Final Report

Homeland Security Netspider

Paulo Drefahl, Zachary Sutton, Alyssa Chiego, Kevin Kostage, Gregory Bateham, Dylan Garcia

Table of Contents

Introduction:

This document is a result of all the milestone documents (Milestone 1, Milestone 2, and Milestone 3) submitted over the course of this semester as well as the Software Requirements and Specifications (SRS) given to the sponsor and the User guide provided to the investigators. This document involves references not only from this academic semester (Spring 2024) but it also from the previous semesters (Fall 2023)

Description:

Homeland Security Netspider is a software tool developed in partnership with the Department of Homeland Security. It aims to assist law enforcement in the battle against human trafficking. Leveraging a comprehensive array of technologies, including Python, Selenium, HTML, CSS, JavaScript, and Electron, this tool is meticulously designed to equip investigators with a potent means of combating crime. The essence of this project lies in its ability to streamline the process of gathering crucial evidence from diverse online platforms. By furnishing investigators with an intuitive GUI, Netspider facilitates seamless addition, editing, and management of keywords and keysets, thereby optimizing the extraction of pertinent data and displaying it in a clean and organized interface. Through the implementation of enhanced web scraping techniques, Netspider ensures precision information retrieval while minimizing false positives. The culmination of these efforts translates into accelerated investigations marked by heightened accuracy, ultimately enhancing the efficacy of law enforcement in dismantling human trafficking networks.

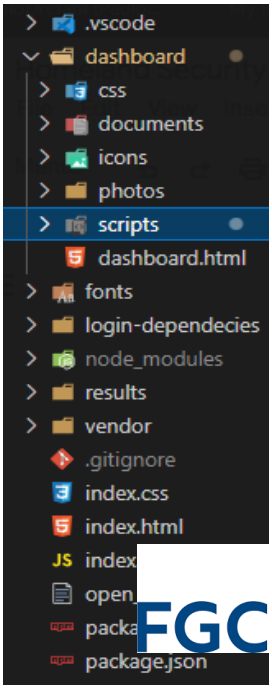
1. User stories or requirements

User Story	Requirement
The Front-End System should utilize Electron.Js, Node.js, and Javascript	**1. The system must be built using Electron.js as the framework for creating the desktop application.** **2. Node.js should be used for handling backend operations within the Electron.js application.** **3. JavaScript (ES6 or later) must be used as the primary programming language for both front-end and back-end development.**

The Front-End System shall be a single-page application.	**The Front-End System must be developed as a single-page application (SPA) to ensure seamless user interaction without reloading the page.**
The Back-End System should utilize Flask to connect to the Front-End System	**The Back-End System must be developed using the Flask framework to handle HTTP requests and responses for communication with the Front-End System.**
The Back-End System shall utilize multiple threads to perform multitasking and maintain a constant connection with the Front-End System.	**The Back-End System must implement multi-threading capabilities to perform multitasking operations efficiently and maintain a constant connection with the Front-End System for real-time data exchange.**
The User shall be able to manage the results and open files from each result from the menu.	**The Front-End System must offer functionality for the user to manage and open files resulting from web scraping operations, ensuring these actions are accessible via the menu.**
The Menu should be able to guide the user through the process of initiating the web scraping.	**The Front-End System must offer functionality for the user to manage and open files resulting from web scraping operations, ensuring these actions are accessible via the menu.**

2.0 Software architecture and design

The software architecture and design of the project were structured to seamlessly integrate front-end and back-end components, leveraging a combination of technologies to ensure functionality and user accessibility. The front end was developed using HTML, CSS, JavaScript, and Bootstrap. Culminating in a single-page application (SPA), it was compiled as a desktop application through the Electron.js library. This approach will be able to provide users with a familiar and intuitive interface, enhancing usability and enabling cross-platform compatibility. The front end consists of a login page, serving as the gateway to the SPA, which has the core features, including the scraper and its settings, keyword editing functionality,



support access, file management capabilities, and result visualization tools.

