

Final Report

Automating Accessibility

AUTHOR: David Thomsen
EMAIL: dthom.cap24@gmail.com

Table of Contents

Table of Contents.....	1
Introduction.....	2
Abstract.....	2
Problem Statement.....	2
Objectives.....	2
Significance.....	3
Necessary Vocabulary.....	4
Virtual Machine.....	4
Host Machine.....	4
Scripts.....	4
Registry Keys.....	4
Ansible.....	4
Vsphere.....	5
Proof of Concept.....	6
What does this mean?.....	6
What would the full build include?.....	6
What would a full build in a working environment look like?.....	7
Why build a Proof of Concept?.....	7
Project Breakdown.....	7
Sprints and Deliverables.....	7
How it Works.....	11
Trial and Error.....	15
Script Creation / Invocation.....	15
Powershell for Registry Edits.....	15
Applications and Settings.....	15
Conclusion.....	16
Bibliography.....	17



Introduction

Digital accessibility has become extremely important in today's interconnected world. Individuals from all walks of life use digital platforms for essential services, communication, information, . The need for inclusive technology is only going to continue to rise, and I believe that building it from the ground up is the best option. I also have a personal connection to the topic of accessibility as a whole, and would like to work to create a better, more accessible digital landscape.

Abstract

In today's digitally connected world, ensuring accessibility is crucial, especially in workplace environments. This project aims to tackle this issue by developing an application that customizes desktop environments according to individual needs. By simplifying the onboarding process, this tool integrates accessibility seamlessly into workplace practices. Personalized configurations empower employees with tailored environments and promotes inclusivity by removing the stigma associated with requesting accommodations. Despite encountering challenges such as failed implementations and scope limitations, this proof of concept demonstrates technical feasibility and lays the groundwork for future development. In the end, the goal is to prioritize inclusivity in technology and contribute to a more accessible digital society.


Problem Statement

Within modern workspaces, there are two main issues that this project aims to address.

- Most Work environments are created based around one standard desktop environment, and the responsibility to adjust the computer to the user's accessibility needs falls on them.
- This also creates an environment where individuals may feel singled out for asking for additional help or accessibility requirements.

Objectives

This project aims to solve the aforementioned problems by integrating desktop accessibility into the onboarding and hiring process. This ensures built-in accessibility for all



employees, thereby enabling personalized desktop configurations tailored to individual needs. Additionally, this fosters an environment where employees with desktop accessibility requirements feel integrated rather than singled out, fostering a culture where every voice contributes to a more accommodating workspace.

Significance

The inspiration for this project stems from a personal commitment to inclusivity and equal access to technology, influenced by familial experiences. By addressing the issue of digital accessibility, this project prioritizes accessibility and equity for all individuals, regardless of their background or abilities, promoting a more inclusive digital society.

Necessary Vocabulary

Before getting into the main core of the project and how it works, there are a few terms that will be used commonly that have been defined here.

Virtual Machine

“A Virtual Machine (VM) is a compute resource that uses software instead of a physical computer to run programs and deploy apps. One or more virtual “guest” machines run on a physical “host” machine. ” (VMWare)


Essentially, this is just a way to emulate desktop environments for multiple systems while using only one main host machine.

Host Machine

A host machine is the physical computer hardware that is running or managing the virtual machines. One host machine can hold many virtual environments.

Scripts

“In computer programming, a script is a program or sequence of instructions that is interpreted or carried out by another program rather than by the computer processor (as a compiled program is)” (TechTarget).



Within this project, the main scripting languages that are being used are Powershell and Python scripts.

Registry Keys

“The registry is a hierarchical database that contains data that is critical for the operation of Windows and the applications and services that run on Windows.” (Microsoft)

The windows registry is where all the values of the Windows settings are stored and this is what is being edited in the application to force setting changes.

Vsphere

Vsphere is the application that is being utilized in order to create and manage the virtual machines. It is a powerful tool that can be manipulated with Powershell scripts in order to interface with the virtual machines.



