Developer Documentation

Overview:

Accessibility Checker is a web-based tool for testing one page at a time accessibility against WCAG standards for contrast, typography, alt text, headings, and buttons.

Condensed Version of Final Spec:

All major components outlined in the initial project spec plan were successfully implemented. The user inputs a URL, which is validated and passed. A page loader extracts HTML content, computed styles, and elements metadata. Then the tool runs five core accessibility checks: contrast, typography, alt text, heading structure, and button/link labeling. This returns the results as structured DataFrames for display through a streamlit interface.

General Design Notes:

- No classes or inheritance are used.
- Each accessibility check is implemented as a standalone function in its own module.

main.py

This file controls the layout and interaction of the Accessibility Checker using streamlit. It runs from top to bottom without custom functions or classes, handling user input and showing the results.

Flow Summary:

1. Open the terminal make sure you are in your virtual environment and run: streamlit run main.py
2. User enters a URL and clicks “Run Accessibility Checker”
3. The app:
   - Validates the URL
   - Checks reachability
   - Loads page content
4. Extracted content is passed through the checker functions.
5. It is then displayed in table format with tabs.
6. User can download content as a CSV file.

Module: page_loader.py

This module is responsible for validating and fetching the target web page content before any accessibility checks are performed. It contains three key functions:

Function #1: valid_url(url)

Checks whether the input URL begins with http:// or https:// using a regular expression. It ensures the input format is correct before proceeding. Returns True if the URL is valid, otherwise False.

Function #2: is_reachable(url)

Uses the requests library to determine if the site is reachable and responsive. Returns True if the HTTP status code is below 400, otherwise False.

Function #3: load_page_data(url, viewport = None)

Uses Playwright (via sync_playwright) to launch a Chromium browser, navigate to the page, and extract relevant content for analysis. This function returns a dictionary with specific elements for the accessibility checker.

Module: accessibility_checker.py

This module is responsible for analyzing the data extracted and returning structured results for accessibility compliance. Each function operates independently and returns a list of results.

Function #1: check_contrast(texts)

Converts foreground/background color strings into normalized RGB values. Returns a list of dictionaries with tag, ID, class, a short text snippet, contrast ratio, and pass/fail status.

Helper Function: parse_px(text_size) → Strips "px" suffix from font size strings and converts to integers.

Dependencies: wcag_contrast_ratio

- Uses the rgb() function from wcag_contrast_ratio to compute contrast ratios.
- Applies passes_AA() and passes_AAA() to determine WCAG compliance.

Function #2: check_typography(texts)

Extracts font size and categorizes it as passing, warning or fail.

Helper Function: parse_px(text_size) → Strips "px" suffix from font size strings and converts to integers.

Function #3: check_alt_text(images)

Identifies images with missing alt tags. Returns the image src, alt text, and a Pass or Fail status.

Function #4: check_heading_structure(headings)

Checks for the presence of exactly one H1 and detects skipped heading levels.

Function #5: check_link_buttons(elements)

Detects missing or unclear labels for buttons/links. Returns a list of dictionaries with tag, ID, short text snippets, reason, and pass/fail status.

Module: results_formatter.py

This module centralizes all formatting logic for the results produced by the accessibility checkers. It converts raw outputs into structured pandas.DataFrame objects for use in the streamlit main.py and prepares grouped JSON-style outputs for potential export.

Category	Functions Include	Output Type	Use Case
Contrast	build_contrast_df, build_contrast_pass_df, get_contrast_json	DataFrame/JSON	Filter by AA/AAA results. Pass or Fail.
Typography	build_typography_df, build_typography_fail_df, get_typography_json	DataFrame/JSON	Categorized by Pass/Warning/Fail
Alt-Text	build_alt_text_df, build_alt_text_fail_df, get_alt_text_json	DataFrame/JSON	Image descriptive alt text. Pass or Fail.
Headings	build_heading_df, get_heading_json	DataFrame	Shows H1 issues and level skips
Buttons	build_button_df, build_button_fail_df, get_button_json	DataFrame/JSON	Flags missing labels for links/buttons

Known Issues:

- Occasionally, if the page takes too long to load or Playwright times out, the app may show an error. It is designed to prompt the user to click the Run Accessibility Checker button again, which usually resolves the issue. In some cases, the message may not appear, retrying the button manually is still the recommended fix.
- The app must be run inside a virtual environment to ensure all dependencies are correctly isolated and the project runs smoothly without version conflicts.

Future Work:

To continue expanding this project, we can add more WCAG checks. Future updates could include support for keyboard navigation and resizing consistency.