On the backend, Python served as the primary programming language, facilitating the implementation of essential functionalities such as web scraping and data processing. Selenium, Chrome Driver, Numpy, and SciPy were utilized to automate web browser interactions, enabling efficient data retrieval from targeted websites. Additionally, a Flask API was incorporated to establish communication between the backend and the frontend, this process is also facilitated through websockets and JSON files, allowing for seamless data exchange and enhancing the application's overall responsiveness and performance. This modular architecture enabled the front-end and back-end components to operate cohesively, ensuring efficient data flow and enabling users to interact with the application seamlessly. Overall, the software architecture and design of the project prioritized usability, performance, and integration, offering users a robust and intuitive platform to combat human trafficking and address illicit activities effectively.

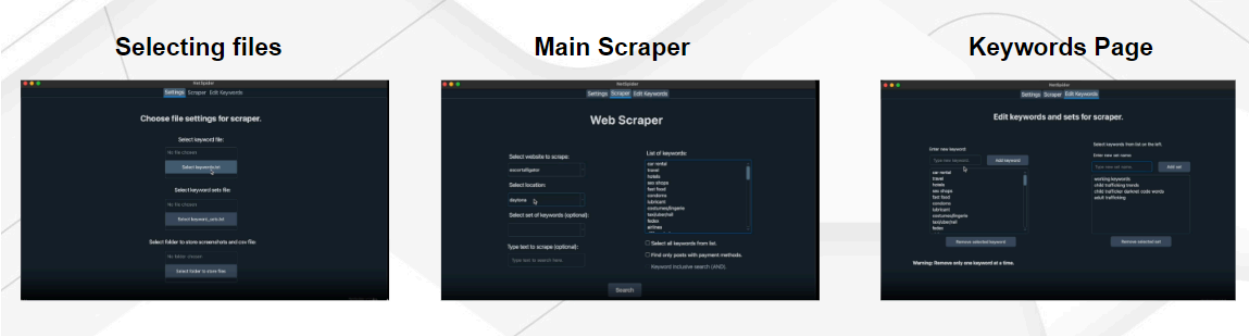
2.1 Evolution of Netspider

The evolution of Netspider during the past two semesters is represented by a significant transition from PyQt6 to Electron.js for desktop application development. This move of frameworks enhanced scalability, performance, and cross-platform compatibility. Additionally, we refined our use of Selenium for web scraping during our time with the project, significantly improving speed, accuracy, and reliability. We've also redesigned the user interface and feature set to prioritize usability and data presentation, empowering investigators with intuitive tools for informed decision-making and efficient data analysis. This evolution reflected our commitment to modernization, innovation, and user-centric design, positioning Netspider as a powerful tool in the fight against human trafficking and illicit activities.

2.2: Version 1v

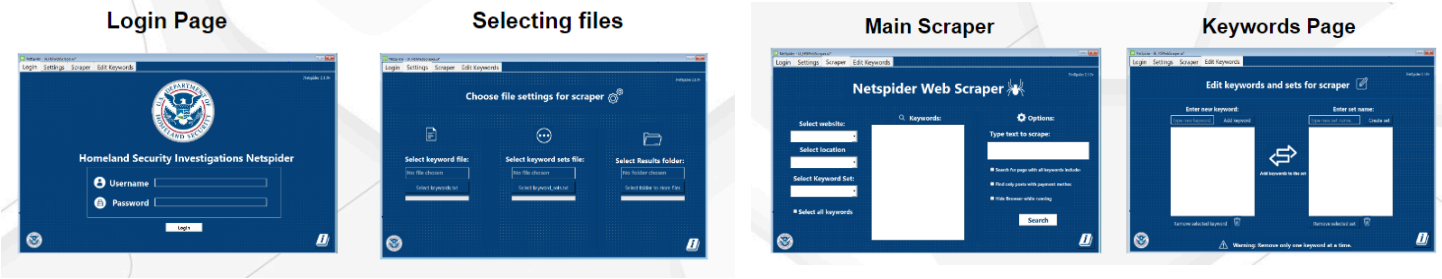
In version 1.0.0, this was the first version of NetSpider, laying the foundation for the program. This version of the program was worked on by a group of software engineering seniors before ours. The previous collaborators were Diego Grisales, Oscar Fox, Ragy Costa De Jesus. Through the use of PyQt, there were three main tabs made for the program: the selecting files tab, the main scraper, and the keywords page tab. The selecting files tab, allowed the users to select the keywords text and keywords set text files, through user-friendly interfaces for text file input. The keywords page allowed the user to add or delete the keywords or keysets. The main scraper is where the main settings are chosen, so the location and specific keywords will be specifically searched

