Accessibility Tools Software Design Document

Requirements / Overview

- Description of MVP

For the minimum viable product users must be able to modify the elements of a HTML webpage to make it more accessible, suited to their individual needs and to the WCAG.
In order to follow the minimum level of WCAG, the software must be able to:

Read out text with text to speech.
Change the font size, face, colour, line height and character spacing of a section of text.
Translate individual words, passages or webpages from one language to another.
Search up words for their definition, IPA, used in a sentence with synonyms and antonyms.
Change the background or colour of the page background, can also be extended to modify the overall contrast of the web page.
Enable a magnifying glass to zoom in on elements/a section of the page.
Provide alt text for images through the use of an image classifier.

The software should also include a settings button that can be used to edit and save preferences, along with a user manual to inform users on how the software functions.

- Intended audience for the project

The intended audience for this projects comprises mainly of elderly internet users who struggle with website navigation or web content as a whole, and anybody else with any impairment that may make using the internet more difficult.

- What is the success criterium for this project?

For this project to be successful, it should provide free control over a webpage as mentioned before, while following a minimum of level A in the WCAG where applicable and possible.

- What are possible failure scenarios and conditions?

One of the key features that may lead to failure is the communication between the front-end and the back-end, along with the efficacy of the back-end application of the image classifier. If images cannot be successfully sent to the back-end for classification, or the server takes too long to respond, one of the key features of the software will be useless.

- Contingency measures for failure scenarios

For the scenario in which the classifier is ineffective, slow, or completely unresponsive, some backup features may be implemented (depending on scope). Most importantly, users should be informed when the back-end is unresponsive through an on-screen message. If an image has already been provided alt-text by the classifier, it could be stored in a database to prevent it from being re-classified (though this is unlikely to scale well). A feature may be added where a user can provide alt-text for an image instead of the classifier, and any existing alt-text that may be difficult to find but has already been provided can be more easily displayed by the software.

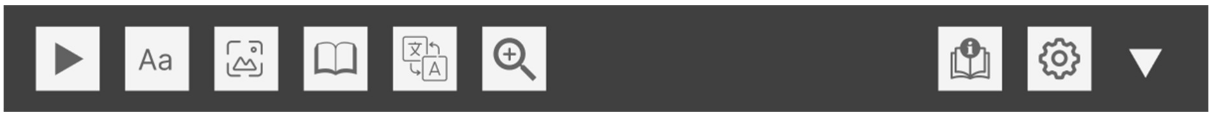
- Functional Requirements (dependencies, integrations with applications)

The software will be presented in the form of a browser extension. For the MVP, it will be developed for Google Chrome and shouldn't require any pre-requisites to be used from download. Porting the software to other popular browsers is unlikely to be a difficult task, so it can easily be integrated into any desktop browser.

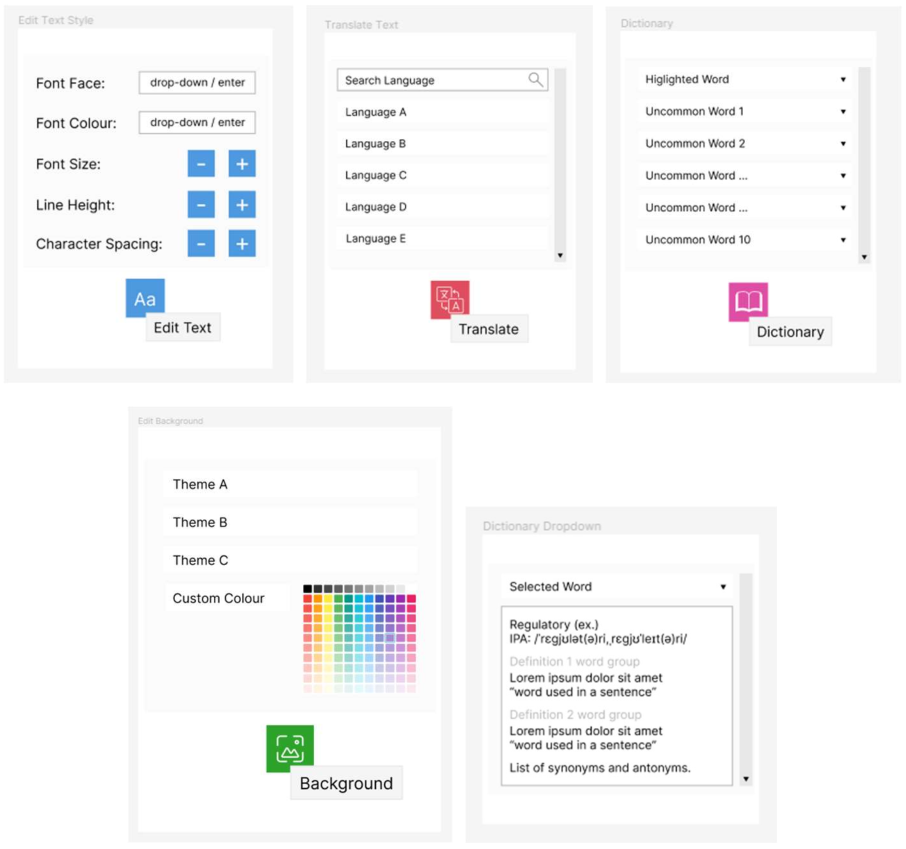
User Interface

- *How will the software be presented?*

The software will be presented as a toolbar at the bottom of any browser page. This toolbar can be hidden with a button and shown again with the same button.

