD sessions and user feedback, ensuring alignment with project goals and stakeholder expectations. Additionally, addressing existing bugs ensures a smooth user experience and builds trust in the reliability of the system.

# Proposed Design (Wireframe)

## Abstract

The first wireframe depicts the general layout of the web application page for the Accessible Video Editor. The page will utilize Western’s theme and component library Ashlar and its common pattern organisms.

From the top down the page will feature the pre-header with its familiar tools, skip to content, display settings, and search bar. The display settings menu will take full advantage of Ashlar’s accessibility features and eliminate the need for a separate accessibility section within the content of the application.

The Western header will follow which will include the default Western logo and title of the application, “Accessible Video Editor.” Below Western header will be the main navigation containing the “About” and “GitHub” from the original web app.

Below this will be the “content” and will include a redesigned and updated editor that features two inline sections, the video editor section on the left that contains the editor and its related interactions, and an alert and history section. This redesigned text box will now include not just the current alert message, but an in-order history that contains descriptions of page alerts, directions, and actions taken by the user. It is followed beneath by a button that reads, “download history” so that a transcript of what has transpired so far within the editor can be saved and reviewed later.

Below the content section will be Western’s site footer, possibly with contact information or related links and the final section will be the Western footer.

Note that all wireframe depictions include these headers and footers from Ashlar as well as the alert and history section and therefore all following wireframe depictions will assume their inclusion and may describe only the content section that differs from one wireframe to another.

## Choose File

The content of Choose File wireframe is the landing page for a new project. It feels and operates very similarly to the original web app. The main content of the page includes the editor section on the left and the alert/history section on the right. The editor section on the left has the title Upload Video, and underneath has text that reads, “Select an MP4 video:” followed by a button that reads “choose file” and ends with text that reads, “no file selected.” This text on the end is the default message when no file has yet been selected from the file explorer. Below this section is another button that reads, “Upload” that is disabled while no file is currently selected. To the right of the editor is the alert and history section described within the abstract.

## Upload File

After selecting a file, the page changes subtly, working the same way as the original web app. Within the main content of the page, the editor reflects the file selection by replacing the “no file selected” text with “… selected” where the ellipsis is the name of the file selected. In the wireframe example the text reads, “MyVideo.mp4 selected.” The upload button below has also changed to be enabled and clickable and now reads, “Upload MyVideo.mp4.” The alerts and history section remains unchanged in layout but will include the most recent alerts and actions in addition to the previous ones.

## Waiting

Once a file has been selected and the upload button has been pressed, there can be a noticeable hang in the app operation while the file is loaded into the app. This next wireframe called waiting reflects our intention to add some sort of notification to the user that this is a) happening and b) a normal process. The main content of the page now features a blurred version of the editor from the upload file wireframe with a text box overlay that reads “please wait” with an abstract gear symbol. The Alerts and history section also posts an alert notification to notify the user that reads, “waiting on load.” This way it is clear that the page has not necessarily hung altogether, and the user knows to wait.

## File Loaded

Now that a file is loaded the editor section reflects this change by showing a new title of the page which now reads, “Video: MyVideo.mp4,” or whatever the name of the selected video is, and under the title the video editor preview box. To the right of the video preview box is the alerts and history section that has all the same functionality as described previously and although specific messages are not drawn the most recent message will reflect that the file has successfully loaded. As the content within the page is more complex, it was necessary for the content of this part of the app to be spread over several wireframes so that the drawing layout looks proportionally correct. The wireframe drawings depict the page as it is scrolled downward from the top in a few steps. The controls for the video and their details are described in the subsequent wireframes.

## Editor Layout

The editor is laid out as follows: to the left of the page in the editor section is the video editor preview box and its associated controls below it. The video preview box shows an illustrated version of the moviepy library’s new preview box that provides more control over the version currently used. It includes its own play button, timeline bar, and larger current location pointer that includes the time at the current position, volume controls and an option for full screen. The editor buttons below include the current time the video is at against the total length of the video, the location of the first mark in time, the location of the second mark in time, the player navigation control buttons, and the marking and editing toolset buttons. To the right is the alerts and history pane that shows indecipherable lines that represent the accumulated history of the project that can be reviewed and its associated download button.

## Editing Controls

There are several editing controls, most of which will be familiar to the original app. As a note, any reference to something listed as a position in time will be displayed in minutes and seconds and accurate to the nearest hundredth of a second.

It will first have the current position in time of the video against the total time of the video. It is then followed by the time of mark 1 and the time of mark 2. Next are player navigation controls which include a play button and additional buttons to move -10s, -1s, +1s, and +10s respectively. Below these are the add mark 1, add mark 2, play selection, trim selection, and delete selection buttons that make up the editing tool controls.

In addition to these we would also like to add additional functionality to the editor in the way of adding additional videos to the project that can be stitched together from the beginning or to the end. The wireframe depicts this as a section similar to the select file functionality described in the select file wireframe with a heading that reads, “Add video to project,” the familiar select video layout and mechanic with the notable exception that there are now two buttons to upload the file. One of these buttons reads, “Upload to beginning” and the other “Upload to end.” This would provide the user with greatly improved functionality to create a single video from several others, instead of only being able to edit a single video.