# PHISHING GUARD

by

Katie Drury, Serge Kikonda, Sarah Matevia

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



| | |
|---|---|
| Katie Drury, Serge Kikonda, Sarah Matevia | April 13th, 2020 |
| **Toni Iacobelli**, Faculty Advisor | April 13th, 2020 |

# PHISHING Guard

***A FUN AND INTERACTIVE WAY TO LEARN ABOUT AND GUARD YOURSELF AGAINST PHISHING ATTACKS***


Prepared by
Serge Kikonda, Katie Drury, and Sarah Matevia

University of Cincinnati
College of Education, Criminal Justice, and Human Services
April 2020

# Table of Contents

# LIST OF ILLUSTRATIONS

## Tables



## Figures

## ACRONYMS AND ABBREVIATIONS

**SSO** = Single Sign On

**MFA** = Multifactor Authentication

**UX** = User Experience

## ABSTRACT

According to Enjoy Safer Technology (ESET, 2019), 97% of people are unable to identify sophisticated phishing attacks. Though phishing attempts use scare tactics, website forgeries, or scams to increase their effectiveness and chances of success, research shows that the heart of the problem resides in a lack of effective training. Phishing Guard is a FREE, comprehensive, and easy to navigate web application designed to provide its users with the ability to receive interactive and effective training. In addition to creating a new approach to learning about phishing, this application will revolutionize the way these attacks are prevented. Phishing Guard has an easy to use menu system that includes short interactive quizzes, up to date videos, games, and more. These resources allow Phishing Guard to stand out among its peers, as it not only effectively decreases the success rate of phishing attempts but also increases user retention and awareness rate.

# INTRODUCTION

## Problem

Today, there are a large amount of problems in the digital world. Sadly, although well-known, phishing has remained one of the most effective forms of cyberattack faced by people throughout the years. These directed attacks do not affect a specific type of person. They are aimed at as many people, departments, organizations, companies, etc. as possible. They can range anywhere from very easy to very hard to detect. ESET (Enjoy Safer Technology) [3] claims that 97% of people are still unable to identify a more sophisticated phishing attack because they often use scare tactics to increase their effectiveness and chances of success. In a way, phishing has remained a never-ending and constantly evolving threat faced by all.

The 2019 Phishing Trends and Intelligence Report [2] conducted by PhishLabs states that phishing attacks grew by 40.9% in 2018. 98% of those attacks were email threats that led to identity theft, theft of sensitive information, email scams, and more. According to the 2018 Check Point Research Security Report [1], 64% of organizations have experienced a phishing attack in the past year. Unfortunately, there is not a silver bullet that can be used to completely eliminate phishing attacks. The problem is that most users are not properly trained to recognize phishing attempts. As a result, they often fall prey when attacked. Their lack of awareness and training will often lead them to click on links or open attachments in suspicious emails without ever considering the potential repercussions of their actions. The IT Governance Blog [4] reports that 52% of users receive training no more than twice per year, and 6% of users never receive security awareness training at all. Because of this, IT departments are affected the most and potentially impacted the worst. This problem has the ability to create a lack trust between an IT Department and users within an

organization because IT Staff cannot be fully confident in their users' ability to recognize, let alone report potential phishing attempts. Simply put, having a lack of trust (from the IT Department) and a lack of awareness and training (from users) creates an exploitable vulnerability within an organization that attackers are often able to take advantage of. So, how can this problem be solved?

## Solutions

To address the problem stated above, our team is introducing a new and unique product called Phishing Guard**.** It is a FREE, comprehensive, and easy to navigate web application. It is designed to provide its users with the ability to receive effective training about phishing and how to report it. Our goal is to ensure that the features within our application are tailored to our users in such a way that they enjoy and easily retain what they learn but, also feel the need to learn more.

Traditionally, IT Security awareness and training are conducted as follow:

1. Users are required to watch a long instructional videos (about 30 to 45 minutes long on average) or read the scripted version of a mandatory training.
2. Once the previous step is completed, they are required to take a quiz and pass with an 80% or above to claim that they have completed the training.

There are 2 key problems with this approach. First, the training provides too much information to its users. Rather than focusing on specific topics, they will generally talk about everything, at once. Second, users are often required to recall specific portions of the training to answer certain quiz questions. This is ineffective especially if a user had stopped paying attention 5 minutes into the training. At that point, the training becomes null and void, as it becomes somewhat of a guessing game for the user rather than the actual training it was intended to be.

This where our team and our product comes in. We believe that the best approach to this solve this problem is to provide users with **interactive training**. As ***Figure 1:*** **Phishing Guard Training Approach** below shows, our training is broken down into multiple sections covering various topics.

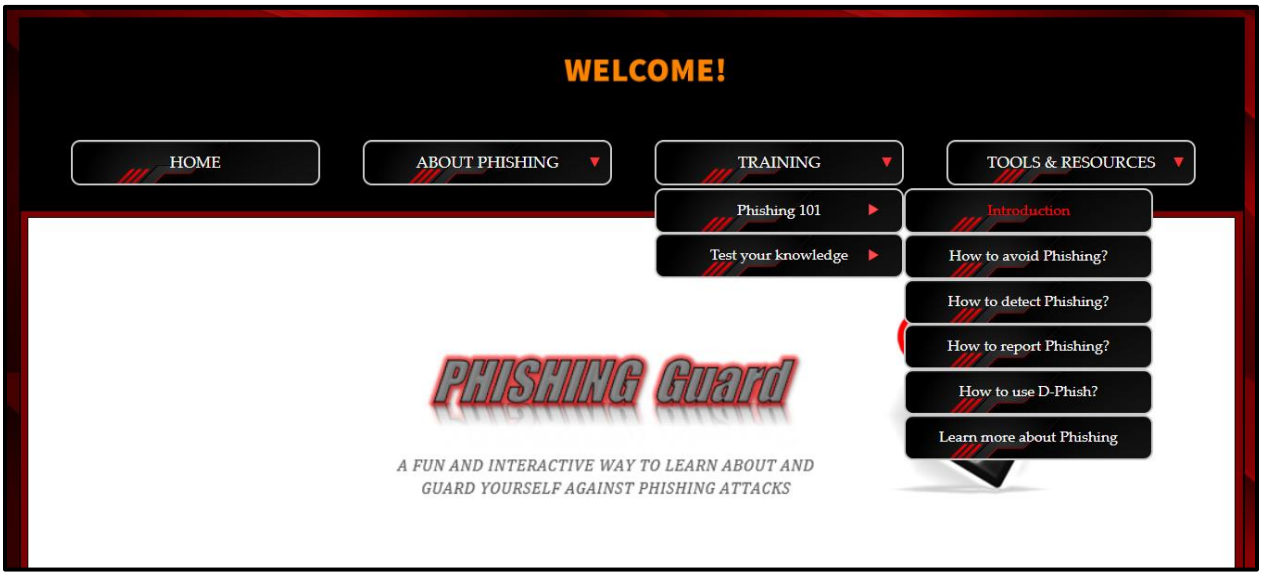



*Figure 1:* Phishing Guard Training Approach

Users are required to watch a short training videos (under 5 minutes long) focused on a specific topic. Then, after being fed the key information they are required to retain, they must take the **interactive quiz** at the end. These quizzes will only ask them questions about what they had learned and nothing more. This approach virtually eliminates a user's ability or desire to skip training.

Another form of interactive training provided within our platform is our interactive phishing games that can be played at any time. ***Figure 2:*** **Avoid being Phished mini-game** is an example of such games. "*Avoid being Phished*" is a game with a simple premise: Phishing attacks vary in complexity and effectiveness. Some are easy to avoid, others not so much. The goal is simple: avoid being phished for as long as you possibly can.

