SOHOIT

by

Travis Banks, Jacob Cudnik, Zach Murphy, Zack Vanderpool

Submitted to

the Faculty of the School of Information Technology

in Partial Fulfillment of the Requirements for

the Degree of Bachelor of Science

in Information Technology

Travis Banks 04/11/2021

Date

Jacob Cudnik 04/11/2021

Date

Zachary Murphy 04/11/2021

Date

Zachary Vanderpool 04/11/2021

Date

Ryan Moore 04/11/2021

Ryan Moore, Faculty Advisor Date

University of Cincinnati

College of

Education, Criminal Justice, and Human Services

April 2021

Table of Contents

List of Illustrations

TABLES

FIGURES

ABSTRACT

SoHoIT is a service dedicated to bringing Information Technology (IT) tools to small businesses who cannot find room in their budget to hire IT personnel. Small businesses can benefit from SoHoIT's services and reduce the cost of their initial IT expenses by using pre-selected software. The software offered can be configured to fit the user's current environment, and consists of services such as webhosting, VPN, and anti-virus. Custom configuration is dictated by our questionnaire that is filled out by a small business or individual looking to download software offered through our platform. SoHoIT is designed to make IT accessible and valuable to small businesses.

INTRODUCTION

Problem Statement

According to a US government study, there were over 5.2 million small businesses with less than 20 employees in 2018 (United States Small Business Administration Office of Advocacy, 2018.) When you run a business with so little manpower, that does not leave much room for an IT department. Without a dedicated IT staff, small businesses are unequipped to handle the ever-growing reliance on technology. A Symantec Internet Security Threat Report found that 55% of small businesses had suffered a cyberattack in 2016. (DiGiacomo, 2017) In this modern age, it is said that different "information technologies have become necessities rather than luxuries" (Lee et al., 2001). That is where our software packaging service, SoHoIT, comes in.

Solution

The solution is SoHoIT, a collection of open-source software that covers the bases of security, web hosting, and networking for small businesses who may not have the time, money, or expertise to set up their own IT. SoHoIT is a website with different open-source software and configuration scripts for each program that will install and configure the program for users. Once run, the user can select which pieces they want to download and install based on their specific needs. From there, the program will install and run an automated setup for each software installed.

When compared to current solutions such as Ninite, this solution will install and configure the software without needing the user to configure settings within installation wizards. This will make software downloaded from SoHoIT more user friendly than the same software downloaded from Ninite or other sites on the web. Since we are using scripts instead of a full installer, downloads from our site will also be lighter in terms of the user's storage.

Project Goals

With SoHoIT, the main goal is to provide small businesses with accessible software to answer IT demands of small businesses that do not have the funds to hire IT associates. We plan to achieve that goal by gathering the software into one place, using configuration scripts to make the installation process as simple as possible, and providing tutorials on how to use each software.

Overview

The following sections of this paper outline key components that led to the completion of SoHoIT. This includes the following sections: project concept, design objectives, methodology, user profile, use case diagram, testing, budget, project timeline, and the problems encountered.

DISCUSSION

Project Concept and Solution

We formed the group in the summer of 2019 when we met as interns at the Cincinnati Insurance Companies. After spending a few months bouncing ideas off each other, we decided on creating a small business security package. When we brought the idea to Professor Ryan Moore, he suggested narrowing down the focus for small businesses and home offices and to provide not only security related software, but also common business, networking, and file storage software.

Design Objectives

Our design objectives are split into two groups, with achieved goals and abandoned goals.

Achieved Goals

- Downloads Page to download the software packages available.
- Configured installs for Apache Web Server, KeePass, OpenVPN, and ClamAV
- GUI's for OpenVPN and ClamAV to make it easier for inexperienced users to use.
- Tutorials Page with videos for each software offered to assist inexperienced users.

Abandoned Goals

Our initial goals were to feature a questionnaire for each software on the downloads page that would offer different configurations to the users. We decided to abandon this goal as we found that the questionnaire became redundant with the base configuration scripts.

We also planned to offer a fifth software, Zabbix, which is a network monitoring software made to keep an eye on both the security and the performance (high CPU usage, pending updates, etc.) for systems on the network. We experienced multiple roadblocks when it came to this software and ended up having to abandon it for lack of time.