Proof of Concept

What does this mean?

This Proof Of Concept aims to demonstrate the feasibility of automating accessible desktop environments within a real-world workplace setting. It showcases the foundational principles and mechanics that enable this level of integration for accessibility. Essentially, this showcases that this is possible and can continue to be developed into a full build within a workspace.

What would the full build include?

There are many things that I had wanted to include that simply were not able to be done due to time constraints, hardware constraints or both.

I would have liked to include support for other operating systems, however, this would have to be an entirely different process, as this version was built around Windows. I believe that recreating this project in a Linux/Unix Environment would be much easier than Windows. With this said, I believe recreating this application for MacOS would be even more difficult, due to availability/budget for this project as MacOS is extremely hard (and questionably legal) to emulate and clone using virtual machines.

I would have also liked to dive into additional applications, however this also starts a rabbit hole of all the possible applications and full functionality within this application. Again, as much as I would like to explore this, this was out of scope for the project.

So what WAS in scope that simply did not get added?

1. Multiple Windows Ease Of Access features simply did not want to work properly when I was working with registry keys. This covered multiple settings that were in the original plans, but had not made it into the final application.

- Captions: This one was the most frustrating setting that I was unable to solve. In many of the sources I had found, they had all pointed to a specific registry key that dealt with the captions and the caption settings. When I went to change these settings in my own VM, the setting was completely absent. I was unable to find a resolution for this and I believe it was due to the Windows box being virtualized.



- Cortana: Unfortunately HAS to be turned on manually.

- Sticky Keys & High Contrast: Both of these fall into the same group. These are both *technically* possible to implement, but with much trial and error within the registry, I was unable to get these to consistently work.

2. A GUI was meant to be implemented, but due to timing, this was not possible and a TUI was decided on. This would have been possible to implement had there not been so many hurdles with the Hypervisor.

3. Active Directory support was planned originally, however was not implemented due to time requirements. I had planned to include User Creation, Domain Joining, DNS record Addition, GPO management, and a few other additions, but had not been able to implement this fully. I believe that this one also could have been completed had I not had so many issues in the beginning.

What would a full build in a working environment look like?

In a fully implemented working environment, the automated accessibility solution would be seamlessly integrated into the onboarding process for new employees. Upon candidate selection, individuals would be prompted to complete a form detailing their specific accessibility requirements. This form has a large range of settings that cover many different areas of accessibility, none of which is disclosed to anyone except the employee filling out the form, and the administrator enabling the settings. Subsequently, the desktop environment would be configured automatically based on these preferences, providing each employee with a personalized workspace tailored to their needs. This comprehensive approach ensures that accessibility is ingrained into the workplace culture, fostering inclusivity and productivity for all employees.

Why build a Proof of Concept?

Building a proof of concept serves several crucial purposes:

- ❖ **Validation of Concept:** It validates the technical feasibility and viability of automating accessibility in workplace desktop environments.
- ❖ **Risk Mitigation:** It allows for early identification and mitigation of potential challenges and obstacles before full-scale implementation.

- ❖ Stakeholder Engagement: It provides a tangible demonstration of the benefits and value proposition of the proposed solution, garnering support from stakeholders and decision-makers.
- ❖ Iterative Development: It serves as a foundation for iterative development and refinement, enabling continuous improvement based on feedback and insights gathered during the POC phase.


Project Breakdown

Sprints and Deliverables

Over the course of the last semester, the capstone project was broken into four large sections called sprints. Each one of these sprints had a different goal and changed multiple times throughout the course of the project.