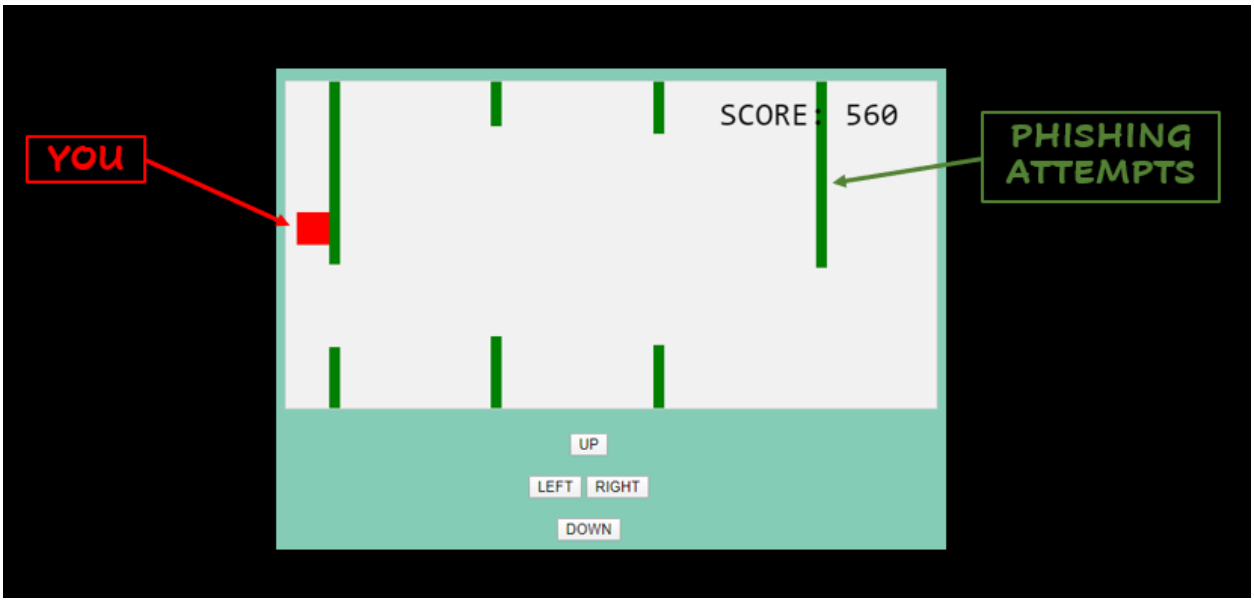



*Figure 2:* Avoid being Phished mini-game

Our product is unique in various ways. Upon research, we discovered some interesting facts about some of the other products that are attempting to tackle this issue. Take Lynda.com for example, the leading online learning platform in the world. It does provide training videos on phishing, but it does not solely focus on it. As such, there is no structure to their training. The content they provide is very generic and non-interactive. Another example would SANS Security. It does offer extensive training, thorough content, reporting and various other solutions. However, to receive them, users are required to buy their products after requesting a Training Demo first, via work email. In other words, this product is best suited for larger organizations and not individual users. Therefore, while both platforms mentioned previously, and many others, attempt to tackle the issues around phishing, not many truly attempt to provide long-term solutions. As a result, users do not receive the training they need to recognize and report phishing attempts. That is where our FREE, comprehensive, and easy to navigate product comes in.

In a nutshell, Phishing Guard is intended to become the platform that provides long-term solutions to the never-ending problem that is Phishing. It is not intended to be the answer to the problem because, as mentioned earlier, there is not a silver bullet that can be used to completely eliminate phishing attacks. It is designed to be a powerful resource that provides users with the confidence, training and awareness they need to combat phishing. Simply put, it is the armory that provides the equipment (training)  for users to fight the war against phishing.

## Project Goals & Brief Methodology

Develop a web application that allows users to receive effective and up-to-date information and training about various phishing attacks. Utilize various platforms and tools (AWS, JavaScript, Python, CSS, etc.) to make the content of the application appealing, interactive and easy to navigate to ensure that users are receiving effective training on how to detect phishing attempts, report them, and remain vigilant.

## Overview

The remainder of this final report outlines in detail how our project was conducted and completed. The report includes the following sections: Project and design concepts, methodology, user profile and use case diagram, technical architecture, testing, budget, timeline and Gant chart, problems encountered, and future recommendations.

# DISCUSSION

## Project Concept/Solution

Phishing Guard was inspired by Katie Drury, Sarah Matevia, and Serge Kikonda. Sarah and Katie already worked for the UC Office of Information Security and saw the need to do a project on phishing. Having already worked with Sarah in the past, Serge decided to reach out to Sarah because he wanted to do his project on Spamming. Being relatively close and similar to each other, the three decided to join forces to create something unique that would revolutionize the way phishing attacks are prevented. Prior to the first day of classes, the team met to try and narrow down the scope of the project. Given the list of ideas and suggestions we had, we decided to brainstorm problems in the Information security world and potential features for the application. We determined that there is a significant lack of proper and effective training, a serious problem with both the fact that training is provided on average about twice a year and that users are unable to retain key information provided during the training sessions they attend. There was also a dire need to improve the user experience during training. Hence why the group decided on the idea to create an application that will try and fix all those issues. Thus, Phishing Guard was established.

## Design Concept

Initially, we wanted to do a lot of things with Phishing Guard. We believed that the features listed below would provide an interactive platform that would give users an incentive to learn more about phishing. We thought that combined, they would provide users with the ability to receive phishing

awareness training and any other information related to the topic at any time. They would also enable them to detect phishing attempts with ease and report them wherever they might be.

- Interactive training and quizzes;
- Meaningful Games;
- Feedback/comment section;
- Reporting Mechanisms;
- SSO and DUO Authentication;
- Visually appealing graphical interface web application;
- Compatibility and operability with all devices and platform;
- Backend development written in Python and JavaScript;
- Web Application Health Monitoring through AWS.

Unfortunately, due to time constraints and unforeseen circumstances, we were forced to narrow the scope of our project. As a result, the following had to be abandoned:

- Integrating machine learning;
- Developing a mobile application;
- Developing phishing games with graphics;
- Fully integrating SSO and DUO Authentication*;
- Learning PHP to develop the comment and feedback section; and
- Making the application compatible with all devices and platforms.

* *This feature was fully developed with Python version 2 and integrated within our application by the end of 2019. Unfortunately, at the beginning of 2020, version 2 was no longer being supported and version 3 was rolled out. Sadly, our feature could not easily be transferred to this version.*

## Methodology/Technical Approach

We wanted to ensure that the design requirements and procedures of our application are deliverable in a timely manner, given the strict timeline of our project. Therefore, our procedure and approach was simple. We knew that we had a lot of stretch goals we could achieve but, had determined key features and functionalities we would love to have and showcase for our final product.

We used an **agile** project management approach. We set up multiple scrum sessions throughout the week to work through the project and ensure we were on track. We made sure that we kept each other updated on what we were working on. Furthermore, we set a coding cut-off deadline date (March 10th, 2020) for 2 reasons:

- Making sure we get all the features we promised to deliver coded into our platform; and
- Making sure we have enough time to fully test our application to ensure that we have a working product that is fully operational and does not contain bugs;

This methodology and technical approach proved to be effective for our project. It enabled us to achieve all the goals we had set for ourselves, gave us more than enough time to refine our application to peak performance, and even go slightly beyond the scope of our project and complete some of the stretch goals we had set for ourselves.

## User Profile

***Table 1*: User Profile – UC Students, Faculty, Staff** and ***Table 2*: User Profile – Admins** below illustrate the user profile for Phishing Guard. They describe potential users of our web application and similar applications. They also explain key interface design requirements that the user will encounter.

**User Profile – UC Students, Faculty, Staff**

The User profile is associated with the University of Cincinnati Community (UC Students, Faculty, and Staff). The user profile for students, faculty, and staff is primarily used for user awareness in regards to phishing. This role will also be inherited by the admin profile. The table below showcases the user profile form for UC students, faculty, and staff.

| **User Profile Form** |
| --- |
| **Project:** Phishing Guard<br>A web application that provides the UC Community the ability to be kept up to date about the different phishing attacks happening within UC, receive effective and interactive training on how to detect phishing attempts, how to report them, but also how to remain vigilant. |
| **Potential Users:**<br>UC Students, Staff, and Faculty. |
| **Software, Interface, and Related Experience:**<br>All Phishing Guard users should have experience with using email, web applications, and Duo Multifactor Authentication (MFA). Single Sign On (SSO) with UC credentials and DUO MFA will be required to access Phishing Guard. |
| **Experience with Similar Applications:**<br>o Lynda.com<br>o Sans Securing the Human<br>o McAfee Training<br>o Phishing Box.com<br>o Outlook<br>o Google Mail<br>o Yahoo! Mail |

| |
|---|
| **Task Experience:**<br><br>At initial opening of the application users will see many options for them to choose from. There will be an option to start training(s). They will be able to view progress and quiz grades. It will also show common phishing attempts currently being seen at the University while having a reporting mechanism which reports suspicious links or files to Abuse@uc.edu.<br><br>Users will have prior experience:<br><br>o Using a web application from their preferred browser<br>o Reporting suspicious emails to Abuse@uc.edu |
| **Frequency of Use:**<br><br>This application, in its beta phase, is intended to be used on release to help train users on phishing attempts and whenever they are unsure about emails that come through their inbox. |
| **Key Interface Design Requirements that the Profile Suggests**:<br><br>o Simple, intuitive UI<br>o Fluid/Responsive design for accessibility on multiple devices<br>o Visually pleasing presentation with the use of images, hierarchical text formatting, etc.<br>o Engaging layout (e.g., "wizard like" survey)<br>o Engaging videos<br>o Authentication (SSO and DUO MFA)<br>o Reporting Mechanism for suspicious URLs or Files |

*Table 1*: User Profile – UC Students, Faculty, Staff

**User Profile – Admins**

The admin user profile is associated with the individuals who created Phishing Guard (Katie Drury, Sarah Matevia, and Serge Kikonda). This role is inherited from the UC Students, Faculty, and Staff user profile. Additionally, admins will have the ability to manage the web application. They will update the content with recent phishing attempts that occurred at UC or attempts that the community should be aware of. The table below showcases the user profile form for admins.