Methodology

Design Requirements

The requirements that we gave ourselves at SoHoIT was to give a simple experience to the user where all they must do is select what software they want and give us some basic information about their business and then it is automatically installed.

We wanted the users to experience simplicity when using our web application. To achieve this practice of simplicity we kept our design extremely simple. Our project mostly focuses on software configuration and network configuration for integrating software into a user's business. To break down our application we can look at the front end of the website, the backend, and the client side when the user is interacting with the software.

Website

Most of the project is written and compiled in Python, which also includes the framework the website is hosted on. The framework used is Django for Python which handles the website traffic and hosting. The reason we chose Django started when we figured out how to compile multiple executables into a single file using python. To keep things simple, we evaluated our options within Python to keep things unified and Django was our choice. Our plan for Django is to run configuration scripts on the back end, which will be discussed further in the backend section of this report.

Backend

As mentioned in the previous section, the backend uses an AWS web server with Django web framework on it. The software used to encode all the files into an executable is a python module called Pyinstaller. Pyinstaller specifically has an argument that can put everything into a single file and then it can be executed without the user ever needing to have python on their system. The executables created by Pyinstaller are stored on the web server and are given to the user as a download when they click on the download buttons on the site.

Client

Most of the work on the client end is handled by the configuration scripts that configure the installation when installed. All that the user must do is run the install executable as admin, and the software will be installed according to the scripts.

Procedures

With the focus of the project being custom installs, users will go to the downloads page where they can select the software they would like to download. Once the user downloads the executable, they will need to run the file as an administrator. Once the executable has run, it will install the application according to the instructions provided by our scripts.

User Profile

Tables 1, 2, and 3 below describe the three main types of users that will most frequently use SoHoIT. Table 1 shows a business-oriented user who wants to set up different services but may not have the knowledge to do so. Table 2 shows a user who has more computer experience and would use our site to experiment with new software that they may not be familiar with. Table 3 shows a user who would be maintaining and updating the site from the backend.

Table 1: Small Business CEO User Profile
Application: SoHoIT Web Interface (Internet Browser such as Chrome and Firefox)
Potential Users: CEO/Leader of a small business who is business focused.
Software, Interface, and Related Experience: For this user profile it focuses on a CEO who is not technically experienced but may know how to use a computer to do basic things such as web browsing and emailing.
Experience with Similar Applications: Experience with surfing the web on desktop browsers and on mobile devices are applicable. The user will also require the ability to understand how to download a file from a web site and run it with admin credentials.

Task Experience: Users who may have used the internet to browse social media or to find out information.
Frequency of Use: The user will only need to interact with our web interface whenever they would like to look for new software to incorporate into their business. Therefore, in most cases, it would not be very often.
Key Interface Design Requirements that the Profile Suggests: The user would be experienced when it comes to using websites and webpages, but the SoHoIT site should still be as simple to use as possible in order to avoid confusion.

Table 2: Advanced Computer User Profile
Application: SoHoIT Web Interface (Internet Browser such as Chrome and Firefox))
Potential Users: Advanced Computer User - Software developer, database analyst, graphics designer, video editor
Software, Interface, and Related Experience: A user who has more extensive knowledge with computers and is someone who may use a computer for professional reasons. Professional uses of a computer can range from creative outlooks such as

graphic design or video editing to more technical professions such as software development or database analyst.
Experience with Similar Applications: A user who has experience with command line, networking, Windows Control Panel.
Task Experience: User often uses the web browser for email, collaboration, and to research potential solutions to business needs. User could potentially send email suggesting applications that they would find useful for the website.
Frequency of Use: This user may stop by the website every so often to see if there is new software that they can incorporate into their environment.
Key Interface Design Requirements that the Profile Suggests: User will already have experience with many different layouts of websites and navigating will be second hand.

