Shift-Left Accessibility for UX/UI: A Figma Preflight versus Code-Time Checks (Pilot)

Noelynn Faith Batalingaya

October 15, 2025

1 Introduction

Accessibility in technology is vital because digital tools shape nearly every part of modern life. People use websites and applications to study, work, receive healthcare, shop, and connect with others. When these systems are not accessible, individuals with disabilities are excluded from opportunities that others enjoy. Accessibility is not just an additional feature; it is essential for fairness and inclusion.

The lack of accessibility has real-world consequences. A student who relies on a screen reader may be unable to complete an online exam. A worker who cannot use a mouse may struggle to submit a job application. A person with low vision may not be able to read important health information if the text contrast is too low. These examples show that inaccessible design blocks access to education, employment, and essential services. This is why international standards such as the Web Content Accessibility Guidelines (WCAG) were created. These guidelines serve as a global framework to make sure that digital products work for everyone.

Even with WCAG, many digital products remain inaccessible. Studies show that most accessibility issues start early in the design process. Designers may pick colors with poor contrast, forget to add alternative text to images, or create layouts that are hard to navigate. Once these choices are built into a design system, they carry over to the entire codebase. When developers or testers finally find these issues, fixing them becomes slow, frustrating, and expensive. Many teams want to make their designs inclusive but do not know when or how to start.

This project uses a different approach by moving accessibility testing to the beginning of the design process. It applies a method from software engineering

called shift-left testing, which means testing earlier in the workflow when changes are cheaper and faster to make. Instead of waiting for code-time tools like Lighthouse or axe-core to find problems, this study develops a Figma plugin that performs accessibility "preflight" checks while designs are still being created. The plugin highlights problems such as poor color contrast or missing text alternatives and provides suggestions before any coding begins. The central research question guiding this project is: can early design checks reduce accessibility barriers later and help create more inclusive technology?

2 Background

Accessibility means making digital products usable by people with a variety of abilities. Some rely on screen readers, others on captions, larger buttons, or voice commands. Accessibility ensures that technology supports different ways of interacting. The Web Content Accessibility Guidelines organize accessibility principles into four main ideas: perceivable, operable, understandable, and robust.

- **Perceivable:** Content should be presented in ways that users can see or hear. For example, images need alternative text, videos should include captions, and text should have enough color contrast.
- **Operable:** Users must be able to interact with the system through different input methods like keyboards, voice control, or assistive technologies. Interfaces should not depend only on mouse actions.
- **Understandable:** The design should be predictable and clear. Instructions should be simple, and forms should explain errors in plain language.
- **Robust:** Content must work across many devices and assistive technologies, from desktop screen readers to mobile zoom features.

The Web Content Accessibility Guidelines also define three levels of compliance: Level A, Level AA, and Level AAA. Level A covers basic requirements, such as making sure images have text descriptions and that all functions can be used with a keyboard. Level AA, the most common standard, adds stronger requirements such as higher contrast between text and background, captions for video, and consistent navigation across pages. Level AAA is the highest level, adding sign language interpretation for video, more advanced color contrast, and

detailed error messages. While AAA provides the best accessibility, many organizations focus on Level AA as a realistic and balanced target.

Timing is another key factor. In many workflows, accessibility is tested near the end, when the product is nearly finished. Fixing problems at this stage means rewriting code or redesigning interfaces, which is costly and discouraging. Shift-left testing solves this by catching issues early. It has already proven effective in fields like security and performance testing, and this project applies the same principle to accessibility. If a designer chooses accessible colors and labels from the start, developers will not have to redo the work later. Accessibility should be treated as a proactive design value, not a reactive fix.

Figma is an ideal platform for exploring this idea. It is widely used in design teams, supports real-time collaboration, and allows developers to build plugins directly inside the design environment. Some plugins like Stark (for color contrast) and Able (for alternative text) already exist, but they only address one issue at a time. What is missing is a plugin that checks for several common accessibility issues at once and helps designers build inclusive habits early in their process.

2.1 Planned Software Deliverable

The main deliverable of this project is a Figma plugin that runs accessibility checks on design files. It will scan text, shapes, and components using accessibility rules and show results directly on the canvas.

The plugin checks for:

- **Color contrast:** Ensures text and background colors meet WCAG standards.
- **Alternative text:** Flags images and icons without descriptive labels.
- **Focus order:** Confirms that interactive elements follow a logical keyboard navigation order.
- **Target size:** Highlights buttons or touch areas that are too small for users to tap easily.
- **Use of color:** Warns when color alone conveys meaning without text or symbols.

When the plugin runs, it lists all issues and links each warning to the specific element. Designers can click, fix the issue, and re-run the plugin to check if

it's resolved. For instance, if text fails a contrast check, the plugin will suggest accessible color alternatives. If an icon lacks alternative text, it prompts the designer to add a label. This feedback loop makes accessibility both interactive and educational, turning it into part of the creative process instead of an afterthought.