**User Profile Form**

**Project:** Phishing Guard

A web application that provides the UC Community the ability to be kept up to date about the different phishing attacks happening within UC, receive effective and interactive training on how to detect phishing attempts, how to report them, but also how to remain vigilant.

**Potential Users:**

Creators of Phishing Guard

- Katie Drury
- Sarah Matevia
- Serge Kikonda

**Software, Interface, and Related Experience:**

All Phishing Guard admins should have experience with using email, web applications, and Duo Multifactor Authentication (MFA). The admins have related experience with Python, HTML, AWS, and determining phishing messages/attempts.

**Experience with Similar Applications:**

- Outlook
- VirusTotal
- Zscaler (Zulu URL Risk Analyzer)
- Other application inherited by UC Students, Faculty, and Staff user profile

**Task Experience:**

Users will have prior experience:

- Using a web application from their preferred browser
- Determinants of Phishing attempts
- Python
- HTML

| |
|---|
| **Frequency of Use:**<br><br>This will be used when admins need to update contents related to phishing. Since the admins are also part of the UC community, they will also inherit the frequency of use same as the UC Students, Faculty, and Staff user profile. |
| **Key Interface Design Requirements that the Profile Suggests:**<br><br>o Simple, intuitive UI<br>o Fluid/Responsive design for accessibility on multiple devices<br>o Visually pleasing presentation with the use of images, hierarchical text formatting, etc.<br>o Engaging layout (e.g., "wizard like" survey)<br>o Engaging videos<br>o Authentication (SSO and DUO MFA)<br>o Reporting Mechanism for suspicious URLs or Files |

*Table 2*: User Profile – Admins

## Use Case Diagram

***Figure 3:*** **Use Case Diagram** below displays the use case for Phishing Guard. The diagram shows all potential users along with the corresponding tasks each user will have when interacting with the web application.

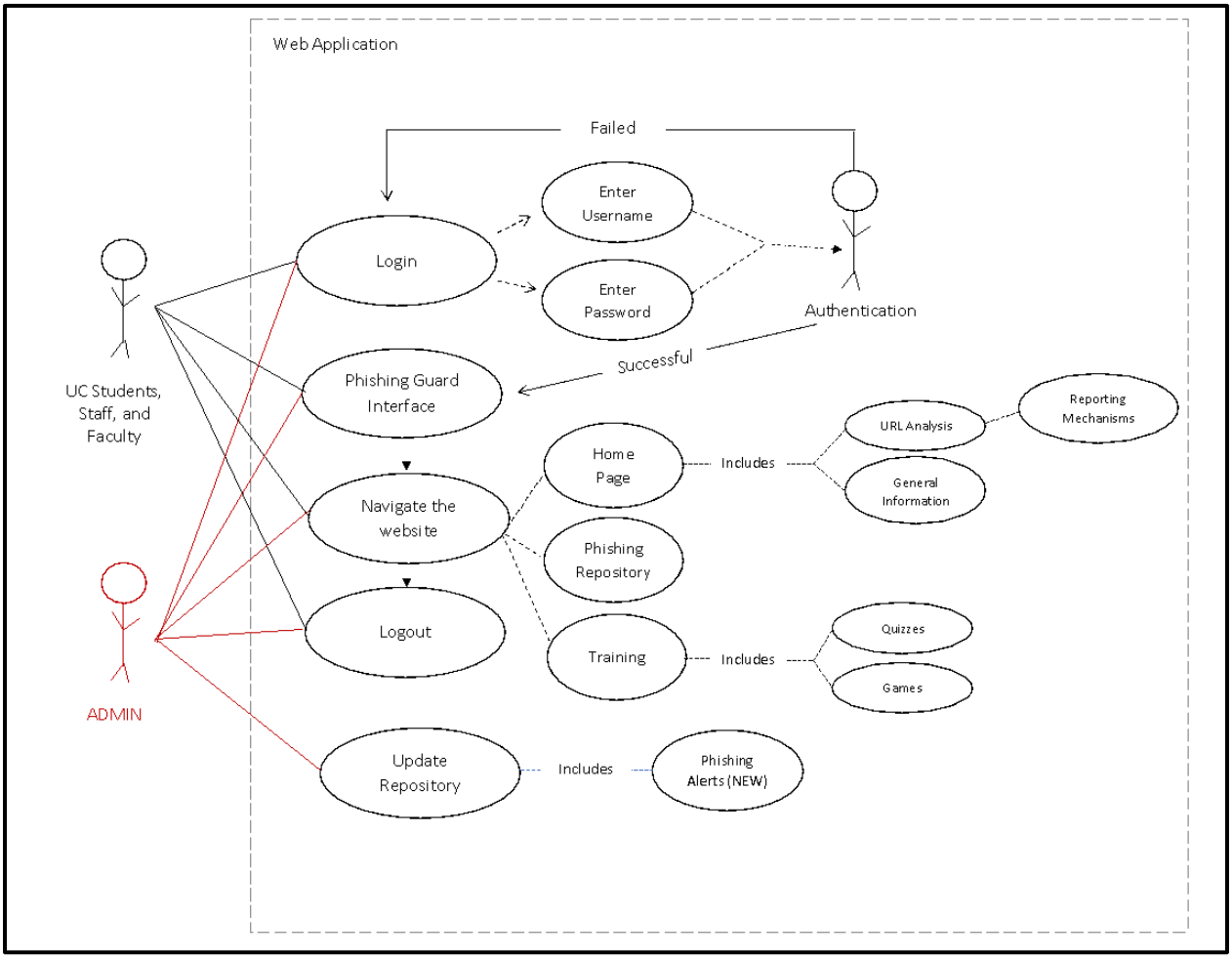



*Figure 3:* Use Case Diagram

## Technical Architecture

- **Software Requirements**

Phishing Guard is aimed towards organizations that provide security and awareness training through web applications. Prior to using this application, our team expects users to know the basics of navigating through web applications. Simply put, just like Amazon, eBay, or any other web applications they may visit, users should have some experience navigating through web pages to find information. Please refer to Appendix B to see the full technical diagram.

- **User Interface**

***Figure 4*: Phishing Guard Home Page** below showcases the homepage of our platform. It is designed to provide our audience with information about what the platform is, why it may be beneficial for them to go through it, and more.

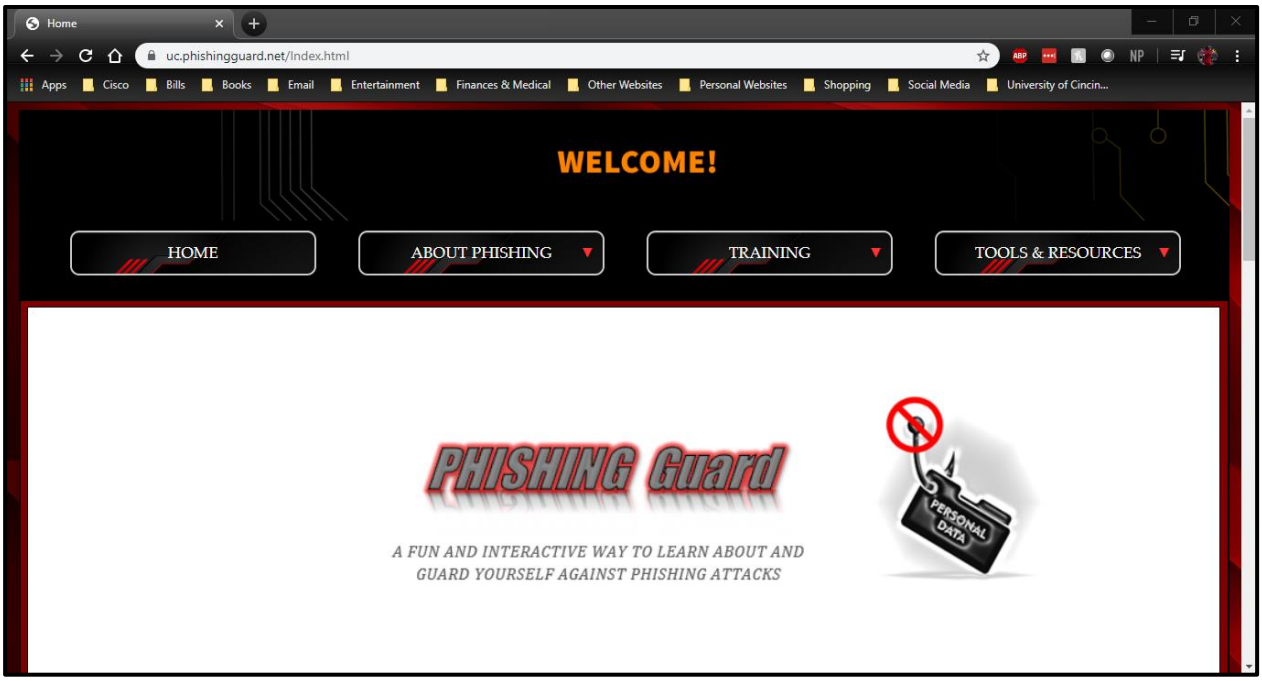



*Figure 4*: Phishing Guard Home Page

Diving deeper into our web application will provide users with a plethora of information and choices. For instance, as ***Figure 5*: Hovering over the About Phishing Tab** showcases, when hovering over the About Phishing tab, users will be prompted to decide whether they want to learn about what phishing is or if they want to learn more about it.