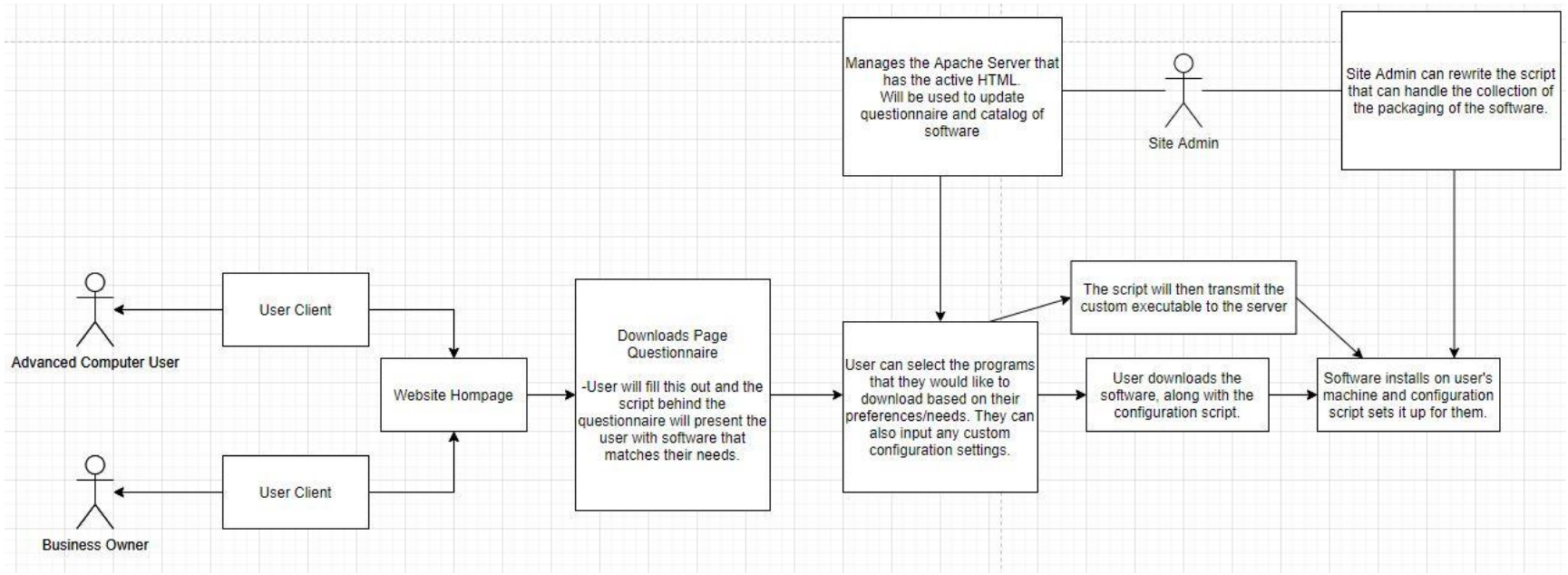
Table 3: Site Admin User Profile
Application: Azure, GitHub, Apache, HTML, CSS, Linux scripting
Potential Users: Site Admin
Software, Interface, and Related Experience: User should know the layout and configuration of both the backend and the frontend.
Experience with Similar Applications: User will need to use tools that are used for web application development and for client scripting. This will be mostly IDE's for editing HTML and software that configures executables on client systems. User will also need to be familiar with cloud-based computing such as Azure, Google Cloud, or Amazon Web Services.
Task Experience: The applications that are listed will be used for creating the application as well as maintaining the website and script.
Frequency of Use: For the initial creation of the web application the user will be using the mentioned development tools

daily and within multiple two-week sprints. After the release of the web application the user will only need to edit the catalog on the front end of what software is available and need to edit the backend only when they find a new application that is worthy to add.
Key Interface Design Requirements that the Profile Suggests: The user will need to understand programming conventions, syntax, and development processes to interact with the development tools to create this web application.

Use Case Diagram

Below is the Use Case Diagram. The users included are the Advanced Computer User and the Business Owner, who are both end users that are using the site to download software, either for home or business use. The third user is the Site Admin, who would be responsible for updating and maintaining the website.