Sprint One

Sprint One focused on getting the hypervisor selected and running in a way that allowed me to begin building a working environment for testing and managing. This sprint ended up being a bit all over the place for multiple reasons. I had selected a Hypervisor called Proxmox, which is an open source and free-to-use tool. There were a few issues with Proxmox that had caused me to migrate over to another hypervisor. The issue with Proxmox was a lack of proper documentation and resources that could be utilized in order to navigate the CLI (Command Line Interface). With that, I had then used the opportunity to migrate the server to a HyperV environment instead. This is a Windows specific hypervisor that allows the creation and management of virtual machines. The reason this had not been the hypervisor chosen was because of an error that I made in the setup of this environment. Rather than installing and managing HyperV on the server's physical hardware, I had nested it inside of Proxmox to avoid deleting my Proxmox environment. This was a mistake as the speeds of these machines were unbearable and no work could be done in an efficient manner. There was also an additional issue with this locally-hosted environment; the server was running on hardware that was being managed within my apartment, causing the electric bill to spike, ruling this option as too expensive.



Sprint One and a Half

This sprint was created out of necessity as I was about to move on to sprint two, and I ended the first sprint back at square one. Luckily, I was able to migrate my project over to a local/cloud server that was being hosted by the school. This was a HUGE benefit for multiple reasons. First of all, I was able to keep the server up and running constantly allowing for unlimited access, as well as not having to deal with the electricity bills of keeping a server running 24/7. Additionally, I was able to use the VSphere licenses that were available from a class I am currently enrolled in, in order to rebuild the environment within Vsphere. This sprint was done just under a week or so and was focused on just building a testing environment that would be used later in the project.

Sprint Two

The second sprint is where the coding and creation of the main parts of the project are created. This is the beginning of the two main files of this project.

First, a google form was created that is used to gather accessibility information from the end-user. This form was then parsed using a Python script in order to collect and write this data to a CSV (Comma Separated Values) File. The way this script works is by gathering the administrator's google credentials (specified within a credentials file), then grabbing the response form and writing all of the information from it into separate, named files, as seen below.

Capstone Form

Your response has been recorded.

[Submit another response](#)

```
Please submit the form at the following link: https://forms.gle/GXCtRy1oD9VZxsdm8
```

```
Press Enter to continue:
```

```
Grabbing CSV from Google Sheets...
```

```
File 'AccessibilityAutomation/Capstone-Utills/CSVs/david-thomsen.csv' already exists.
```

```
File 'AccessibilityAutomation/Capstone-Utills/CSVs/abijah-buttendorf.csv' already exists.
```

```
File 'AccessibilityAutomation/Capstone-Utills/CSVs/report-test.csv' created successfully.
```

```
Selecting CSV
```

```
AccessibilityAutomation > Capstone-Utills > CSVs > 📄 david-thomsen.csv
```

```
1 david,thomsen,No,No,No,No,Yes,No,English,No thank you!
```

Additionally, the Powershell modules that are needed in order to create basic functions were created within this sprint. This allows the functionality for easy module creation and function creation for later use in the runner file and driver file. Below is a screenshot of the simple banner functionality as an example.

```
# Capstone Banner
function CapBanner(){
    $banner = @"
    David Thomson
    Automation
    Accessibility
"@
    Write-Host -ForegroundColor Cyan $banner
    Write-Host -ForegroundColor Green "https://github.com/dthomsen116/AccessibilityAutomation"
}
```

Sprint Three

This is where most of the functionality from the modules file is created. This sprint is where a large portion of the modules were built out, as well as the creation and testing of virtual environments. All of this functionality is integrated into the later scripts that are used once it is all working.

Modules created for use include:

Connect/Disconnect-Cap: Used to establish or erase a connection to the VSphere server

Select-VM: Used for internal VM selection

CreateClone: Used to create the base image of a Windows virtual machine

TurnOnNewClone: Powers on the newest clone

Create-Script: This was the beginning of the function that would actually make all of the accessibility adjustments that were necessary.

This sprint also included much testing for editing the registry keys and enabling other settings. The final choices for the settings that can be changed or enabled was decided and includes Narrator, Magnifier, Scaled Display, On-Screen Keyboard, Visual Alerts, Dark Mode, and a large selection of languages. This went through multiple iterations, but that is explained further in the trial/error section.