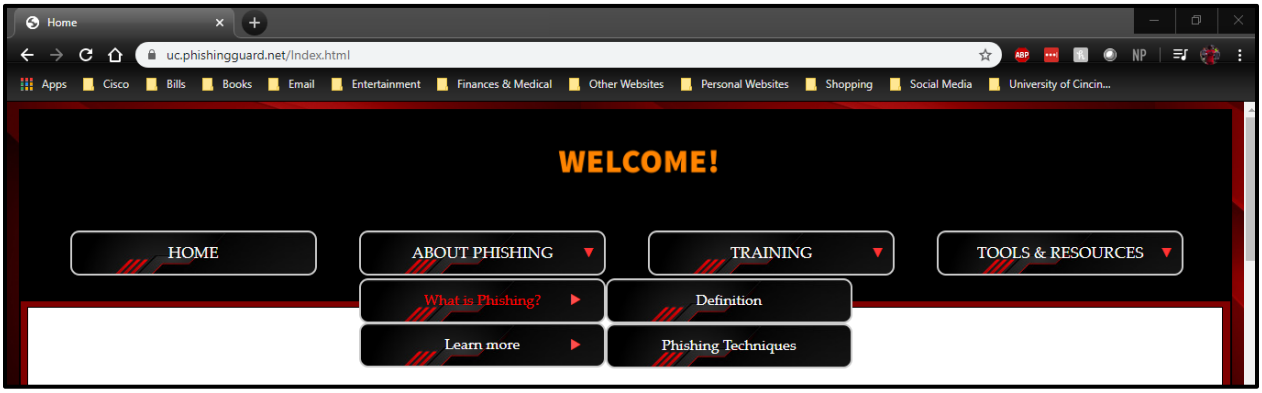



*Figure 5*: Hovering over the About Phishing Tab

Similarly, when hovering over the most important tab, as shown in ***Figure 6:*** **Hovering over the Training Tab**, users will be presented with 2 initial choices, that will expand and provide specific training they can choose to receive. We believe that this approach gives users the ability to choose the training they wish to receive, fully knowing they have to complete them all regardless.

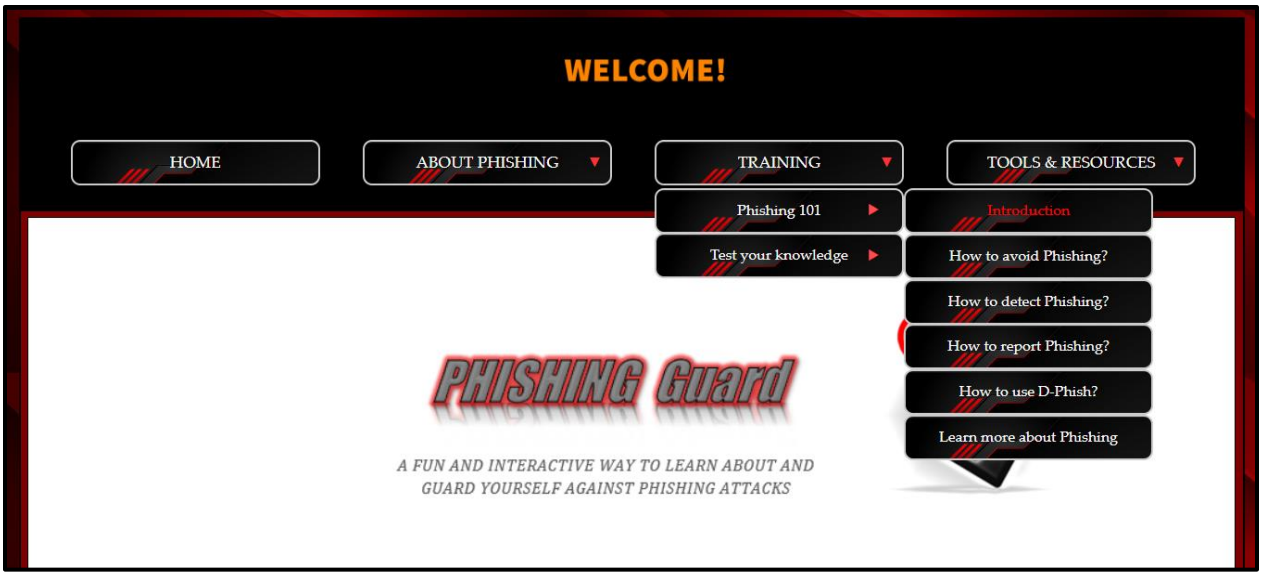



*Figure 6:* Hovering over the Training Tab

- **Network**

Phishing Guard is hosted on AWS. This is a cost effective solution and is very easy to scale. Katie Drury, the project manager, created a joint account so that all team members could have admin access to the server. Hosting our application through AWS gives us the benefit of having all of our computer networking requirements handled by AWS. This choice greatly reduced our team's workload and overhead, enabling us to focus more on the development of key features for our application.

- **Application**

The front-end of our application was coded primarily in HTML and CSS while the backend utilized JavaScript and Python. The GitHub repository ensured that our code was properly handled and went through multiple reviews prior to being published to our production environment. Each push submitted had to be thoroughly tested and verified prior to integration.

- **Security**

Least privileged access was implemented for our AWS platform. Implementing least privilege access enabled us to reduce the security risk and impact that could result from errors or malicious intent. Furthermore, Phishing Guard intended to utilize the University of Cincinnati's (UC) Single Sign On (SSO) and Duo Multi-Factor Authentication (MFA) platforms. As mentioned earlier, this security feature was fully integrated into our application using Python version 2 which, at the time of development was still supported. We were not aware that it would become legacy at the beginning of 2020. And, even though our team was unable to fully transfer this feature into to Python version 3 before our coding cut-off deadline, we were still able to integrate this feature into our application. However, this came at the cost of sharing a suspicious looking hyperlink to users:

- Regular link to we would send to our users: https://uc.phishingguard.net/Index.html

- Link we would have to send our users if we wanted them to use the security feature: http://phishingguardfinal.5q7mx4pjwp.us-east-1.elasticbeanstalk.com/login

This was a difficult decision to make. However, we ultimately decided against providing our users with a link whose legitimacy will be put in question. Instead, we focused our efforts on obtaining a security certificate to show our users that our platform is private, secure, and can be trusted. ***Figure 7*: Phishing Guard Security Certificate** below showcases the additional layer of security that was intended to be provided within our application alongside SSO and Duo MFA.

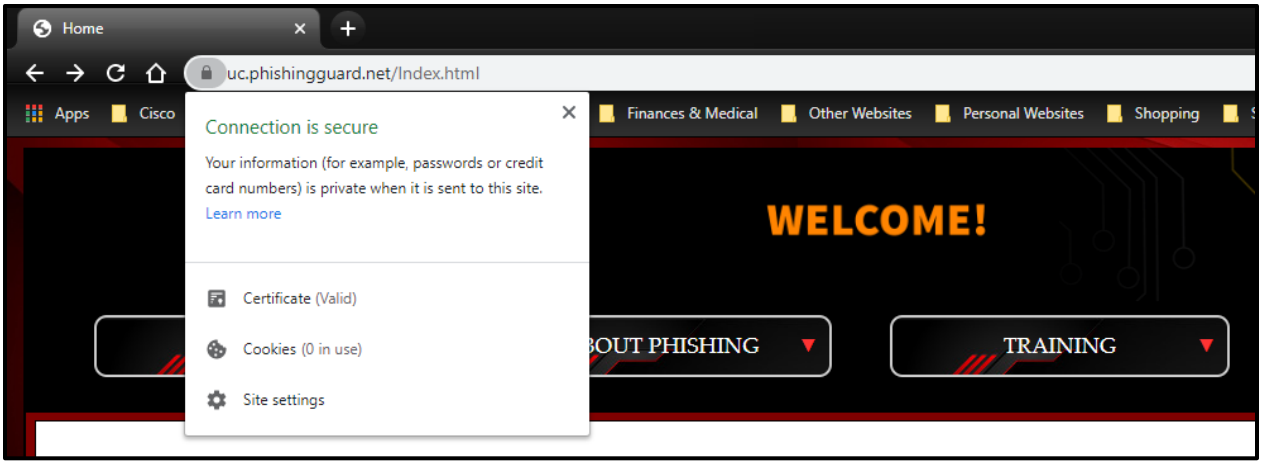



*Figure 7*: Phishing Guard Security Certificate

The SSO was supposed to only allow UC Students, Faculty, Staff and Alumni to sign into Phishing Guard and prevent unauthorized access. Its purpose was to prevent potential "attackers" from gaining insight of ongoing phishing campaigns at the University. The Duo MFA was intended to verify the identity of all users with a strong two-factor authentication before granting them access to Phishing Guard. Its purpose was to decrease the risk of compromised users gaining access to our web application. Finally, the security certificate would be added to bring all security features together.