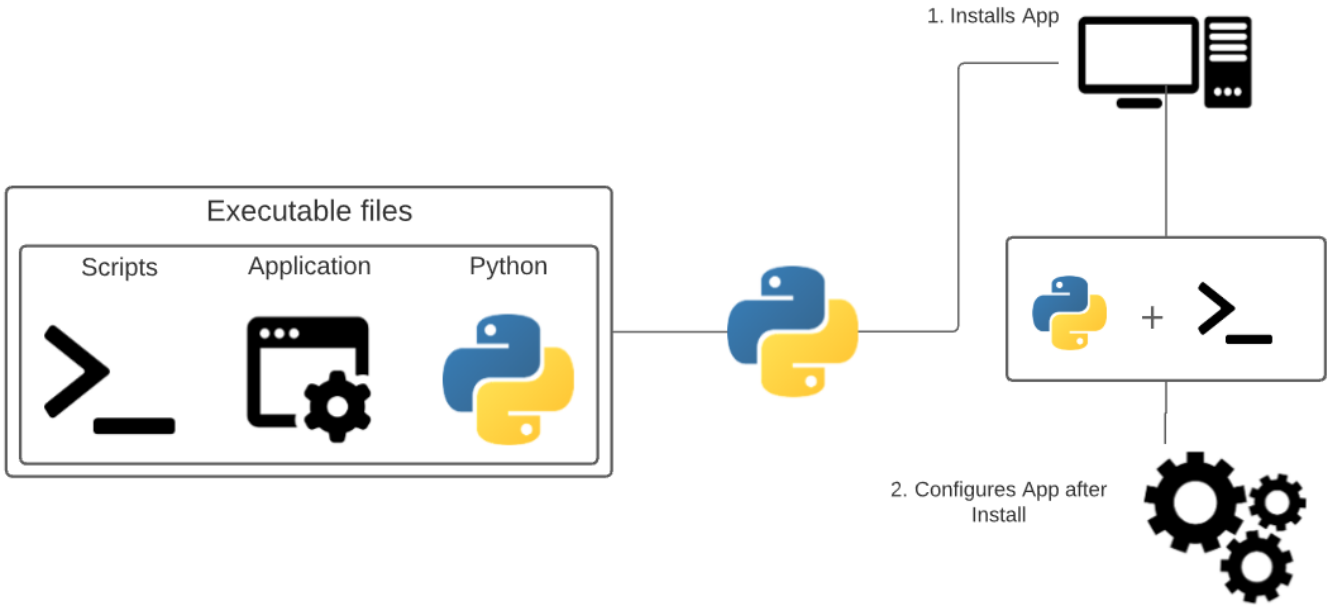
Figure 1: Use Case Diagram



Technical Architecture

Figure 2: Architecture Diagram of User connecting to SoHoIT.org

Figure 3: Architecture Diagram of Installation of Executable Hosted on SoHoIT.org



Testing

Testing Methodology & Approach

Testing SoHoIT started with our approach being broken up into two sections: the user-end, being our website, and the other section being the back-end. With the user-end consisting of our website, in which a client would be loading the site on their local browser via Windows PC. The back-end portion of our testing approach consisted of testing the pre-configured downloadable software installs we offer on the site. Since this project was developed for Windows users, specifically Windows 10 clients, we took the approach to test the pre-configured downloadable software installs on various Windows 10 client desktops. The end-goal of this approach to testing SoHoIT was to ensure the website functioned as intended and provided the user with pre-configured downloadable software installs for their small business.

Test Plan

User-End

For the user-end testing plan, we had the user first go to the static home page of the website, and from there we had the user navigate to the downloads page. The downloads page is where most of our project is held. This is where we had the user click on each of the offered software downloads and download it to their local machine. Keeping an eye out for any potential issues, once the downloads had completed, the team would then see if the user ran the executable files as administrator on their Windows PC. Since it is required for all the software to be ran as administrator on the user's PC, the team wanted to verify this in order to run an effective test. From there the user would then navigate to the tutorial page and follow the video tutorial we provide the user in order to complete the configuration. This consists of the step-by-step testing procedures in order to achieve our goals for the project.

OpenVPN

The back-end testing of our project consisted of pre-configured downloadable software installs offered on SoHoIT.org. The team tested running the executables on various Windows 10 clients and observed any errors and incorrect configurations. The pre-configured software installation for OpenVPN required the most testing on a user's end due to the timing of how the executable completes all its configuration tasks. To test OpenVPN, the team had to check on the test machine ensuring all the file dependencies for OpenVPN to operate were created and written to the correct files and directories.

