Guide for Accessibility Management

# Accessibility: Who is Responsible for It?

In short, everyone within the organization has a responsibility and has an impact on integrating accessibility. The onus is not just on designers, developers, and quality assurance testers.

To start incorporating accessibility, it is critical to review the organizational structure and determine where accessibility is needed, determine the responsibilities of job roles, and how progress toward accessibility will be reported.

The function of core accessibility team, or centralized accessibility group, is to aid, provide guidance, and establish positive working relationships with teams that promote accessibility integration within product teams.

## Where to Begin?

Essentially, accessibility must be part of the software development lifecycle for it to be integrated and maintained successfully. It will be critical to assess product teams’ accessibility knowledge and determine what skills and education are needed for each team role. If hiring for new roles, knowledge of accessibility should be included in job descriptions and assessed during the interview process.

### High-level Overview of Team Responsibilities

* **Product Owner**: The product owner oversees accessibility compliance not only from a compliance perspective, but in a manner that prioritizes users with disabilities. The product owner defines requirements based on user stories and specific WCAG success criteria to which the product must conform.
* **Visual and Interaction Designers**: Visual designers need to ensure that their designs can be perceived and understand by a wide variety of users. Interaction designers need to understand how people with disabilities and their assistive technologies are used to browse and interact with their design. They need to provide guidance to developers as to what information should be conveyed to assistive technologies.
* **Content Authors**: Content authors need to write and produce content in a manner that can be easily understood by a wide variety of users. They should be creating content that is written in plain language, uses appropriate headings, describing images when necessary, structing tables with proper headers, and composing accurate link labels.
* **Developers**: Developers need to ensure that their code functions in an accessible manner according to the designs provided to them. They need to frequently test their code for accessibility with automation tools to catch errors as they build.
* **Testers**: Testers need to ensure that the product satisfies the accessibility requirements defined for the product. They conduct automated testing as well as manual testing with various assistive technologies. If issues are found, they report issues along with actionable accessibility fixes.

## Establish Checkpoints for Accessibility

If there are existing processes and systems in place, investigate how accessibility can be embedded in them. For instance, if there are guidelines for style and branding, specify accessibility requirements for those guidelines.

Here are some checkpoints to consider for accessibility:

* Check that accessibility is included in design specifications and functional design documentation.
* Evaluate the information architecture for accessibility.
* Ensure user acceptance criteria includes accessibility requirements.

Here is how accessibility should generally be addressed at each phase:

1. Plan and Analyze:
	1. Identify that WCAG 2.2 AA conformance is the primary goal for the product.
	2. Define the “Definition of Done” for accessibility (Which WCAG success criteria and assistive technologies the product will conform to and support? This needs to be specific and measurable.
	3. Compose user stories about people with disabilities. Specify what these users will expect while using the product.
	4. Plan and budget for real user testing for accessibility.
2. Design: Utilize user stories and requirements as a guide and embed these requirements in design specifications.
3. Build: Continuously evaluate accessibility compliance for parts of the product, and ensure code supports accessibility programmatically. Create accessible code libraries to reuse accessible components.
4. User Testing: Test using various assistive technologies, including screen readers, keyboard-only testing, and screen magnification. If possible, include real users in testing as soon as functional prototypes are available.
5. Evaluate: Test against accessibility requirements (WCAG success criteria) and report issues.
6. Regression and Monitoring: When fixes are implemented, monitor them over time as well as any new code changes that may impact accessibility.

### User Stories for Accessibility

The overall goal for the product is for people with disabilities to be able to use the product effectively. With this goal in mind, user stories that describe what people with disabilities expect should be used to guide development teams in accessibility implementation.

*As a user who has a disability, I want to be able to read content, browse content, and complete all actions so that I can have an equivalent experience as a user who does not have a disability.*

### Example User Stories

*As a screen reader user, I want headings to be descriptive and logical so that I can easily understand and browse the information on the page.*

*As a user who is deafblind, I want correct and accurate transcripts so that I can have equivalent access to audio and video information.*

*As a user who has photosensitivity, I would like the option to turn off moving and animated content to reduce the chance of triggering a seizure.*

*As a screen magnification user, I would like form labels to be near text fields and controls so that I can find what entry information is needed.*

*As a user who is colorblind, I would like for links and buttons to be distinguishable from other types of information on a page so that I easily perceive them and interact with them.*

*As a user who relies on voice input technology, I would like to use the visual labels I see on screen so that I can click interactive elements with my voice.*

*As a user with difficulty recalling information, I would like to use the autofill features within a form so that I can complete the form more efficiently.*

User stories help reframe accessibility requirements for teams to think about and consider how actual users, how real people, are impacted by inaccessibility. Teams can take it a step further and create user personas to help them think about people, and not just the technical requirements for accessibility. In fact, when user stories and personas are utilized regularly, the technical requirements for accessibility take care of themselves.