## Testing

- **Overview**

The testing plan covers the functionalities of Phishing Guard that were fully tested before deployment. The testing was based off of the initial scope incorporated in our contract and the key features that our application was expected to deliver.

- **Scope**

The scope of our testing it to test the security and functionality in the following features:

1. Site access secured with modern authentication tools, such as SSO and DUO;
2. Reporting Mechanisms;
3. Continuous platform updates;
4. Web App health monitoring; and
5. Ensure the training (videos, games, quizzes, general information) work properly.

- **Objective**

The objective of having a test plan is to ensure that the team can properly track features that are and are not working up to par with testers.

- **Test cases**

The following are test cases:

- Elastic Beanstalk deploys user created python code to an existing AWS environment (Phishing Guard)
- The test run will confirm implementation is successful for the following

    - SSO
    - DUO MFA
    - Web App Health Monitoring and alarms
    - Root access policies
    - Monitoring
    - Reporting Mechanisms
- If any deployment fails, all created resources will roll back to original state

- **Entry and Exit Criteria**

**Entry Criteria**

- Phishing Guard successfully running
- Deployment is successful
- Scripts configured with AWS Elastic Beanstalk

**Exit Criteria**

- All tests are executed
- No errors in Python script or errors in health monitoring
- Bugs are fixed and documented

- **Logging Test and Reporting**

Our approach to testing was very simple. If any errors occurred during testing, they were to be reported, documented, fixed, and retested to see if it would reoccur. Our team met twice a week to discuss errors, bugs, or issues to prioritize and quickly remediate them.

Thanks to this approach, our code was thoroughly tested. On the one hand, most pages contain no errors, as they have all been fixed, as shown in ***Figure 8:*** **Phishing Guard Homepage Validation**.

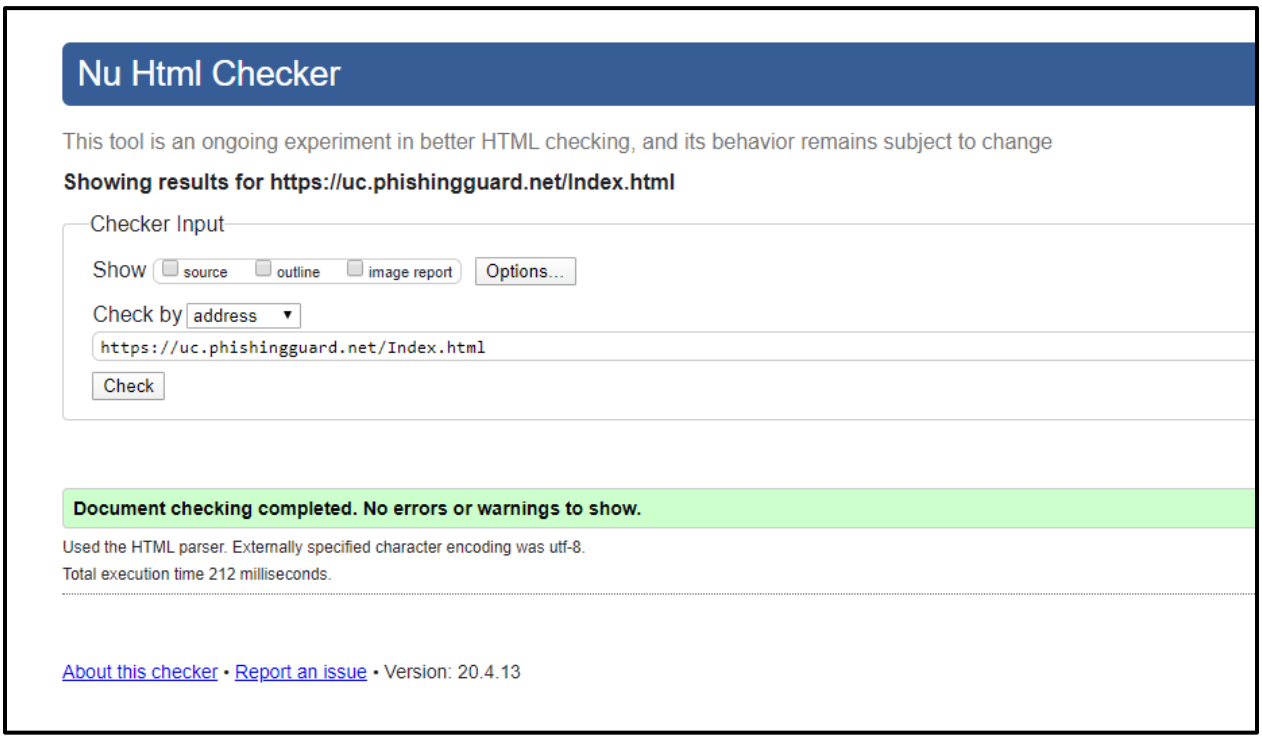



*Figure 8:* Phishing Guard Homepage Validation.

On the other hand, some pages will contain errors that we have decided to leave. For instance, as the title implies, ***Figure 9:*** **Common Errors Found during Testing** below showcases some of the common errors that are going to be seen across several pages in our platform. Most of these errors are subjective. For example, though *iframe* is obsolete, YouTube (the platform we used to link most of our training videos) still uses it if people wish to embed a video into their platform. As such, though this error can easily be fixed by using CSS, as suggested, we determined that doing so would create more problems than it would solve. As a result, we decided to accept this "bug" given as though the platform, videos, and content were not affected in any way, shape or form.


*Figure 9:* Common Errors Found during Testing

- **Testing Procedures**

**UX Test:** UX test cases and results display the individuals who tested each item, their expectation of the test outcome, the test results, and what could be improved. ***Table 3:*** **UX Test cases and results** below showcases each test and the associated results.

| Tester | Item | Expectations | Actual | Improvements | Pass/Fail |
|---|---|---|---|---|---|
| Cindy Lusby | SSO and DUO | Allow them to login with UC credentials and DUO Multifactor Authentication | Allows user to login with CLS and DUO | None | Pass |
| Collin Hater | Stability (UX) of web application | Expected the web application to flow nice, and work without any errors | Overall, my experience using this site was pleasant. The animations worked, navigation was fairly simple, and the page itself is intuitive. | Minus a few rough edges around aesthetics and spelling, the site is very well made, and it works properly, allowing an interactive way for UC to educate students about phishing and how to prevent it. | Pass |
| Stephen Perry | Stability (UX) of web application | expected web app to be easy to navigate and pleasant to look at | It seems to work, fine, but i did not spend too much time on the site to check how stable the connection is. That being said it really looks like it is about to crash any second, but that could just be the 2008 website aesthetics. | Update web design | Pass |
| Christian Stidham | SSO and DUO | Allow them to login with UC CLS and DUO | allowed them to login with UC CLS and then authenticate with DUO | None | Pass |

*Table 3:* UX Test cases and results

**Functionality Test:** Functionality test ensures the URL analysis lets end users upload and scan files or URLs. It also ensures that the reporting mechanism properly reports the suspicious files or URLs to the appropriate party. ***Table 4:*** **Functionality test case and results** below showcases each test and the associated results.

| Tester | Item | Expectations | Actual | Improvements | Pass/Fail |
|---|---|---|---|---|---|
| Anthony B | D-Phish (URL Analyzer) | Expected it to analyze URL | It performed as expected. | None | Pass |
| Cindy Lusby | D-Phish (URL Analyzer) | Expected it to analyze URL | Rate as a 10. It did what was expected and easy to use. | None | Pass |
| Holly Drury | D-Phish (URL Analyzer) | Expected it to analyze URL | It analyzed the URL as they expected. They did not expect it to take them to another site, but they do like that it provides all of the different detections and whether the URL is clean or malicious. | Open up VirusTotal in new tab | Pass |
| Collin Hater | D-Phish (URL Analyzer) | Expected it to analyze URL | They did not expect to be thrown to another site to check the URL. This is not necessarily a problem because it does what it is supposed to do. | Would personally prefer VirusTotal to be opened in a new tab so they do not have to backtrack to get back to the Phishing Guard site. | Pass |
| Dwayne Lewis | D-Phish (URL Analyzer) | Expected it to analyze URL | worked as expected. | No improvement suggestions | Pass |
| Stephen Perry | D-Phish (URL Analyzer) | Expected it to analyze URL | Seems pretty good. | They wish it would open the results tab in a new page though, so that they wouldn't be stuck clicking the back button to return to the website. | Pass |

*Table 4:* Functionality test case and results

- **Pass/Fail Conditions**

Our approach to the Pass/Fail conditions was very simple: is the feature doing exactly what it was developed to accomplish (Yes or No)? After thorough testing, we determined that all key features developed within the Phishing Guard web Application have passed all tests conducted. As mentioned earlier, whenever a feature failed to meet our predefined conditions, it was immediately reported, and the team would take the appropriate actions required to fix the issues, errors, or bugs.

