Technical specification for the feature: Testing the whole website

This is the technical specification for the Testing the whole website feature. The feature was created as part of an EU-funded project called We4Authors Cluster. All features, their technical specification and their video documentation are published on www.accessibilitycluster.com.

If you have any questions or comments, please do not hesitate to reach out to research@funka.com.

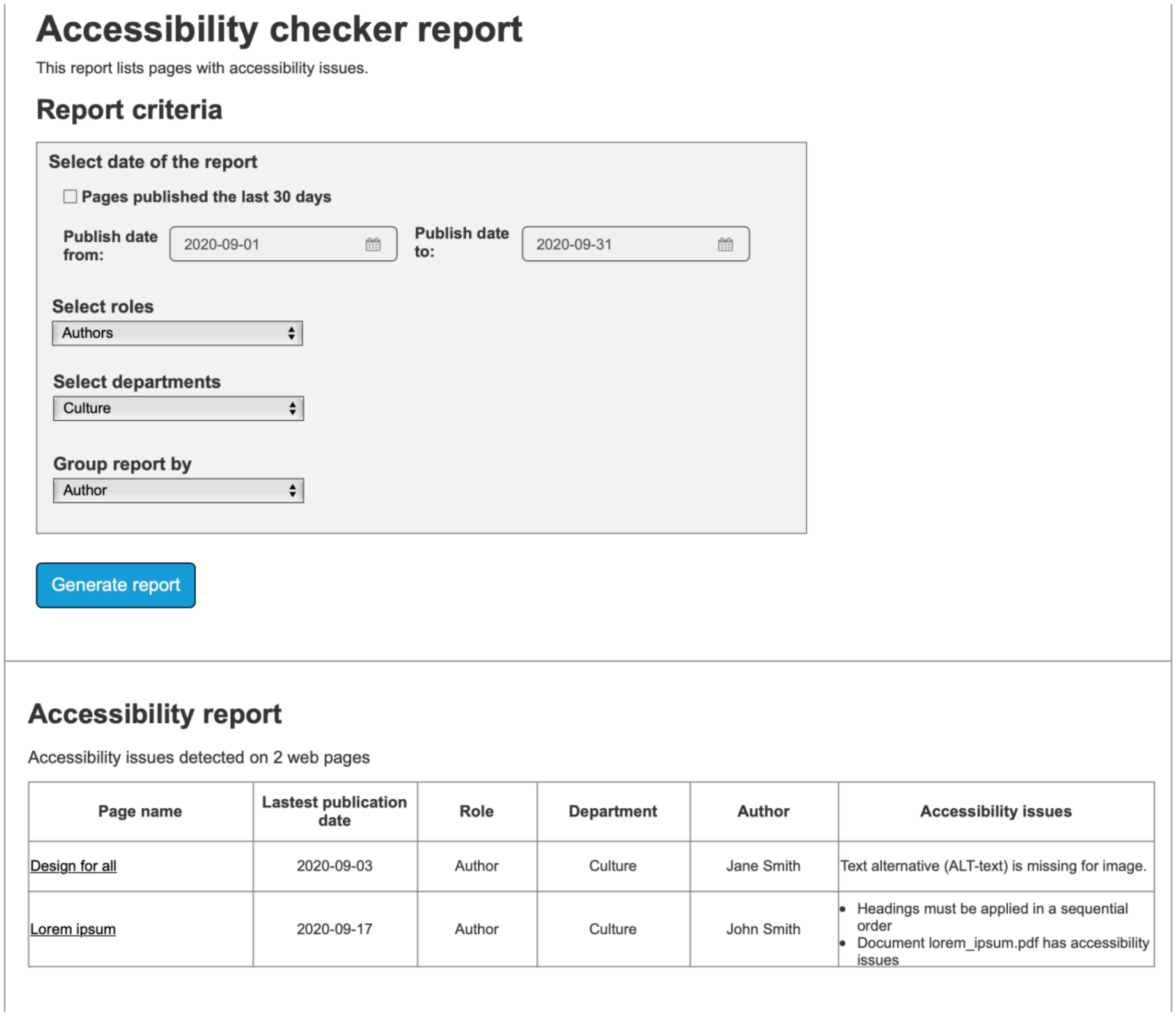
The technical specification contains several perspectives:

- the web interface of the feature, based on the experience of the end user or, when it comes to the testing-features, the web author;
- the code of the feature ensuring accessibility by default;
- a description highlighting the key elements of the code;
- reference to a video documentation;
- recommendations for implementation.