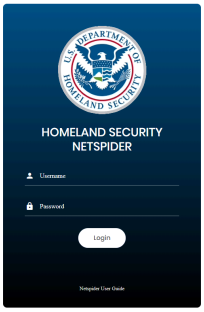
for in the main scraping process. The main scraper tab works as the control center for the program. With PyQt's flexibility, the program's interface was designed with both usability and accessibility in mind, ensuring a seamless experience for users of all backgrounds.



2.3: Version 2v

In version 2.5.5, we renovated the program by enhancing both its aesthetics and functionality. This involved implementing vibrant color palettes, integrating captivating images, and refining the overall visual appeal. Additionally, we enhanced its functionality to ensure a more seamless user experience. These updates not only improved the program's appearance but also elevated its usability, making it more engaging and intuitive for users. Some of the functionality included adding a login page which helped users authenticate themselves before accessing the program, which prevented unauthorized access and protected sensitive data. The main scraper was given more functionality, such as the ability for the program to run “headless”, which means that it will run in the background, not allowing the users to see what the program is looking for. There was also a timer added, showing how long the scraper has been running. This is the second version of the program that was delivered to Homeland Security Investigations.





2.4: Version 3v

In version 3.0.1v (the one referred in this sprint) of *Homeland Security Netspider*, we added significant enhancements aimed at improving user experience and functionality. A major highlight is the introduction of a completely revamped user interface (UI) featuring a new, intuitive system for adding keywords, an interactive platform for adjusting scraper settings, and a results manager. These updates empower users with a more seamless and efficient workflow, facilitating easier management of investigative tasks. Additionally, we implemented a novel method for incorporating extra files, enhancing the tool's versatility, and adaptability to diverse data sources. Furthermore, we recognized the importance of providing comprehensive support to stakeholders, we integrated a dedicated support section within the application, complete with references and resources to aid users in maximizing the utility of Homeland Security Netspider. These enhancements reflected our commitment to innovation and responsiveness to user feedback, ensuring that Homeland Security Netspider remains a cutting-edge solution in the fight against human trafficking and illicit activities.

2.5: Consideration of Factors in the Development

In developing Homeland Security Netspider, our primary focus was on combating human trafficking while ensuring alignment with public health, safety, and welfare, as well as global, cultural, social, environmental, and economic considerations. We prioritized ethical and legal standards to safeguard against this egregious crime, implementing stringent data protection measures to protect victims' privacy and rights. Features promoting inclusivity and accessibility were integrated to address the diverse cultural and social contexts in which human trafficking occurs. Additionally, by streamlining investigations and law enforcement operations, Netspider contributes to reducing the social and economic costs associated with human trafficking. Through these efforts, Netspider stands as a powerful tool in the fight against human trafficking, promoting ethical, sustainable, and socially responsible practices to protect the most vulnerable in our society.

3. Code

The code file for Homeland Security Netspider is extensive due to the comprehensive array of technologies and functionalities incorporated into its development. Given its complexity and size, it is not feasible to include the entirety of the code in this submission. However, to provide interested parties with access to the source code and facilitate further exploration, we have prepared a GitHub repository where the code can be accessed in its entirety. The GitHub link is provided below for reference **https://github.com/PauloDrefahl/NetspiderHSI**. This repository contains all the necessary files, documentation, and resources related to the project, enabling users to delve into the codebase, review implementation details, and contribute to its ongoing development. We encourage interested individuals to visit the GitHub repository to access the code and explore the intricacies of Homeland Security Netspider.

3.1 Branches and Development Process

In our development process, we adhere to the principles of Agile methodology, characterized by iterative development, frequent collaboration, and adaptive planning. Each branch in our repository reflects a specific aspect of the project, with distinct focuses and contributions from team members.