## Budget

***Table 5*: Project Budget** illustrates the potential budget of our project. Materials and labor were the two expenses our project mainly encountered. In order to keep cost down, the backend was hosted using Amazon Web Service's (AWS), Elastic Beanstalk free trial and Flask (python). All frameworks and development tools used to create the application were free or open source. Included in the table below is what we anticipated the simulated wage cost to be equal to, under the assumption that each programmer and developer would earn at least $22/hr. The result was $23,760. However, in actuality, the true cost of the project is $0 since we are using free tier trials and open source.

| CATEGORY | ITEM | DESCRIPTION | SIMULATED COST | ACTUAL COST |
|---|---|---|---|---|
| Materials | Hardware | Computer servers and devices used to create, test, run, and deploy the application. | $0 | $0 |
| | Software | Frameworks and development tools used to develop the application. | $0 | $0 |
| Labor | Conferences and Meetings | Funding for employee collaboration, learning events, and meetings. | $0 | $0 |
| | Actual Wage Costs | The actual wages that will be distributed for development of the project. | $0 | $0 |
| | Simulated Wage Costs* | The predicted wages that would be distributed if the project were done in the current market. (We will assume 2 programmers making $22 USD / hour and 1 developer making $22 USD / hour). | $23,760 | $0 |
| **TOTALS** | | | **$0** | **$0** |

*Table 5*: Project Budget

**Simulated Wage Costs are not included in the cost totals.*

## Project Timeline and Gantt Chart

***Table 6:*** **Project Objectives/Deliverables Due Dates** below illustrates the specific timeline of the entire project. It showcases what our team accomplished each week prior to IT Expo.

| Task Name | Duration (in Days) | Start Date | End Date |
|---|---|---|---|
| **I. PROJECT MANAGEMENT AND DELIVERABLES FALL SEMESTER** | **116** | **08/18/19** | **12/14/19** |
| 1.1. Team Building | 7 | 08/18/19 | 08/25/19 |
| 1.1.1. Ideas and Brainstorming | 7 | 08/25/19 | 09/01/19 |
| 1.2. Fall Semester Assignment 0: Team Members & Project Name | 7 | 09/01/19 | 09/08/19 |
| 1.2.1. Phishing Statistics for problem statement | 2 | 09/08/19 | 09/10/19 |
| 1.2.2. Revision of Problem Statement and Solution | 2 | 09/08/19 | 09/10/19 |
| 1.2.3. Approval of Problem Statement and Solution | 4 | 09/15/19 | 09/19/19 |
| 1.3. Fall Semester Assignment 1: Team Contract | 13 | 09/10/19 | 09/23/19 |
| 1.3.1. Team Work Breakdown | 3 | 09/10/19 | 09/13/19 |
| 1.3.2. Project Logo and Branding | 3 | 09/19/19 | 09/22/19 |
| 1.3.3. Gantt Chart | 12 | 09/10/19 | 09/22/19 |
| 1.3.4. Approval for use of SAML and DUO Multifactor Authentication | 10 | 09/18/19 | 09/28/19 |
| 1.3.5. Team Meeting with Advisor | 1 | 09/23/19 | 09/23/19 |

| | | | | |
|---|---|---|---|---|
| 1.4. | Fall Semester Assignment 2: Project Abstract for Tech Expo | 7 | 10/01/19 | 10/13/19 |
| | 1.4.1. Project Abstract for Tech Expo Draft | 9 | 10/01/19 | 10/10/19 |
| 1.5. | Fall Semester Assignment 3: Team Contract Resubmission | 7 | 10/01/19 | 10/13/19 |
| | 1.5.1. Team Contract Review | 5 | 10/05/19 | 10/10/19 |
| 1.6. | Fall Semester Three - Minute Elevator Speech | 10 | 10/10/19 | 10/21/19 |
| | 1.6.1. Brainstorm Elevator Speech | 3 | 10/11/19 | 10/14/19 |
| | 1.6.2. Practice Elevator Speech | 5 | 10/14/19 | 10/19/19 |
| 1.7. | Fall Semester Assignment 4: User Profile | 10 | 10/11/19 | 10/21/19 |
| 1.8. | Fall Semester Assignment 5: Use Case Diagram | 12 | 10/09/19 | 10/21/19 |
| 1.9. | Fall Semester Assignment 6: Draft Report | 14 | 10/21/19 | 11/04/19 |
| 1.10. | Fall Semester Assignment 7: Final Fall Semester Report | 21 | 11/11/19 | 12/02/19 |
| 1.11. | Fall Semester Oral Presentations | 14 | 11/18/19 | 12/02/19 |
| | 1.11.1. Presentation Practice | 5 | 11/27/19 | 12/02/19 |
| **II.** | **PROJECT MANAGEMENT AND DELIVERABLES SPRING SEMESTER** | **107** | **01/13/20** | **04/30/20** |
| | 2.1 Spring Semester Assignment 1: Testing Plan/report | 27 | 01/24/20 | 02/10/20 |

| | | | | |
|---|---|---|---|---|
| | 2.2 Spring Semester Assignment 2: Abstract | 7 | 02/20/20 | 02/17/20 |
| | 2.2.1 Project Abstract for Tech Expo Draft | 7 | 02/20/20 | 02/17/20 |
| | 2.3 Spring Semester Assignment 3: Draft Tech Expo poster due | 15 | 02/16/20 | 03/02/20 |
| | 2.4 Spring Semester Assignment 4: Final Tech Expo poster due | 15 | 02/23/20 | 03/09/20 |
| | 2.5 Spring Semester Final Presentations | 14 | 03/23/20 | 04/06/20 |
| | 2.5.1 Presentation Practice | 5 | 04/01/20 | 04/06/20 |
| | 2.6 Spring Semester Assignment 5: Final Report | 15 | 03/22/20 | 04/06/20 |
| | 2.7 Spring Semester Assignment 6: Safe Assign for Final Report | 15 | 03/22/20 | 04/06/20 |
| | 2.8 Tech Expo | 10 | 04/03/20 | 04/13/20 |
| | 2.8.1 Tech Expo Exhibit and Preparation | 7 | 04/06/20 | 04/13/20 |
| | 2.9 Spring Semester Assignment 7: Final Library Copy | 15 | 04/05/20 | 04/20/20 |
| **III.** | **RESEARCH** | **35** | **09/10/19** | **10/15/19** |
| 3.1. | Statistics | 10 | 09/10/19 | 09/20/19 |

| | | | |
|---|---|---|---|
| 3.1.1. University of Cincinnati Phishing Statistics | 10 | 09/10/19 | 09/20/19 |
| 3.1.2. Other known institutions phishing statistics | 10 | 09/10/19 | 09/20/19 |
| 3.2. Software Requirements | 30 | 09/20/19 | 10/20/19 |
| 3.2.1. Research Web Application Hosting Server | 15 | 09/20/19 | 10/05/19 |
| 3.2.2. Research Web Application Database | 15 | 09/20/19 | 10/05/19 |
| 3.2.3. Determine Front End Development Languages | 10 | 10/05/19 | 10/15/19 |
| 3.2.4. Determine Back End Development Language | 10 | 10/05/19 | 10/15/19 |
| 3.2.5. Determine Video Editing Software | 5 | 10/15/19 | 10/20/19 |
| 3.3. Networking Requirements | 27 | 09/23/19 | 10/20/19 |
| 3.3.1. Research Load Balancing | 14 | 09/23/19 | 10/07/19 |
| 3.3.2. Research Auto Scaling | 14 | 09/23/19 | 10/07/19 |
| 3.3.3. Research Application Health Monitoring | 14 | 10/06/19 | 10/20/19 |
| 3.4. Security Requirements | 26 | 09/30/19 | 10/25/19 |