ClamAV

ClamAV was a relatively easy product to test, we first started by downloading ClamAV as it was intended and running the configuration commands and program commands in PowerShell to see if the software worked as expected. Once we created the GUI we began testing its functionality. We started with the basics by testing button function and then worked on linking the commands to those buttons. We spent about 4 days troubleshooting how to get the scan results to appear on screen in the GUI. We tested the scan by testing directories on our own machines to see if the scan was accurate.

Test Results

User-End

Testing the user-end turned out quite well as the team did not experience any issues with the site loading for a user on their local browser. The main part of testing for the user was to get their insight on how they felt about the navigation and functionality of our downloads page. We had the user download OpenVPN from our downloads page and they expressed it was not apparent that the OpenVPN card was a click-able button. This user's insight brought the team to re-consider implementing

some UI changes on the downloads page to where if you hover over the card, it dynamically changes the fill color of that download card. Moreover, the team had the user run the executable for OpenVPN on their system to ensure that all the file dependencies were properly configured. There were a few tests where the user's OpenVPN executable created every file needed except for the client certificate. This issue was resolved during our back-end testing with OpenVPN which can be found below.

OpenVPN

After the team underwent extensive tests, the team were able to identify and resolve issues that arose with the user-end and the back-end. The biggest issue we came across was with testing OpenVPN. The pace in which the configuration tasks are completed gave the team some complications. This means that there were issues with certain files not being created and moved to their respective directories before the next line of code in the executable would run, causing the OpenVPN configuration install to be missing necessary files. In some of our testing among various Windows 10 client machines, we discovered the client certificate was not being created. Without the client certificate created, the user will not be able to connect to their OpenVPN hosted server, as the client certificate is needed in the client-connection for the server to trust the client. Appendix B is a snippet of the PowerShell script, where line 29 has "Start-Sleep –Seconds 5" added in order to dampen the time of how long it takes on a user's system to run the batch script within the executable. The team investigated various methods to essentially "pause" the script until each task was completed hoping to provide a better fix for these issues team encountered during testing. However, the team quickly realized there was no simple solution with PowerShell's functionality that allowed the team to pause the script until a file was indeed created. That is why the team decided to stick with setting a static sleep timer to the script to best combat the issues with the client certificate not being created. Concluding our testing, running the

OpenVPN executable on various Windows 10 client machines, we were able to iron out the issues with the client certificate not being configured.

ClamAV

After about a week of testing the team had ironed out all bugs with the ClamAV install and GUI. The GUI worked as expected. The results displayed on screen in the GUI, and all button and scan functionality worked as we wanted. The scripted installation loaded the pre-configured files that we wanted and created a desktop icon for the ClamAV GUI.

Budget

Below is our initial budget we had laid out in estimation and our finalized budget. Table 4 is an approximate representation in costs for a group of personnel to host this web site service. However, we had considered the budget based on what we were spending for the project, and as none of us were getting paid we left the labor costs as free. Table 5 below is our Fall Semester revised budget, written from the perspective of a company creating the project for themselves and hiring staff to create it with the same timeline that was used for this project. Table 6 is the Spring Semester Revised budget, changed to reflect the cost of doing this as a student project.

Table 4: Initial Project Budget

SoHoIT Initial Budget		
ITEM	QUANTITY	TOTAL
SERVICES		
Microsoft Azure	1	$50/month
Web Domain Name OR Personal Computer Hosted Web Server	1	FREE
Web Domain Name	1	FREE
LABOR		
Web Developer	~2	FREE
Client Script Developer	~2	FREE
TOTAL	MONTHLY	$50
	YEARLY	$600

Table 5: End of Fall Semester Revised Project Budget

SoHoIT End of Fall Semester Revised Budget		
ITEM	QUANTITY	TOTAL
SERVICES		
Amazon Web Services EC2 Virtual Machine	1	$11.83/month
Web Domain Name OR Personal Computer Hosted Web Server	1	FREE
Web Domain Name	1	FREE
LABOR		
Web Developer	~2	$20/hr 20 hrs/mo
Client Script Developer	~2	$20/hr 20 hrs/mo
TOTAL	MONTHLY	$811.83
	YEARLY	$9,741.96

Table 6: End of Spring Semester Revised Project Budget

SoHoIT End of Spring Semester Revised Budget		
ITEM	QUANTITY	TOTAL
SERVICES		
Amazon Web Services EC2 Virtual Machine	1	$11.83/month
Web Domain Name OR Personal Computer Hosted Web Server	1	FREE
Web Domain Name	1	FREE
LABOR		
Web Developer	~2	FREE
Client Script Developer	~2	FREE
TOTAL	MONTHLY	$11.83
	YEARLY	$141.96

Project Timeline

Below is Table 6, which is a timeline of the fall semester and our projected schedule for the upcoming spring semester. With a foundation and much of the research completed as a part of the fall semester, we expect that the spring semester will consist of development improvements and testing.