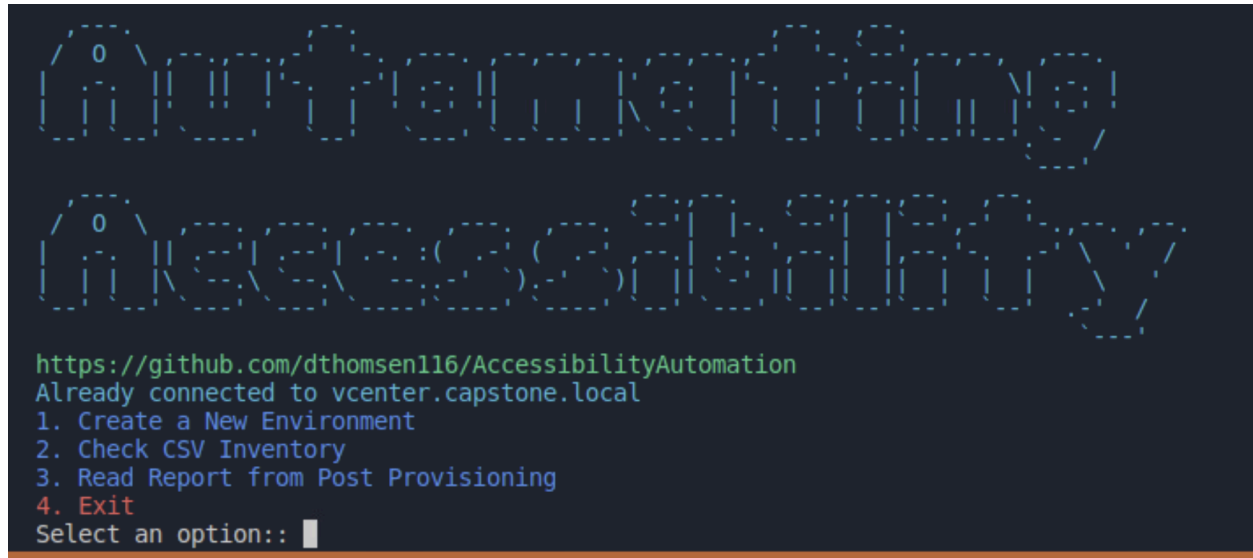
Sprint Four

The final sprint includes much of the wrap-up work that was necessary. The runner file was cleaned up in order to go through the creation process efficiently and without issues. The driver file was created to act as a TUI (Terminal User Interface) for easier operation of the application. Demo Videos were filmed throughout the process in order to ensure that the project was working properly. Documentation was finalized, reports were written, and presentations were made and given. All of this clean up and final testing was done in the hopes to clean up the final product and have a more fluid presentation piece.

How it Works

I have filmed a demonstration video that can be found at the following link: [HERE](#). The use of this application is straightforward and easy, as I wanted all parts of this tool to be easily accessible. There is one file called the driver file that houses all of the functions and scripts that are going to be utilized throughout the process.

First, the user is brought to the main menu of the application.

A screenshot of a terminal window displaying the main menu of the 'AccessibilityAutomation' application. The title 'AccessibilityAutomation' is shown in a large, stylized, dashed font at the top. Below it, the GitHub repository URL 'https://github.com/dthomsen116/AccessibilityAutomation' is displayed in green. The terminal then shows the message 'Already connected to vcenter.capstone.local'. A list of four options is presented: '1. Create a New Environment', '2. Check CSV Inventory', '3. Read Report from Post Provisioning', and '4. Exit'. The 'Exit' option is highlighted in red. At the bottom, the prompt 'Select an option::' is followed by a cursor.