| | | | | |
|---|---|---|---|---|
| | 3.4.1. Research SSO | 7 | 09/30/19 | 10/07/19 |
| | 3.4.2. Research DUO Authentication | 7 | 10/07/19 | 10/14/19 |
| | 3.4.3. Research Monitoring, Logging, and Tracing capabilities | 10 | 10/07/19 | 10/17/19 |
| | 3.4.4. Updating Web Application | 10 | 10/15/19 | 10/25/19 |
| | 3.4.5. Research Compliance | 10 | 10/15/19 | 10/25/19 |
| **IV.** | **DESIGN CONCEPTS** | **15** | **10/20/19** | **11/05/19** |
| 4.1. | Create System Diagrams | 7 | 10/20/19 | 10/27/19 |
| 4.2. | Create Wireframe Diagrams | 10 | 10/23/19 | 11/02/19 |
| 4.3. | Create Database Diagrams | 10 | 10/23/19 | 11/02/19 |
| 4.4. | Design Revisions | 3 | 11/02/19 | 11/05/19 |
| 4.5. | Final Design Team Approval | 3 | 11/02/19 | 11/05/19 |
| **V.** | **SETUP** | **76** | **09/24/19** | **12/10/19** |
| 5.1. | Setup AWS Elastic Beanstalk | 21 | 09/24/19 | 10/14/19 |
| | 5.1.1. Create Account | 10 | 09/24/19 | 10/03/19 |
| | 5.1.2. Configure | 10 | 10/03/19 | 10/14/19 |

| | | | | |
|---|---|---|---|---|
| 5.2. | Integrate Flask | 55 | 10/14/19 | 12/08/20 |
| 5.2.1. | Setup python Virtual Environment with Flask | 15 | 10/14/19 | 10/29/19 |
| 5.2.2. | Create Flask Application | 15 | 10/29/19 | 11/13/19 |
| 5.2.3. | Develop Site with EB CLI | 20 | 11/13/19 | 12/03/19 |
| 5.2.4. | Cleanup | 5 | 12/03/19 | 12/08/19 |
| **VI.** | **DOMAIN NAME** | **5** | **12/03/19** | **12/08/19** |
| **VII.** | **ENVIRONMENT CONFIGURATION** | **120** | **09/24/19** | **01/22/20** |
| 7.1. | Auto Scaling | 40 | 09/24/19 | 11/03/19 |
| 7.1.1. | Configuring Auto Scaling Group | 20 | 09/24/19 | 10/14/19 |
| 7.1.2. | Configure Triggers | 10 | 09/24/19 | 10/24/19 |
| 7.1.3. | Configure Health Check Status | 10 | 10/24/19 | 11/03/19 |
| 7.2. | Load Balancer | 40 | 11/03/19 | 12/13/19 |
| 7.2.1. | Configuring Load Balancer | 40 | 11/03/19 | 12/13/19 |
| 7.3. | Database | 40 | 12/13/19 | 01/22/20 |
| 7.3.1. | Adding DB Instance | 10 | 12/13/19 | 12/23/19 |

| | | | | |
|---|---|---|---|---|
| | 7.3.2. Connecting to DB | 15 | 12/23/19 | 01/07/20 |
| | 7.3.3. Configuring DB | 15 | 01/07/20 | 01/22/20 |
| 7.4. | Software Settings | 120 | 10/15/19 | 02/15/20 |
| | 7.4.1. Configuring Environment properties | 25 | 10/15/19 | 11/09/19 |
| | 7.4.2. Software Setting Namespaces | 25 | 11/09/19 | 12/04/19 |
| | 7.4.3. Accessing Environment Properties | 50 | 12/04/19 | 01/23/20 |
| | 7.4.4. Debugging | 25 | 12/04/19 | 12/29/19 |
| | 7.4.5. Log Viewing | 25 | 12/29/19 | 01/23/20 |
| 7.5. | Create Landing pages | 20 | 01/23/20 | 02/12/20 |
| | 7.5.1. Create Navigation Bars | 10 | 01/23/20 | 02/02/20 |
| | 7.5.2. Cross Browser Testing | 5 | 02/02/20 | 02/07/20 |
| | 7.5.3. Optimization | 5 | 02/07/20 | 02/12/20 |
| **VIII.** | **SECURITY** | **20** | **01/23/20** | **02/12/20** |
| 8.1. | Setup Permissions | 5 | 01/23/20 | 01/28/20 |
| 8.2. | Create User Authentication with SSO | 10 | 01/23/20 | 02/02/20 |

| | | | | |
|---|---|---|---|---|
| 8.3. | Create User Authentication with DUO | 10 | 02/02/20 | 02/12/20 |
| **IX.** | **TESTING** | **30** | **02/12/20** | **03/12/20** |
| 9.1. | Set up Mock Clients | 10 | 02/12/20 | 02/22/20 |
| | 9.1.1. Set up Desktop Clients | 10 | 02/12/20 | 02/22/20 |
| | 9.1.2. Set up Mobile Clients | 10 | 02/12/20 | 02/22/20 |
| 9.2. | User Interface Testing | 10 | 02/22/20 | 03/03/20 |
| 9.3. | Functionality Testing | 10 | 03/03/20 | 03/13/20 |

*Table 6:* Project Objectives/Deliverables Due Dates

***Figure 10:*** **Phishing Guard Gantt Chart**, seen below, depicts the Gantt Chart used for the Phishing Guard project.

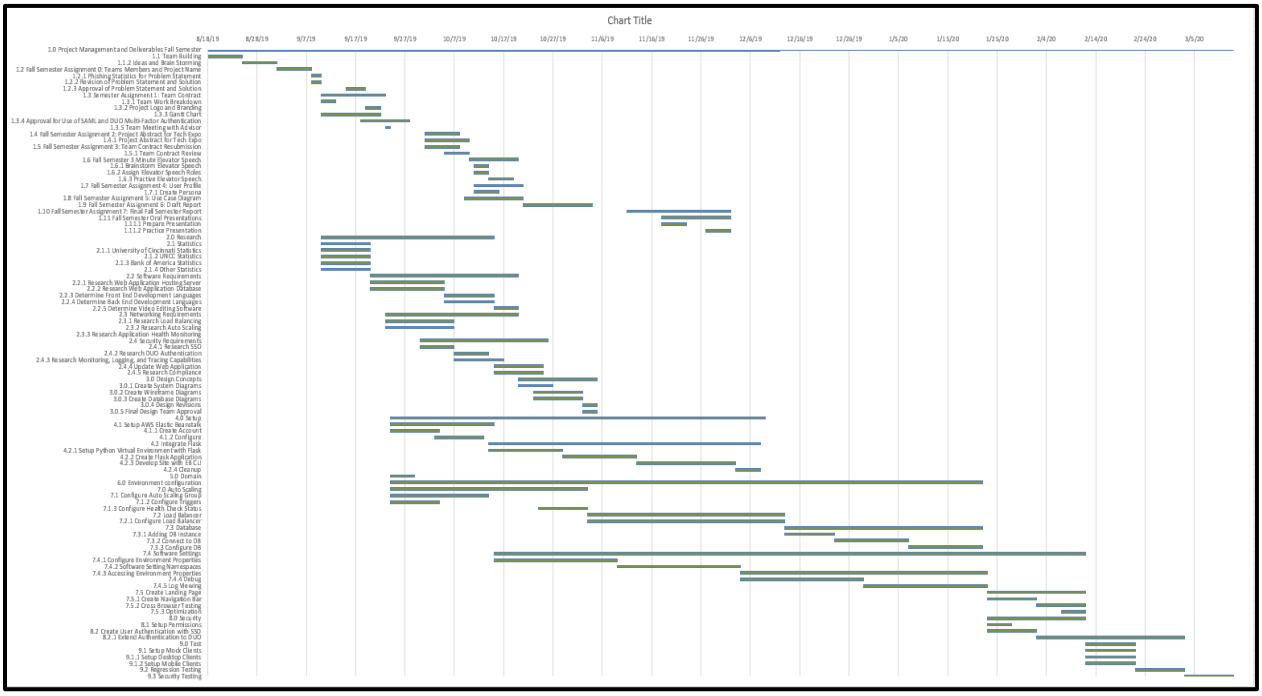



*Figure 10:* Phishing Guard Gantt Chart

## Problems Encountered

As a group we have encountered quite a few issues while working on this project.

First, we had communication issues. It was hard for us to meet consistently and keep each other updated on what we were working on. As a result, our meetings were not structured. To address this, we setup multiple slack channels to ensure that we kept each other updated on what we were each doing and where the project was headed. This solution enabled us to stay on track and meet each of the deadlines we set for ourselves across both Fall and Spring semester.

Second, we realized that the amount of paperwork that we had to complete for the project would be quite large, mainly due to numerous ideas that we wanted to implement. To address this, we once

again relied on Slack. We divided the paperwork equally and each of us knew what part they had to work on and deliver on prior to the submission of our assignment.

Finally, deciding which feature to implement and which one to ignore was by far one of, if not the most challenging thing we encountered. There was so much we wanted to do but, quickly realized that if we did not narrow down the scope of our project, we would either deliver an incomplete or inoperable product, or worse simply give up and not put in as much effort as we did. We addressed this by determining which features would be key to our application, which ones would be crucial as far as addressing the problem we are trying to solve was concerned, and which ones would be great to have only if we have the time to develop them (in other words, our stretch goals).