Table 7: Project Timeline

Task #	Task Name	Duration	Start Date	End Date
1.	Team Contract	1 Week	August 25th, 2020	September 1st, 2020
2.	Project Validation	6 Days	September 1st, 2020	September 7th, 2020
3.	Work-Flow Diagram	3 Days	September 7th, 2020	September 10th, 2020
4.	Workstation setup/Learning the Tools	4 Days	September 10th, 2020	September 14th, 2020
5.	Use Case Diagram	3 Days	September 14th, 2020	September 17th, 2020
6.	Registering Domain Name: SoHoIT.org	2 Weeks	September 17th, 2020	October 1st, 2020
7.	Setting Up Apache Web Services:	2 Weeks	October 1st, 2020	October 15th, 2020

	-Software Installs -Main Website/UI			
8.	Configuration Scripting: Packaging Selected Installs Automated Installations (User input processing)	2 Weeks	October 15th, 2020	October 30th, 2020
9.	Automated Installations running with pre-set config from user input on Questionnaire. Polishing Alpha	4 Weeks	October 30th, 2020	November 23rd, 2020
10.	Beta Build This would be building upon the applications we have chosen and	8 Weeks	November 30th, 2020	January 30th, 2021

	setting up the installer to be customizable to the input from the user.			
11.	Optimization: Making sure that there are no configuration issues and that this works for a wide arrange of environments such as different web browsers and operating systems.	4 Weeks	January 30th, 2021	February 28th, 2021
12.	Final Build/Testing	2 Weeks	March 1st, 2021	March 13th, 2021
13.	Prepare for Presentation/Demo. Also making sure that we have a backup instance that we could	1 Month	March 13th, 2021	April 12th, 2021

	readily deploy in case something happens.			
14.	Present	1 Day	April 13th, 2021	April 13th, 2021

Problems Encountered and Analysis of Problems Solved

The most significant ongoing problem we faced during the development and finalizing of this project was narrowing down its technical methods. When we say technical methods, we mean the very specifics of each moving part of the project to function smoothly. For example, approaching the most standardized and feasible method for pre-configured automated software installs was a big challenge. In the end we decided to go with utilizing Python and Python modules PyInstaller and Django. PyInstaller enabled us to package a software installation executable file with a Python script that sets certain pre-configurations and install switches to that executable. The best part about PyInstaller, and its main function, is that it allows you to take Python script code and package it altogether under a single executable file. This executable file also does not require Python on a user's system for the executable to successfully run. Our finalized approach to script automation was a big turning point in our project's developmental process as we found the feasibility in its application to other software.

Our group also struggled with the decision on how we would host a web site for users to access our service. This required us narrowing down our options after we realized Microsoft Azure virtual machines still charge a balance under "dormant" usage while the virtual machine is shut down. The final decision was to host the website via an Amazon Web Services EC2 t2.micro virtual machine. Luckily the first year using AWS EC2 virtual machines is free, so our team did not have to worry about taking costs into account.

Future Recommendations for Improvement

In the future, this project could be improved upon in two major ways. The first would be the addition of different software to cover different areas of IT solutions, such as remote file storage, network and system monitoring, and coding IDE's. These are just a few examples, but the list of programs available could be expanded to anything useful to customers so long as each addition met the site's requirements of being completely open source (as in, not needing a paid version for business use) and being configurable using scripts in order to help simplify difficult install processes for the users.

The second way would be to include alternate configuration scripts based on user input that is accepted through the website. This is covered in the Abandoned Goals section on page 8, but allowing install configuration through user input would give user's with more knowledge the ability to install these programs in ways that better fit their environment while still allowing users with less IT knowledge to use the default configurations that are currently implemented.