The "BACKEND-2.5.2v" branch pertains to the backend development, focusing on refining functionalities and addressing issues in the current version. Meanwhile, the "GUI-1.8.3v" branch indicates the initial stages of GUI development, where basic features and layouts are established.

Branches
BACKEND-2.5.2v
BACKEND-2.5.2v-debugger-frontend
GUI-1.8.3v
GUI-3.0.1v
GUI-3.0.1v-Merge
GUI-Alyssa-3.0.1v

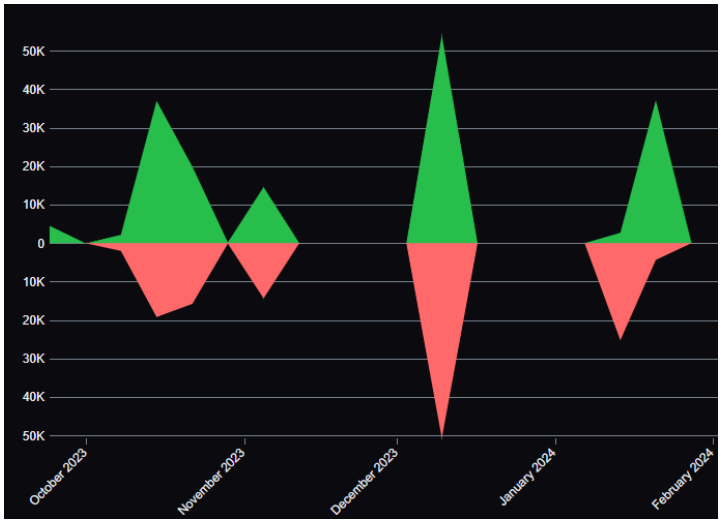
GUI-Dylan-3.0.1v
GUI-Greg-3.0.1v
GUI-Paulo-3.0.1v

As development progresses, new branches are created to accommodate feature enhancements and bug fixes. For instance, branches like "GUI-3.0.1v" and "GUI-3.0.1v-Merge" signify updates and merges related to the GUI interface, incorporating new functionalities and improvements.

Additionally, branches like "GUI-Alyssa-3.0.1v", "GUI-Dylan-3.0.1v", "GUI-Greg-3.0.1v", and "GUI-Paulo-3.0.1v" denote parallel development efforts by individual team members, each focusing on specific aspects or features of the GUI. This approach enables efficient collaboration and specialization, ensuring that each team member can contribute effectively to the project.

Throughout the development process, we emphasize multiple meetings per week and dedicate more than 20 hours of work per week per group member. These frequent interactions and extensive time commitments facilitate ongoing communication, feedback, and progress tracking, enabling us to respond swiftly to changes and deliver high-quality results in line with Agile principles.

3.2 Code Additions and Deletions (Monthly)



Obs: The metrics refer to the number of lines added in each specific month for each sprint (main branch).

3.3 Latest Version Demo

The latest version demo of our project is accessible via this link. This demo showcases significant advancements aimed at combating human trafficking and supporting law enforcement efforts.

3.4 User Guide

In order to guide new investigators through Netspider, we created and linked within our application a user guide document that can be accessed in the “help” section. The user guide contains an overview of the software as well as a step by step guide of how to customize, manage, and ultimately use and get the best out of what Netspider has to offer.

4. Key Features

This paragraph is dedicated to pointing out and explaining the main components of Netspider as well as its usage and implementation, this includes only items in the newest stable version available and given to the Homeland Security Investigation Officials:

4.1 Login Page: Allows for users to use a designated key (Username and Password) to login to Netspider.

4.2 File Configuration: Allows users to select specific files they would like to use in the data-scraping process (selecting the keyword file they would like to choose keywords from, selecting the result folder in which data will be stored, etc.)

4.3 Main Scraper: A data-scraping program that collects information from websites using specific settings within the program. The main feature of the application.

4.4 Website Selection: Allows a user to select a specific website they would like to collect data and information from.

4.5 Keysets Selection: Allows for a user to select a specific keyset (or set of keywords) that will be used within the scraping process.

4.6 Location Selection: Allows for a user to select a specific location (within the Florida area) they would like to collect data and information from. Situational based on the website selected.