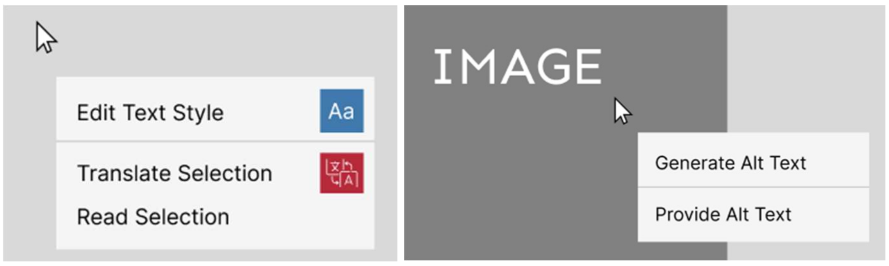


A demonstration of the toolbar is shown above. Icons denote functionality but will be redesigned.



Above is a demonstration of the popup that may appear after clicking the toolbar buttons. For the play button, the selected portion of text will be read out, and if text is not selected, the page body will be read from start to finish. Because of this, nothing would appear if the button were pressed.

Features may also be implemented by right clicking screen elements.



For alt text generation, this menu may be changed to automatically appear over any hovered image, given the toolbar is active. This would make navigating the software a little easier.

As for the user manual and settings, these will be fleshed out further given later requirements of the software and will be used to set preferences for smaller features in the software, such as playback speed for audio.

Once a button has been pressed on the toolbar, the corresponding menu will appear directly above it, and the shade of the button will change, or an outline will be displayed to demonstrate the button is currently in use. This change will also occur when a button is being hovered over.

In order to handle dimensional constraints, the toolbar will either scale with the window, or become thinner so the buttons move closer to eachother. All modern dimensional constraints should be considered when developing the toolbar, along with any dimensional constraint available on Google's dimension list on Chrome (can be accessed through F12).

At this stage, errors on the front-end are hard to predict, though some optimization and potential bugs can be foreseen. If the software is running slowly, optimizations need to be made, and any issues with dimensional constraints as mentioned before need to be handled to ensure correct display of the UI.

If internet access is lost, the toolbar's offline features (most of the program) should still function without any problem.

Any buttons that are pressed should activate and deactivate any popup windows and features without problems, and should also be easily accessibile (it isn't a maze to disable a window).

Text to speech should work with all languages available on the translation list – if it doesn't work with a specific language it should be noted on the translation list (this could arise from text to speech bots struggling with non-latin alphabets).

All translations should succeed without problems and not hinder the UI of the page by too much. This also applies to text resizing, though systems may be put into place to ensure resizing text doesn't break the functionality of the website (at least without prior warning).

Dictionary entries should be fully functional and account for all definitions and pronunciations of a word (in case of homographs). It should also account for all word forms. For example, the plural form of a word should simply return the definition of the non-pluralized form of the word.

If themes are correctly implemented, any default packages that may be developed should take into account the WCAG contrast ratio, and all other features should demonstrate full WCAG compliance where applicable.

Milestones and Technical Breakdown

- Front-end milestones

The main milestones for the front-end are listed below in order of ease and priority.

Must Have:

- Implementing the element for the toolbar so it displays correctly on every page, this includes place-holders for buttons (even if not yet functional).
- All UI successfully displays when buttons are pressed, even if as place-holders.
- Implementation of text modification (font size, line height) for full page use.
- Implementation of text modification (font size, line height) for sectional use.
- Dictionary feature successfully returns definition of word from selection, and from the list of uncommon words on the webpage.
- Implementation of screen reader (text to speech) for most if not all included languages.

Implementation of translation for full page use.
Implementation of translation for sectional use.
Implementation of background/theme changer for the whole web page.
Implementation of background changer for the background only.
Implementation of the magnifying glass.
Implementation of the user manual.
A simple, easy to navigate UI for any other features (e.g. alt text generation and display).
Complete WCAG compliance (e.g. alt text for all buttons).

Should Have:
A wide list of preference settings that can help the user modify the toolbar to better suit their needs.
Buttons that change colour/have something to signify when hovered or pressed.
Multiple methods to access controls, for example through keyboard hotkeys.
Preferences for text, theme, and language characteristics.

Could Have (scope dependant):
Extra controls for videos for automatic captioning.
Options for language simplification.
Extra menu for the extension when icon is clicked.

- Client-side back-end milestones
The main milestones for the client-side back-end are listed below in order of ease and priority.

Must Have:
Manifest and extension icon complete with sufficient information in the manifest.
Elements are grabbed from the page to modify them for later use.
Images are successfully sent to back-end for classification.
Classification data is successfully received and displayed on the page.
Selected words are defined through either the use of a dictionary API or a database.
Page is successfully updated and restored through sessions when changes are made.

No foreseen should have/could have features.

- Server-side back-end milestones
The main milestones for the server-side back-end are listed below in order of ease and priority.

Must Have:
Implementation of an image classifier for training on a dataset.
Fully trained image classifier that successfully returns text for classified image within a reasonable time frame.
If no API for the dictionary, a comprehensive list of dictionary data for every supported language.
Full implementation of all features displayed on the front-end that require back-end support.

Should Have:
Confidence levels for objects classified in images to be sent to the front-end.

Could Have (scope dependant):
Implementation of natural language processing AI.
Automatic captioning of videos using AI.
Database for storing generated/inputted alt text for images.

- *Project Scope*

Depending on development speed, some features from the should have and could have list may or may not be implemented. Since the list is priority based, everything in the “must have” list must be implemented for the minimum viable product. Given the time frame of development, the must have features are guaranteed to have been developed by the deadline, where some of the should have features have been considered or already implemented.

- *Other concerns*

The existence of the back-end (due to implementation of AI) means that none of these features can be ran for free. Because of this, the software will be left entirely open source (on GitHub) and any features that require back-end use will be used only for demonstration purposes. This software is to help those with impairments, and is not a business model, so if a “free” version (front-end only) of the software is possible – it can be released.

A name for this software is currently undecided.