Back-end Software Development: A Look into the Importance of Data Verification
(Technical Paper)

How Software is Designed to be Accessible

(STS Paper)

A Thesis Prospectus Submitted to the

Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements of the Degree

Bachelor of Science, School of Engineering

Jonathan Hail

Spring, 2023

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Signature _____ Date _____

Jonathan Hail

Approved _____ Date _____

Joshua Earle, Department of Engineering and Society

Approved _____ Date _____

Rosanne Vrugtman, Department of Computer Science

Approved _____ Date _____

Briana Morrison, Department of Computer Science

Approved _____ Date _____

Kathryn A. Neeley, Associate Professor of STS, Department of Engineering and Society

Introduction

My technical project follows the work I did at my summer internship as a Walmart Software Engineer. During my internship, I developed a back-end service to ensure Walmart financial services associates have the proper training for the service that they are providing to the customer. My STS project focuses on how websites can be designed to remain accessible to potential users. These two are connected because this research will allow me to properly research how software could be more accessible, so I can design better software at my jobs in the future and not exclude a certain population.

This topic is important to ensure that new technologies can be utilized by the widest audience as possible, as excluding people from technologies that can empower them is not ideal. If more developers are aware about differing disabilities that prevent a certain population from using their software, they could add more design features that allow the wider audience to user their software. This would be mutually beneficial for both developers and the general public as the developers would increase the audience of their software and it would allow the public to have equal opportunity to put said software to use.

While these features can help bridge the gap between the general populace and users who rely on such features, these features can benefit all users. Features that help make software more accessible can also be of use for convenience, and generally improve software quality. These features can be categorized into several areas, and can benefit people in different ways. Some current websites lack important features, and it is important to recognize their shortcomings and see how their structure can be improved upon.

Technical Project

Walmart was sued for having insufficient prevention for fraud in their financial services department. This was in part due the claim that associates were not properly trained for the financial service they were providing. To address this, I created a back-end service to determine whether an associate is trained for the transaction type they are performing. I implemented this remedy in Java Spring Boot as a RESTful API. When the API was finished, it returned a list of the required courses an associate has not taken for each action. In the future, this service could be expanded to departments of Walmart other than Financial Services—the Pharmacy department, for example.

STS Project

My research question is: How can websites be designed such that the software remains accessible to all potential users? This can be found through research in how to make websites accessible and where accessibility is currently lacking in modern websites. Accessible design choices help people with disabilities such as colorblindness, blindness, dexterity disabilities, hearing, or cognitive disabilities.

Accessibility in software is important because issues excluding populations should be resolved, as they can have certain political consequences, some directly and some indirectly. The direct consequences of this would be the immediate exclusion of the potential users from using the software. This is not ideal as it not only excludes potential users for the developer, but also is preventing certain people from using it. The indirect consequence of this is whatever the software could be enabling users to do, such that potential users that cannot access the software are also excluded from the end use. The relevant social groups are people who need accessibility features (such as colorblindness).

My timeline consists of reviewing literature to find how websites are currently inaccessible to find the features that could be improved upon. From there, I will research how design can be shifted to include an accessibility design perspective and be changed to be more accessible for existing software or ensured to be accessible as being developed. I will also find examples of websites that embody newer accessibility features and ones that are currently lacking in them. During this, I will report my findings in my STS paper. The STS research method I will use is literature review to research shortcomings of accessibility in websites and how to improve them.

Key Texts

Experiencing some sort of disability that could be aided by software accessibility in the United States is quite prevalent. These types of disabilities can be classified into the following: vision, dexterity, hearing, and cognitive (Grieves, 2009). Each of these categories can be mitigated by implementing features in the software design process. Possible solutions for these categories in general application are as follows. For vision disabilities, an option to increase font size or style, and screen readers can help increase ease of access (Grieves, 2009). For dexterity, features like on screen keyboards or other methods of input can decrease strain of the precise mechanical input of a keyboard (Grieves, 2009). For hearing disabilities, options for volume control, text captioning, or sign language let users who have trouble understanding audio in software to understand the language properly (Grieves, 2009). Finally, features that assist with cognitive disabilities are a simplified user interface, intelligent suggestions for user input, reminders for user action, and reading/learning aids (Grieves, 2009). These aforementioned features not only help with those who need them to use the software equally, but it also helps

other users generally using the application. For example, many people enjoy using closed captioning in software such as Netflix to assist in clearly understanding everything said.

With the increasing number of people with disabilities and the number of older citizens increasing, having websites that are accessible is more important than ever (Adam & Kreps, 2006). Adam and Kreps (2006) describe that according to a survey in the UK, less than 20 percent of websites meet even the most basic accessibility standards. There is currently legislation in website accessibility, it is relatively new and hasn't made a huge impact in ensuring websites remain accessible (Adam & Kreps, 2006).

Currently, the visually disabled use technology to read the content of a webpage to them (Adam & Kreps, 2006). However, with the introduction of images, the website can become more inaccessible especially if a description of the image is not provided in the HTML (hypertext markup language) in which the website is designed in (Adam & Kreps, 2006). Other common design issues also manifest themselves in tables, where it can be confusing how the data is organized if read out row by row (Adam & Kreps, 2006).

However, this inaccessibility from older design can be changed. According to Harold (2008), one of the first steps is replacing images with text. Even though images can have descriptions making them more accessible, replacing them with text and CSS (Cascading Style Sheets) styling drastically improve the accessibility (Harold, 2008). Another method is adding labels to form input, which not only increases usability by the visually impaired, but also can improve browser functionality (Harold, 2008). To improve accessibility to tables, which as aforementioned can be difficult to understand when listed out by a screen reader, captions and summary elements can be included (Harold, 2008). This way, they can understand the meaning behind the table without having to listen to the whole table or having to memorize the data

(Harold, 2008). These are some of the ways existing websites can be reformatted, and as seen, some of these modifications can be minor, but can provide a major help to those who need the features. In addition, these features not only help the blind: “The changes you make to improve accessibility don’t just help people who meet the legal definition of disabled. They help everyone whose seeing, hearing, or motor skills are less than perfect.” (Harold, 2008, ch. 6).

To ensure these features make their way into modern websites, it is required to test for them. This can either be manual or automated. There are some issues with relying solely on automated testing: “the problems of inaccessibility are further compounded by a reliance on automatic checkers, which cannot possibly verify the accessibility of a web site without a human check” (Adam & Kreps, 2006, How is the web accessible or inaccessible?). As seen, manual testing is still important, however, not all features must be checked manually, and manual testers can work in more niche areas (Palani, 2019). Automated testing can take place using tools such as Selenium, which can test if the website conforms to accessibility principles as it’s being developed, which has been best practice since 2017 (Palani, 2019). In conjunction with tools such as JAWS and NVDA (screen reading programs), automated testing can be applied with good coverage (Palani, 2019).

Bibliography

Adam, A., & Kreps, D. (2006). Enabling or disabling technologies? A critical approach to web accessibility. *Information Technology & People*, *19*(3), 203–218. https://doi.org/10.1108/09593840610689822

Grieves, J. (2009). *Engineering Software for Accessibility*. Microsoft Press. https://learning.oreilly.com/library/view/engineering-software-for/9780735642102/

Harold, E. (2008). *Refactoring HTML: Improving the Design of Existing Web Applications*. Retrieved December 8, 2022, from https://learning.oreilly.com/library/view/refactoring-html-improving/9780321552044/

Palani, N. (2019). *Advanced Selenium Web Accessibility Testing: Software Automation Testing Secrets Revealed*. Momentum Press. http://ebookcentral.proquest.com/lib/uva/detail.action?docID=5742318