4.7 Keywords List: Displays a list for the user to select specific keywords or all keywords to be scraped and collected from the specified website.

4.8 Extra Settings: Allows the user to scrape for specific text, only payment methods, only if every keyword is found, and headless.

4.9 Headless mode: Allows the user to have the scraper run in the background and not appear on screen during its runtime.

4.10 Scraper Control Panel: This is where the user will start and stop the scraper and can see if the scraper is running with a status and how long it has been running.

4.11 Keywords and Keysets Manager: Allows the user to add keywords, remove keywords, create keysets from keywords, and delete keysets**.**

4.12 Help section: Contains information regarding our GitHub repository, as well as a general quick guide and documentation about the application.

4.13 Results Manager Section: The output results from the scraper in the result folder are listed where the user can select to perform operations on selected results.

4.14 Visualizing Diagrams: Selecting a result from the manager will allow the user to view diagrams of statistics relating to geographical findings of the results and correlations of certain attributes.

4.15 Visualizing Raw and Clean Data: The web scraper has functions that allow to store of the raw data and preprocessing of the data into a clean format to allow further manual review and automated analysis while keeping the raw format.

4.16 Screenshots: Pictures taken from the live data-scraping process of websites that were flagged as including specific keywords or phrases. These are stored within the results folder.

4.17 Generating Data Report: The data report is a collection of evidence gathered from every post that is flagged during the scraping process to have matching keywords inputted into the web scraper.

4.1 Test cases and test reports

Test Case ID	Test Cases	Steps	Expected Output	Actual Output	Status
1	Testing Connection between the front and back end port with websockets	1. Run Flask Server 2. Run Front-end debugger client 3. Check the text for Connection Status	Status states connected	Status states connected	Pass
2	Implementation of the new website	1. Run Flask Server 2. Run Front-end debugger client 3. Check the text for Connection Status 4. Initiate Web scraping process 5. Check to see if the website is properly being searched through 6. Check to see if the results are being collected properly	Results were collected properly and the website was searched	Results were collected properly and the website was searched	Pass
3	Testing the starting and ending of the processes and sub-processes	1. Run Flask Server 2. Run Front-end debugger client 3. Check the text for Connection Status 4. Initiated the scraping process and waited to finish 5. Checked the amount sub-processes at start and end	The amount were the same and there were no errors	The amount were the same and there were no errors	Pass

		6. Initiated the web scraping process again 7. Checked the sub-processes to ensure the amount was the same and for any errors of forced closes			
4	Testing the opening and closing threads	4. Run Flask Server 5. Run Front-end debugger client 6. Check the text for Connection Status 7. Initiated the scraping process and waited to finish 8. Check the thread count at start and end 9. Initiated the process again 10. Check the threads to ensure the amount is the same	Threads open and close without errors, initializing and releasing resources properly.	Threads open and close without errors, initializing and releasing resources properly.	Pass
5	Flask Executable	11. Created the executable of the flask server 12. Run Front-end debugger client 13. Check the text for Connection Status	Flask executable launches the application without errors, enabling smooth communication between back-end and front-end.	Flask executable launches the application without errors, enabling smooth communicatio n between back-end and front-end.	Pass

5. Backlog

The backlog for our project is linked to the GitHub repository. This repository serves as a centralized hub where we manage and track the prioritized list of tasks, enhancements, and features planned for implementation throughout the development process. By maintaining a transparent and accessible backlog, we enable efficient collaboration, communication, and prioritization among team members. This ensures that everyone is aligned on project goals and can contribute effectively to its successful completion.

6. Any major project decisions (e.g., de-scoping and change in plans).

The decision to change the front-end stack from PyQT6 to Electron.js, node.js, and javascript allowed the team to split into two sub-teams: the front-end team and the back-end team. The menu had to be remade from the ground up due to the change. This change was brought about as Paulo wanted us to further our skills and use a framework that would be more relevant when we graduated. PyQT6 was a very outdated old framework that was drag and drop and left massive limitations for what we could do with the GUI. Switching over to Electron provides future groups that might work on this project with much greater possibilities for expansions without the limitations of PyQT6. We had wanted to pursue possibly working with chat websites like telegram or similar services but such would require too much clearance to do so.