# Important Accessibility Features and Requirements

Even though user stories can assist with defining and reinforcing, it is important to name and define the requirements themselves so that it is clear what is expected of the product’s accessibility.

## Accessibility Features

### Multimedia

* Video-only that convey important content should be accompanied by an audio description track or transcript that describes the visual content.
* Video and audio content together should have captions that present important speech and describe non-verbal sounds, as close to verbatim as much as possible, if visual descriptions aren’t part of the original audio track, an audio description track should be added. Pearson also requires a transcript for video/audio content for users who are deafblind.
* Audio-only content should have a transcript that present important speech and describe non-verbal sounds, as close to verbatim as much as possible.

### Animated Content

* Content that blinks or flashes three times in one second should be avoided.
* A mechanism to pause or stop content should be provided for automated content that moves for longer than five seconds.

### Audio Control

* A mechanism should be provided to pause, stop, or control the volume of audio that automatically plays for three seconds or longer.

### Text Resize and Reflow

* Text can be resized up to 200% without assistive technology and without loss of content or functionality.
* Page design can be resized up to 400% without assistive technology and without loss of content or functionality.

### Keyboard Access

* Any interaction that can be done through a mouse should also be performed through keyboard.
* A visual focus indicator, or focus ring, should be available when tabbing through the product.

### Dragging

* For any interaction that involves dragging a mouse pointer, ensure that the same interaction can be done with one click of the mouse

### Page Titles

* Web-based product pages should have unique page titles that describe the topic or purpose of the page.

### Bypass Blocks

* A method to bypass repeated blocks of content should be provided for keyboard-only and screen reader users.
	+ A Skip to Main Content should be provided for sighted keyboard-only users.
	+ Landmarks and headings should be provided for screen reader users to bypass content.

### Navigation

* There should be multiple ways to find content within a product, including:
	+ Search
	+ Main navigation
	+ Sitemap
	+ Quick links

### Timeouts

* Users should be notified of time limits, and a mechanism should be provided for the user to either:
	+ Turn off the time limit before encountering the time limit.
	+ Adjust the time limit before encountering. Users should be able to adjust the limit up to at least 10 times the default setting.
	+ Extend the time limit. Users should be given at least 20 second to extend the time limit. Users should be allowed to extend the time limit at least 10 times.

### Form Error Prevention

* Errors should be indicated visually and programmatically for assistive technologies. Feedback should tell users how to correct mistakes.
* For legal and financial transactions:
	+ A method should be provided to either check or reverse the submission, or
	+ A method should be provided to review, correct, and confirm submission before submitting it.

## Accessibility Requirements

### Images

* All images require a text alternative:
	+ For decorative images or images described by adjacent text, an empty text alternative can be used. A screen reader ignores images with empty text alternatives (alt=""), which is allowed for decorative and redundant images.
	+ For informative images, a text alternative that presents equivalent information conveyed visually is needed.
	+ For complex images:
		- The text alternative should summarize the information.
		- A longer description that presents the detailed information should be provided either as text near the image or generated dynamically (i.e., a dialog, a disclosure region).

### Semantic Structure

* Actual headings defined in the code are used for titles and subtitles.
* Headings should be logical and not skip levels.
* Every page should at least have one heading level 1 on the page.
* Lists are defined in the underlying code.
* Data tables have table headers defined and associated properly to data cells.
* Tables used for layout are avoided.

### Custom Widgets/Component

* For custom widgets:
	+ All interactive controls must have programmatic labels (what is known as the accessible name).
	+ The type of element must be defined in the code (link, button, tab, menuitem, dialog, combobox, etc.).
	+ Properties and states of controls must also be defined in the underlying code, for example:

“About us tab 1 of 4 selected”

* + - “About us” is the accessible name
		- Tab is the type of element, or role
		- 1 of 4 tells users they’re in a set list of tabs (property)
		- selected is the state of the element. It tells users which tab they have selected
	+ All custom widgets must utilize appropriate keyboard patterns according to W3C ARIA Authoring Practices whenever possible.
	+ For custom interactions, instructions are provided that tell users how to use the widget.

### Forms

* All forms should have visual and programmatic labels.
* Form fields and controls related to each other should be grouped together in the underlying code with a visual and programmatic group label.
* Required fields should be denoted visually and as such in the code.
* Instructions and data entry information should be associated visually and programmatically with their corresponding form element.
* Successful submission messages as well as error messages should be conveyed visually and programmatically.

### Target Size

* Ensure that target sizes for clickable/actionable elements are at least 24 px, even if the target itself is under 24 px