After “Create a New Environment” is selected, the user is prompted to fill out a Google form that is linked:

```
Please submit the form at the following link: https://forms.gle/GXCtRy1oD9VZxsdm8
Press Enter to continue:

Grabbing CSV from Google Sheets...
File 'AccessibilityAutomation/Capstone-Utils/CSVs/david-thomsen.csv' already exists.
File 'AccessibilityAutomation/Capstone-Utils/CSVs/abijah-buttendorf.csv' already exists.
File 'AccessibilityAutomation/Capstone-Utils/CSVs/report-test.csv' created successfully.
Selecting CSV...
```

This then creates the user’s CSV file which holds all of the selections as well as comments the user may have included at the end of the form.

```
Press Enter to continue:

Grabbing CSV from Google Sheets...
File 'AccessibilityAutomation/Capstone-Utils/CSVs/david-thomsen.csv' already exists.
File 'AccessibilityAutomation/Capstone-Utils/CSVs/abijah-buttendorf.csv' already exists.
File 'AccessibilityAutomation/Capstone-Utils/CSVs/report-test.csv' already exists.
Selecting CSV...
1. abijah-buttendorf.csv
2. david-thomsen.csv
3. report-test.csv
Enter the index of the file you want to select: █
```

The user is selected, then the runner file begins to run. A new clone for the user is created, then a login prompt is provided.

```

Selecting CSV...
1. abijah-buttendorf.csv
2. david-thomsen.csv
3. report-test.csv
Enter the index of the file you want to select: 3
You have selected: /home/david/Documents/AccessibilityAutomation/Capstone-Utills/CSVs/report-test.csv
Creating Clone...
CSV file loaded from /home/david/Documents/AccessibilityAutomation/Capstone-Utills/CSVs/report-test.csv

Name                PowerState Num CPUs MemoryGB
-----
report-test         PoweredOff 2          4.000
Full Clone created: report-test
report-test         PoweredOn  2          4.000
VM report-test powered on
Waiting for the clone to be ready...
...
...
still waiting...
Almost there...
Enter the username for the VM: █

```

After logging in, the specific user modifications are beginning to change.

```

VM                : report-test
ExitCode          : 0
ScriptOutput      :
Uid               : /VIServer=capstone.local\david-adm@vcenter.capstone.local:443/VirtualMachine=VirtualMachine-vm-3055/VMScriptResult=-1145432912_0/
Length            : 0

Narrator enabled

VM                : report-test
ExitCode          : 0
ScriptOutput      :
Uid               : /VIServer=capstone.local\david-adm@vcenter.capstone.local:443/VirtualMachine=VirtualMachine-vm-3055/VMScriptResult=-1145432912_0/
Length            : 0

Magnifier enabled
█

```

After all of these changes go through the system, the virtual machine is restarted and a report is generated detailing what was changed for the user, as well as mentioning the comment from the user for additional requirements.

```
Language settings applied. Restarting VM
Report saved to AccessibilityAutomation/Capstone-Utills/Reports/report-
test.txt
```

This report can be read from the TUI by pressing '3' and navigating to the report that is needed.

```
https://github.com/dthomsen116/AccessibilityAutomation
Already connected to vcenter.capstone.local
1. Create a New Environment
2. Check CSV Inventory
3. Read Report from Post Provisioning
4. Exit
Select an option:: 3
[0] abijah-buttendorf.txt
[1] david-thomsen.txt
[2] report-test.txt
Select a file by number: 2
Content of report-test.txt:
Accessibility Report for report test
-----
Requested Settings:
-----
```