CONCLUSION

Considering our team all lacked proficient project management experience, working on the development of a web site has taught us quite a bit. We underestimated how complex things can get when learning new technology. When you consider all the options each piece of software offers, it was hard to predict the labor time needed. In simpler terms, this complicated our time management and posed a challenge for us.

During this project, our team has learned a lot about web development, scripting, time management, project management, and use-case scenarios. We learned about the variety of automated installation tools and services out there for people to utilize. Using the PyInstaller module was a learning experience. Discovering the unique features it offered from packaging a Python script file into an executable file, to implementing a custom icon image on the executable. We also obtained extensive knowledge about HTML, CSS, JavaScript, and web design aesthetics. Our team inherited more skills with

time management, project management, and the approach for our intended user case. To offer the most intuitive experience, we worked with various configuration options to offer to the user. That consisted of us figuring out how we wanted to present the software questionnaire to the user. As with any project, our team had to make compromises. Keeping our horizons open for the project helped us come to negotiations on how we want the user experience to be. We decided to provide the user with the questionnaire first, to get an insight on software recommendations.

Our focus for the spring semester is to fine tune the user experience and clean up the web design with pleasing aesthetics. We also aim to offer more software for the user to choose from on our web site. To achieve this, we will conduct extensive testing to ensure our project is at a state we are comfortable with.

References

DiGiacomo, John. "Cyber Attacks Are on the Rise for Small Businesses." *Revision Legal*, 23 Jan. 2017, revisionlegal.com/internet-law/data-breach/cyber-attacks-rise-small-businesses/.

Lee, J., & Runge, J. (2001). Adoption of information technology in small business: Testing drivers of adoption for entrepreneurs. *The Journal of Computer Information Systems, 42*(1), 44-57. https://search-proquest-com.proxy.libraries.uc.edu/docview/232580492?accountid=2909

*United States Small Business Administration Office of Advocacy, 2018 "*2018 Small Business Profile," Retrieved from https://www.sba.gov/sites/default/files/advocacy/2018-Small-Business-Profiles-US.pdf

Appendices

Appendix A

Technologies/Programs/Languages used in Project Creation.

Throughout the duration of the project several technologies and programs were used in its development. The list as follows:

Amazon Web Services EC2 T2.micro virtual machine – Hosted SoHoIT.org

Visual Studio Code – Integrated Development Environment (IDE) used to code in.

Django Web Framework – Software used to allow users to interact with SoHoIT.org, handles back-end configuration.

Apache Web Service – Opensource web hosting software offered on SoHoIT.org.

ClamAV - Opensource Antivirus Software offered on SoHoIT.org.

OpenVPN - Opensource Virtual Private Network (VPN) software offered on SoHoIT.org.

KeePass – Opensource credential management software on SoHoIT.org.

Python – Object-oriented coding language used in the development of SoHoIT's scripted installs and Graphical User Interfaces (GUI).

Windows PowerShell – Microsoft's .NET Framework-Based Scripting Language was used in SoHoIT to automate various configuration tasks for the OpenVPN pre-configured downloadable software install.

Appendix B

PowerShell Script for Prep-OpenVPNServer.ps1

```
16  (Get-Content -Path $easyrsaInit) -replace $binSh, './easyrsa --batch init-pki' | Set-Content -Path $easyrsaInit
17  Add-Content -Path $easyrsaInit -Value './easyrsa --batch --vars=vars build-ca nopass'
18  Add-Content -Path $easyrsaInit -Value '#./easyrsa --batch gen-dh'
19  Add-Content -Path $easyrsaInit -Value './easyrsa --batch build-server-full ServerVPN nopass'
20  Add-Content -Path $easyrsaInit -Value './easyrsa --batch build-client-full iOSclient nopass'
21  Add-Content -Path $easyrsaInit -Value 'bin/sh'
22
23  Set-Location -Path "C:\Program Files\OpenVPN\easy-rsa"
24
25  $EasyRSAstart =  "C:\Program Files\OpenVPN\easy-rsa\EasyRSA-Start.bat"
26
27  Start-Process -FilePath $EasyRSAstart
28  💡
29  Start-Sleep -Seconds 5
```