Specifications of the feature: Testing the whole website

An accessibility test of the whole website is valuable to the website owner for compliance reasons, but also to be able to detect any specific departments or kind of content that needs refinement or groups of staff who need training.

Web interface: web author view



Accessibility checker report

This report lists pages with accessibility issues.

Report criteria

Select date of the report

☐ Pages published the last 30 days

Publish date from:	2020-09-01	Publish date to:	2020-09-31

Select roles: Authors

Select departments: Culture

Group report by: Author

Generate report

Accessibility report

Accessibility issues detected on 2 web pages

Page name	Lastest publication date	Role	Department	Author	Accessibility issues
Design for all	2020-09-03	Author	Culture	Jane Smith	Text alternative (ALT-text) is missing for image.
Lorem ipsum	2020-09-17	Author	Culture	John Smith	• Headings must be applied in a sequential order • Document lorem_ipsum.pdf has accessibility issues

Figure 1: Web author view - How to generate an accessibility report and the report generated

Possible built-in success criteria conformance tests to be done

- Using <h1>-<h6> to identify headings.
- Providing heading elements at the beginning of each section of content.
- Sequential headings.
- Adjacent links.
- Using <ol>, <ul> and <dl> for lists or groups of links.
- Providing short text alternatives that provide a brief description of the non-text content or role="presentation" attribute.
- Using <caption> elements to associate data table captions with data tables.

- Using table markup to present tabular information, <th> and <td>.
- Using the scope attribute to associate header cells and data cells in data tables.
- Parsing, in content implemented, elements have complete start and end tags, elements are nested according to their specifications, elements do not contain duplicate attributes, and any IDs are unique.

CivicActions maintains a list of open source tools for evaluating page-level accessibility:
https://accessibility.civicactions.com/guide/tools#page-level-evaluation

Tools like Sa11y and Editoria11y allow web authors to focus on what they can control. Most accessibility errors are introduced in the editor's body field, so it is possible to target the WYSIWYG to that field and generate accessibility errors that target that section of the page specifically.

Sa11y:
https://ryersondmp.github.io/sa11y/#install

Editoria11y:
https://itmaybejj.github.io/editoria11y/

Editoria11y needs to have JavaScript added to the HTML headers for the page:

```
<script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
<link rel="stylesheet" media="screen" href="/css/editoria11y.css">
<script src="/js/editoria11y-prefs.js"></script>
<script src="/js/editoria11y-localization.js"></script>
<script src="/js/editoria11y.js"></script>
```

There is also a Drupal module:
https://www.drupal.org/project/editoria11y

Guide to implementation

The authoring tool does a request to the 3rd party service (the testing tool that usually includes some authorisation information (that is stored in the authoring tool itself), because the third-party service is usually not open to the public and may include usage fees. The 3rd party service then responds with results of the check (either HTML or JSON/XML via a REST API or something similar), which is embedded in the authoring tool view. The embedding often happens at the browser level (with JavaScript) because this is the most lightweight and flexible method of embedding external content from a 3rd party service.

Video documentation

This technical specification has been developed in the We4Author Cluster project to reflect recommendations for accessibility features that can be implemented in any authoring tool. The specifications are complemented by a video documentation, covering a live description of:

- web author challenges, and
- feature solutions.

Recommendations for implementation

To make sure the implementation of the features is not causing accessibility problems for web authors with disabilities:

- avoid drag-and-drop for choosing template, and/or always have a keyboard alternative;
- always provide one pointer alternative without specific gestures;
- always support keyboard navigation;
- consider keyboard shortcuts;
- do not rely on sensory characteristics as the sole indicator for understanding and operating content;
- do not indicate important information using colour alone;
- make sure there is enough contrast between text objects and its background colour.