Something like this would require making an account with the service and then getting invited to groups where illegal activities were occurring. This would then require us to scrape these chats and find what we were looking for, but as stated our sponsor told us that was out of our scope and not wanted for this current project. Our most recent addition was to add a task manager, designed to sort through the files generated by Netspider's finds. The result manager not only gives the user a choice of multiple functionalities including options such as viewing clean files or raw data, but allows the user to see all the results compiled in one area for readability and efficiency.

7. Team Member Contributions:

Name	Contribution	Number of commits	Team

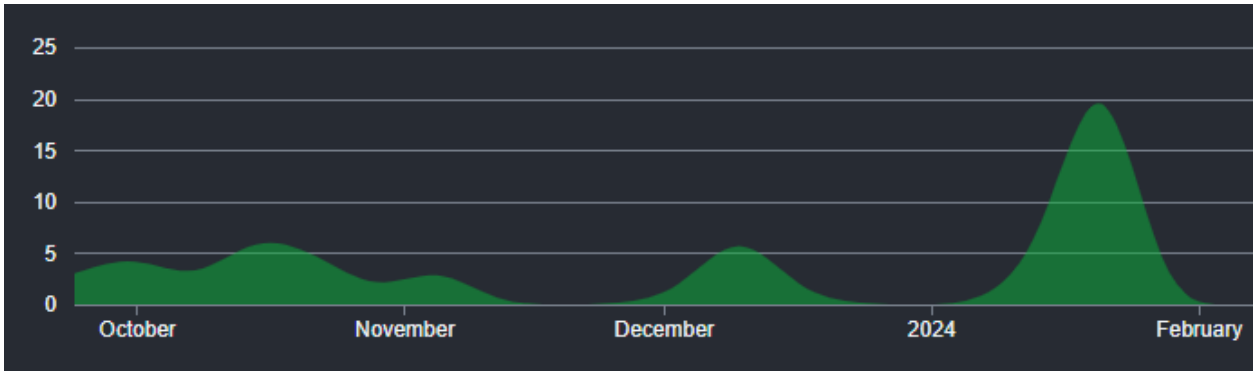
Greg	Worked on Front-End Development, Electron UI cosmetic work, and connectivity. Worked on list boxes and overall design, including the results manager.	**~10**	**Front-End**
Paulo	Project Manager, Worked on Back-End and Front-End Development, Organized meetings, and communicated actively with sponsor	**~50+**	**Front and Back-End**
Alyssa	Worked on Front-End Development, Electron UI cosmetic work, and connectivity. Implemented new fields and edited keywords and set boxes, including the results manager.	**~10**	**Front-End**
Kevin	Worked on Back-End Development, connecting Flask to Front-End, assisted in debugging executable, updating the format of the results, creating result manager, etc.	**~37**	**Back-End**
Zach	Worked on Back-End and Front-End Development, fixed Front-End bugs, connected Flask to Front-End, debugged executable, created thread manager, implemented web sockets, and created Front-End debugger.	**~50+**	**Front and Back-End**
Corey	Worked on Back-End Development, connecting Flask to Front-End, helped out with ports and sockets, etc.	**~10**	**Back-End**
Dylan	Worked on Front-End Development, worked on some Java script functionalities as well as Scraper Settings box.	**~10**	**Front-End**

7.1 Commit Table:

Developer	Commits
Greg	https://github.com/PauloDrefahl/NetspiderHSI/commits/GUI-Greg-3.0.1v?author=gdxbs
Paulo	https://github.com/PauloDrefahl/NetspiderHSI/commits?author=PauloDrefahl
Alyssa	https://github.com/PauloDrefahl/NetspiderHSI/commits/GUI-Alyssa-3.0.1v?author=alyssachiego

Corey	https://github.com/PauloDrefahl/NetspiderHSI/commits/main?author=CoreyRecord
Zach	https://github.com/PauloDrefahl/NetspiderHSI/commits?author=zesutton2619
Kevin	https://github.com/PauloDrefahl/NetspiderHSI/commits/main?author=Keko787
Dylan	https://github.com/PauloDrefahl/NetspiderHSI/commits/GUI-Dylan-3.0.1v?author=DylanG05

7.2 Commits Frequency (specifically to the Main Branch)



Obs: Commit frequency does not show commits to their personal branches or local branches. Only to the main origin/main branch

8. Ethical and professional responsibilities and informed judgments, which considered the impact of engineering solutions in global, economic, environmental, and societal contexts.