3 Related Work

Most existing accessibility tools focus on code-time testing. Popular examples include axe-core, Lighthouse, and WAVE. These tools scan finished websites or applications and report accessibility issues such as missing ARIA labels, poor heading structures, or broken landmarks. While powerful, these tools are reactive,they catch problems late, when changes are expensive to make.

Design-stage tools exist but are fewer and more limited. Stark checks color contrast, and Able helps add alternative text. These tools are helpful but narrow. Some academic research has tested tools that scan design files for accessibility issues, but adoption is limited. Many designers still assume that accessibility is the responsibility of developers. If tools require leaving Figma or add extra steps, designers are less likely to use them consistently.

Cultural barriers also play a role. Accessibility is often seen as a compliance requirement instead of a core design value. This leads to last-minute fixes instead of early prevention. Research suggests that tools work best when they not only flag problems but also explain why they matter. When designers understand the human impact of accessibility choices, they are more likely to develop long-term habits that lead to inclusive designs.

This project builds on these insights by creating a plugin that integrates multiple checks into one system. It combines automated detection with short educational tips so that designers learn as they work. By meeting designers where they already operate inside Figma, it encourages continuous attention to accessibility. The plugin promotes a mindset that accessibility is everyone's responsibility, not just a specialist's task.

3.1 Future Work

This project is a pilot that lays the foundation for further development. In the future, the plugin can be expanded to include more WCAG 2.2 rules, such as motion sensitivity or orientation constraints. A next step could involve user studies

to see how the tool changes real design workflows, how much time it saves, and whether it reduces accessibility issues before coding begins.

The plugin could also be integrated with code-time tools like axe-core, creating a continuous pipeline from design to development. Another valuable direction is education. The plugin could serve as a teaching aid in classrooms, helping students visualize accessibility issues in their own projects. These extensions would strengthen the plugin's practical and academic value while providing real data on how shift-left accessibility improves inclusion.

4 Conclusion

Accessibility should never be treated as something to fix at the end. When it is ignored early, barriers multiply and exclude real people. This project demonstrates how accessibility can move "left" in the development timeline, placing it at the heart of design rather than at the edge. By using a Figma plugin to identify issues like poor contrast, missing labels, and unclear focus order, designers can build inclusive interfaces from the very beginning.

The contribution of this work is both practical and conceptual. It creates a working tool and shows how shift-left testing can be applied to accessibility. More importantly, it highlights that inclusive design is good design. When accessibility becomes part of the design process, technology becomes more equitable, effective, and human-centered.

This junior project also provides the foundation for a senior independent study. Future work can extend the plugin, conduct experiments with design teams, and collect data on the impact of early accessibility checks. By combining research and practice, the project not only contributes to computer science but also to the broader goal of building digital spaces that work for everyone.

References

1. Abdulaziz Alshayban, Iftekhar Ahmed, and Sam Malek. "Accessibility Issues in Android Applications: State of Affairs, Sentiments, and Ways Forward." In *Proceedings of the 42nd International Conference on Software Engineering (ICSE)*, IEEE/ACM, 2020.

2. Yufei Chen, Han Wu, and Ruijie Wang. "Assessing Accessibility Levels in Mobile Applications from Figma Templates." In *CHI Conference on Human*

Factors in Computing Systems (CHI '24), ACM, 2024.

3. Wentao Duan, Yiheng Zhou, Chen Wu, and Xiang Anthony Cao. "Generating Automatic Feedback on UI Mockups with Large Language Models." In *CHI Conference on Human Factors in Computing Systems (CHI '24)*, ACM, 2024.

4. Wentao Duan, Yiheng Zhou, Chen Wu, and Xiang Anthony Cao. "UICrit: A UI Critic Agent and Critique Dataset." In *ACM Symposium on User Interface Software and Technology (UIST '24)*, ACM, 2024.

5. Rony Ginosar, Hila Kloper, and Amit Zoran. "Parametric Habitat: Virtual Catalog of Design Prototypes." In *Designing Interactive Systems Conference (DIS '18)*, ACM, 2018.

6. Jingyi Huang, Walter S. Lasecki, and Liangjie Feng. "Beyond the Guidelines: Assessing Accessibility in Figma Prototypes." In *CHI Conference on Human Factors in Computing Systems (CHI '24)*, ACM, 2024.

7. Júlia Holanda Muniz, Daniel Mesquita Feijó Rabelo, and Windson Viana. "Assessing Accessibility Levels in Mobile Applications Developed from Figma Templates." In *International Conference on Pervasive Technologies Related to Assistive Environments (PETRA '24)*, ACM, 2024.

8. Wei Shi and Leah Findlater. "It Could Be Better, It Could Be Worse: Understanding Accessibility in UX Practice with Implications for Industry and Education." In *CHI Conference on Human Factors in Computing Systems (CHI '23)*, ACM, 2023.

9. Hanxiao Zhang, Anhong Guo, Xiang Anthony Chen, and Yang Li. "Screen Recognition: Creating Accessibility Metadata for Mobile Applications from Pixels." In *CHI Conference on Human Factors in Computing Systems (CHI)*, ACM, 2021.