## Future Recommendations (improvements)

If we had to do it all over again, we would start working on the project itself earlier than we did so we would deliver an even better product. We would meet a lot more often to ensure each team member is on track and getting their part done. Furthermore, we would try to add some of the features we were not able to include this time around, such as integrating machine learning and developing a mobile application to support our web application. Were we to do this again, we would start by having a specific scope for our project. Though the goal would still be to help users familiarize themselves with what phishing is, what they can do to detect it and, how they can report it, we would try to think of different way that could be accomplished. We would even potentially look at cost-effective tools that users could use alongside our product to ensure they are increasing their knowledge, awareness and retention rate.

# CONCLUSION

## Lesson Learned

Above all else, working on this project taught us that deadlines approach much faster than one expects. Staying on top of things, consistently communicating with your team and setting objectives to achieve are the keys to success. We learned a lot about project management and developed our time management skills. We also learned a plethora of new technical skills, such as learning and becoming proficient with new coding languages. In a nutshell, working on this project has developed each of us as young professionals, and taught us valuable lessons that we will carry throughout our careers.

## Abilities/Skills enhanced

Our technical writing skills have greatly improved. In addition to enhancing our understanding and proficiency in the coding languages we already knew, such as Python, HTML or CSS, we also learned and became proficient with new coding languages and platforms as well, such as JavaScript and AWS.

## What have we accomplished since Fall 2019?

We have accomplished everything we had set out to do for Spring semester 2020, and then some. We used the prototype we had developed last semester and enhanced it in every way possible. We added new features to the product such as the interactive quizzes and games that were added after learning JavaScript. We fully and thoroughly tested our application. We prepared ourselves for our final

presentation and IT Expo ahead of time and, ensured that we had everything we needed done early. Simply put, all our hard work enabled us to fully develop and test our application before our deadline, as we had intended. We eliminated unnecessary headaches or turmoil by being proactive, rather than reactive.

## What did we learn from IT Expo?

IT Expo taught us a lot about knowing how to present your product into the market. In other words, it taught us about the importance of knowing how to present ourselves as professionals and how to professionally deliver and introduce a product into the market. We believe that this experience has set us up for success in the future, especially when we get assigned a big project in the companies we work for that we will need to present to upper management or potential customers.

## REFERENCES


[1] Check Point Research 2018 Security Report Summary. *Phishing Simulation & Awareness Training, 2018.* Retrieved August 31, 2019, from www.phishingbox.com/news/phishing-news/check-point-research-2018-security-report-summary.

[2] PhishLabs. *2019 Phishing Trends and Intelligence Report.* Retrieved August 31, 2019, from https://info.phishlabs.com/hubfs/2019%20PTI%20Report/2019%20Phishing%20Trends%20and%20Intelligence%20Report.pdf.

[3] ESET. *5 Reasons Phishing is so Successful.* Retrieved September 18, 2019, from https://www.eset.com/uk/about/newsroom/blog/5-reasons-phishing-is-so-successful/.

[4] IT Governance. *Phishing attacks: 6 reasons why we keep taking the bait.* Retrieved September 18, 2019, from https://www.itgovernance.co.uk/blog/6-reasons-phishing-is-so-popular-and-so-successful.

## APPENDIX A: Phishing Guard Poster

University of CINCINNATI

# Phishing Guard

College of Education, Criminal Justice, and Human Services
School of Information Technology

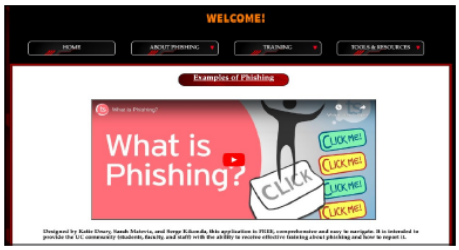
Katie Drury | Sarah Matevia | Serge Kikonda | Advisor: Tony Iacobelli | Team 26

### What's the Problem?

- Enjoy Safer Technology (ESET) claims that 97% of people are still unable to identify a more sophisticated phishing attack
- Phishing has remained a never-ending and constantly evolving threat faced by all
- Most users are not properly trained to recognize phishing attempts

### What's our Solution?

- FREE, comprehensive, and easy to navigate web application (Phishing Guard)
- Tailored towards the UC Community (Students, Staff, Faculty)
- Provides the ability to receive effective training about phishing and how to report it.




### Key Benefits

- Hosted in AWS
- URL Analysis
- Familiar with Python Back-end
- Unique reporting capability
- Multi-Factor Authentication
- Repository for up to date information

### Technology

Amazon Web Services

Python

Duo Multi-Factor Authentication

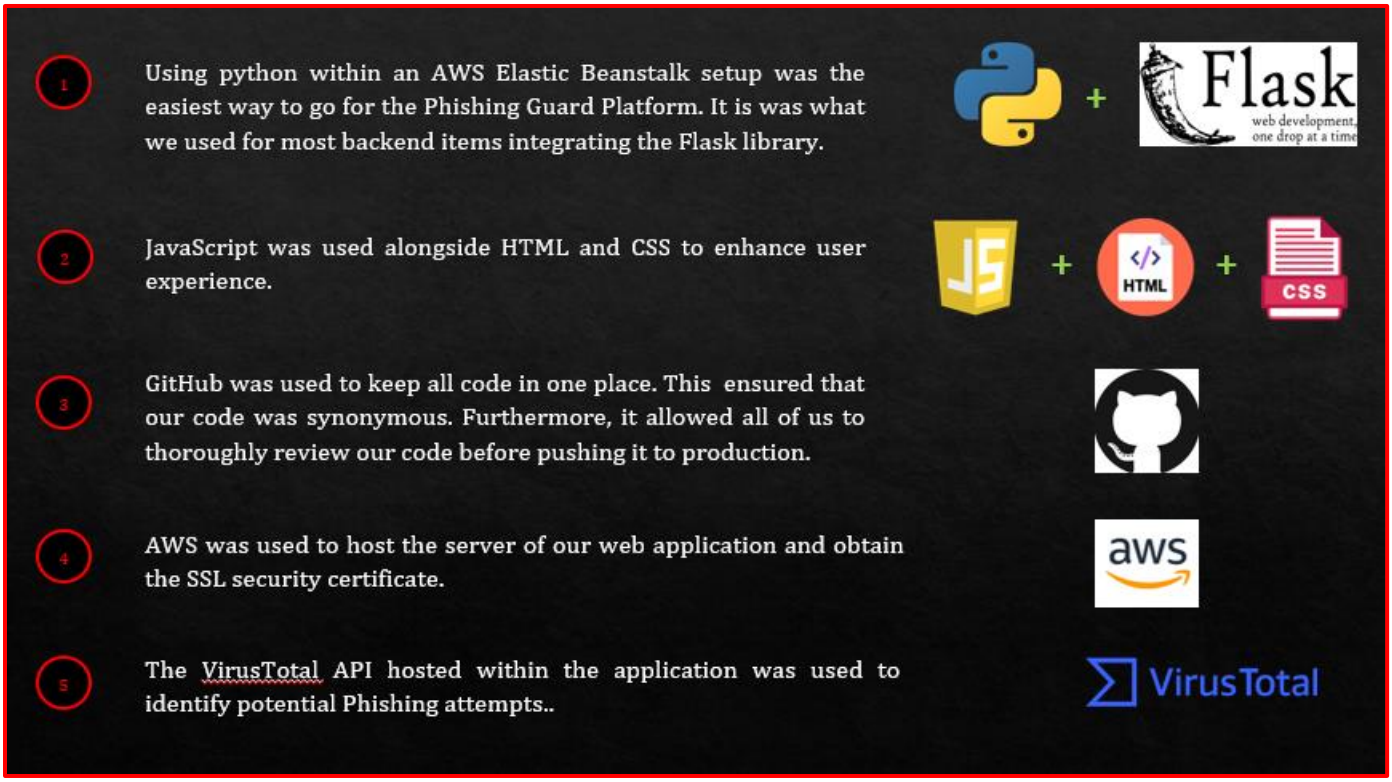
GitHub

AWS Elastic Beanstalk

AWS Cognito

VirusTotal

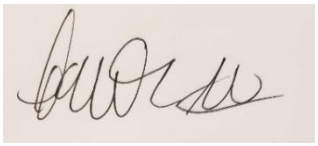
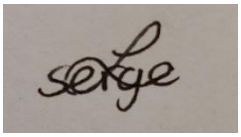
## APPENDIX B: Technical Diagram



## APPENDIX C: Team Signature

**University of Cincinnati, CECH College (School of Information Technology)**

**Phishing Guard- Official Team signatures**

| Team Member | Date Signed | Signature |
|---|---|---|
| **Katie Drury**<br>Project Manager and Security Analyst | **4/27/20** |  |
| **Sarah Matevia**<br>Software Developer and Security Analyst | **4/27/20** |  |
| **Serge Kikonda**<br>Network Engineer and Front-End Development | **4/27/20** |  |
| **Toni Iacobelli**<br>Advisor | **4/27/20** | **Toni Iacobelli** |