In light of the ethical nature of our products, we must carefully weigh the responsibilities we bear. Given that our web scraper navigates through inappropriate websites unsuitable for public viewing, one aspect we must address is the location of our project work. Testing cannot occur in venues like libraries or other public settings where our findings might become public. The potential global and economic ramifications of this project are significant. A primary objective is to identify potential victims of human trafficking, considering the estimated $150 billion in illicit revenue generated. It's crucial to acknowledge this objective as human trafficking extends far and wide, impacting numerous individuals.

9. Project feedback and performance review:

9.1: Sponsor feedback:

Our sponsor, Clinton Thompson, has been extremely pleased with our progress throughout the semester. We met with him at the office five times and he thoroughly enjoyed our advancements. We were able to deliver the basic needs of the project in the first week and exceeded expectations with additional features and improvements over the following months. Clinton was very impressed with our work and availability. We also presented our progress at the Intercept meeting with Homeland Security officials, receiving exceptionally positive feedback, which translated into additional requirements that we successfully delivered. We are grateful for Clinton's support and guidance throughout the project.

9.2 Mentors Feedback:

We also met with our mentors, Dr. Gonzalez and Prof. Osheroff, who provided us with valuable feedback and approval of our changes. They were impressed with our progress and the positive impact our software was making. Their feedback helped us refine our approach and deliver a final version that met their expectations. We appreciate their guidance and support throughout the project.

10. Summary Netspider 3.0.1v

Netspider has progressed a lot through this sprint as we have made many advancements. The biggest was the switch to Electron.js as the advancement in technology was a very large leap. The previous framework was PyQT6 which was drag and drop and caused many issues with linking and limitations of what we could do with the framework itself. Zach Sutton also presented at an Intercept meeting for Homeland Security where he presented our project to many high-level government officials. Giving a demonstration of what the project is capable of and what it can do. Another notable change throughout the semester was that more websites were added. As well as improvements to scraping and the design of logging the data from our scraping to Excel.

11. Company Impressions

Homeland Security Investigations (HSI) expressed satisfaction with our teamwork, highlighting our collaboration and coordination. HSI appreciated our proactive approach, adaptability to changing situations, and the ability to leverage each team member's strengths to overcome challenges. They expressed gratitude for our

commitment to excellence and highlighted our valuable contributions to their operations. Our proactive mindset involves thorough planning, risk assessment, and the ability to find emerging issues before they happen. The ability to adapt to challenges that would show up unexpectedly. Our partnership with HSI exemplifies the power of teamwork and underscores the importance of proactive planning, adaptability, and leveraging individual strengths.

12. Team Reflection

As a team of seven, we originally went into this project thinking that things would be complicated due to the amount of developers within our team. As time progressed, we learned to work with each other in more ways than we could imagine, eventually leading to the almost-perfect integration of a front-end and back-end team that would tackle problems individually and then combine during the end of each sprint. The hardest part as a team was coordinating different meeting times, as everyone being an active student made consecutive operations difficult to maintain. We found solutions in collaborating with schoolwork and our studies for other classes while we also tackled the mountain that was creating this project from start to finish.

Our team was split in 2 (front-end and back-end) with Paulo Drefahl being a mediator and contributor to both aspects of the project, allowing for a balanced workspace where both the design and functionality of the application were constantly being worked on. The leading factor that helped this team create Netspider V3 was the unrivaled dedication to our communication and collaboration as we tackled different obstacles within our code on a weekly basis.

For future reference, a team of seven developers is not ideal and would not be preferred for a senior project, but we ended up making it work in our favor. We do not encourage that a team this large should be formulated and in fact encourage that a smaller team should be used if the project is to be continued. That way, communication and development can be on a smaller and controllable scale.

Paulo Drefahl

Project Manager