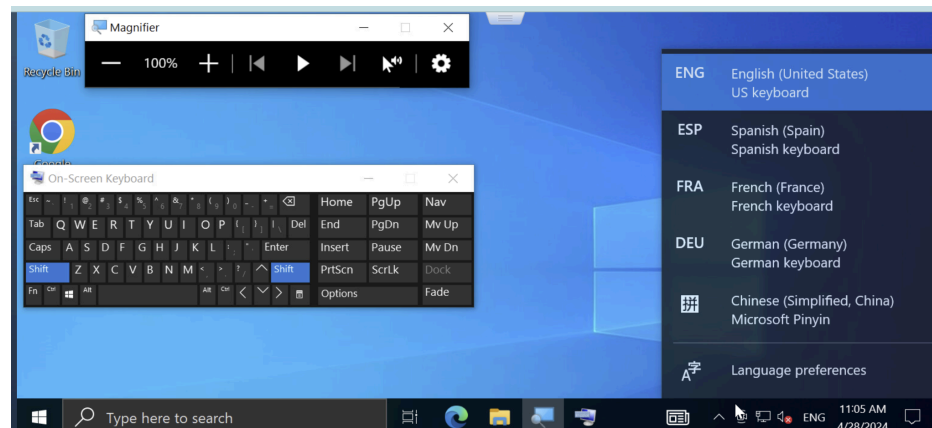
```
Content of report-test.txt:
Accessibility Report for report test
-----
Requested Settings:
-----
- Narrator
- Magnifier
- Scaled Display
- On-Screen Keyboard
- Dark Mode
- Visual Audio Alerts

Applied Settings:
-----
- Narrator : Yes
- Magnifier : Yes
- Scaled Display : Yes
- On-Screen Keyboard : Yes
- Dark Mode : Yes
- Visual Audio Alerts : Yes

Languages Enabled:
-----
English
Spanish
French
German
Chinese (Simplified)

Additional Comments/Needs:
-----
No thank you!
-----
EOF
```

Now, when the desktop is booted and logged in, the user's settings will be enabled on startup.





Trial and Error

Throughout this project, there was much trial and error as well as additional features that did not quite make the final project, due to physical limitations as well as improper time management in the beginning of the project. During this project, many mistakes were made in order to reach the end product.

Script Creation / Invocation

The number one item within this project that was changed around was how the settings were actually going to be changed. This has taken many forms and I believe that with further fine-tuning, I would be able to make this a much more efficient program. My first idea for creating a script was to edit the registry keys directly. While this was able to change a few minor settings, overall this had not worked for a few reasons. First, when invoking a script remotely, there is no field for user input, so if the virtual machine is asking you to verify something, this can cause issues when trying to force a change. When creating and running a .reg file (used to make changes to the registry directly), there is a warning prompt for the end user stating that, “Adding information can unintentionally change or delete values and cause components to stop working correctly...”. This is in place for the inexperienced users who may or may not know what the registry keys are capable of. Luckily, there was a workaround for this issue which leads us into error two.

Powershell for Registry Edits

After running into issues with .reg files, I learned that there was a way to edit registry keys using powershell, and on top of that, there is a “-Force” flag which removes any warnings and just forces the change to be made regardless of the system’s safety. This had worked in order to change the registry keys, however there were a few applications that were not able to be changed using registry keys and needed a work around. When combining the powershell and registry keys functionality, I had made errors within the creation code so it would never quite work 100% of the time. This led me to the current/final version where each setting has its own customized Powershell script that will execute for each setting without issues.

Applications and Settings

After figuring out how to properly invoke these settings, I needed to figure out which ones I was going to implement into this project. Windows has a multitude of Ease of Access applications, however between Powershell and Registry keys, some of them could only be invoked manually. There are multiple settings that I had tried to enable with varying levels of success for each.

Closed Captioning - As mentioned above (Proof Of Concept), closed captioning was a setting I was unable to get working on my machine. After doing research into how to enable it, many sources said to look at a specific registry key that my virtual machine just did not have. I am unsure why this was but if this were not a virtual environment, I believe I could have successfully enabled this setting as an option as well

Cortana - after looking into multiple sources, while installing Cortana remotely is possible, there is no way that I found in order to enable or configure Cortana from the command line.

High Contrast / Color Themes / Sticky Keys - All of these fall into a very similar category, all of these have registry keys and settings that can be edited remotely, however I was unable to find enough resources in order to get them fully working. For example, with high contrast, I was able to find the key values, but was unable to figure out how to adjust it to do what I was aiming for. This was the common theme for all three of these settings.

Conclusion

In conclusion, my project "Automating Accessibility" has made significant strides towards addressing the pressing need for inclusive technology in today's digital landscape. Through the development of a tool that customizes desktop environments based on individual accessibility requirements, the project aimed to seamlessly integrate accessibility into workplace practices. While the journey encountered challenges and limitations, the proof of concept demonstrates technical feasibility and lays a strong foundation for future development.

Bibliography

Brink, S. (2021). Ten Forums - Windows 10 Help and Support Forum. www.tenforums.com.

<https://www.tenforums.com/>

CAST. (2018). Universal Design for Learning Guidelines Version 2.2. UDL Guidelines; CAST.

<https://udlguidelines.cast.org/>

Chambers, J. A. (2020, December 10). Modify Google Sheets (API) Using PowerShell /
Uploading CSV Files. James A. Chambers -- Legendary Tech Blog.

<https://jamesachambers.com/modify-google-sheets-using-powershell/>

Microsoft. (2023, April 10). Launch the Windows Settings app - UWP applications.

[Learn.microsoft.com](https://learn.microsoft.com).

<https://learn.microsoft.com/en-us/windows/uwp/launch-resume/launch-settings-app>Microsoft. (2024a).

Microsoft Support. [Support.microsoft.com](https://support.microsoft.com). <https://support.microsoft.com/>

Microsoft. (2024b, April 12). Microsoft PowerToys. [Learn.microsoft.com](https://learn.microsoft.com).

<https://learn.microsoft.com/en-us/windows/powertoys/>

Mozilla. (2024, March 15). Sending form data. MDN Web Docs.

[https://developer.mozilla.org/en-US/docs/Learn/Forms/Sending and retrieving form data](https://developer.mozilla.org/en-US/docs/Learn/Forms/Sending_and_retrieving_form_data)

OpenAI. (2024). ChatGPT. [Chat.openai.com](https://chat.openai.com); OpenAI. <https://chat.openai.com/>

proxmoxer. (2023, November 27). Proxmoxer: A Python wrapper for Proxmox REST API.

GitHub. <https://github.com/proxmoxer/proxmoxer>

Renaud, K., & Coles-Kemp, L. (2022). Accessible and Inclusive Cyber Security: A Nuanced and Complex Challenge. SN Computer Science, 3(5).

<https://doi.org/10.1007/s42979-022-01239-1>

Ro, J. (2016, February 10). Windows 10: How to Enable Dark mode in PowerShell. Joon's Blog.

<https://joonro.github.io/blog/posts/windows-10-enable-dark-mode-posh/>

Snow, J. (2019, January 30). How People with Disabilities Are Using AI to Improve Their Lives. PBS.

<https://www.pbs.org/wgbh/nova/article/people-with-disabilities-use-ai-to-improve-their-lives/>

Stone, M. (2023, April 19). The Importance of Accessible and Inclusive Cybersecurity. Security Intelligence.

<https://securityintelligence.com/articles/importance-of-accessible-inclusive-cybersecurity/>

TechTarget. (2021, December). What is script? - Definition from WhatIs.com. WhatIs.com.

<https://www.techtarget.com/whatis/definition/script>

VMware. (2022, January 27). What is a Virtual Machine? | VMware Glossary. VMware.

<https://www.vmware.com/topics/glossary/content/virtual-machine.html>

W3C. (2018). Essential Components of Web Accessibility. Web Accessibility Initiative (WAI).

<https://www.w3.org/WAI/fundamentals/components/>

W3C. (2019). Home. Web Accessibility Initiative (WAI). <https://www.w3.org/WAI/>

W3C. (2023, September 21). Web Content Accessibility Guidelines (WCAG) 2.1. W3.org.

<https://www.w3.org/TR/WCAG21/>



W3Schools. (2019). HTML Forms. W3schools.com.

https://www.w3schools.com/html/html_forms.asp