**GONZAGA UNIVERSITY**

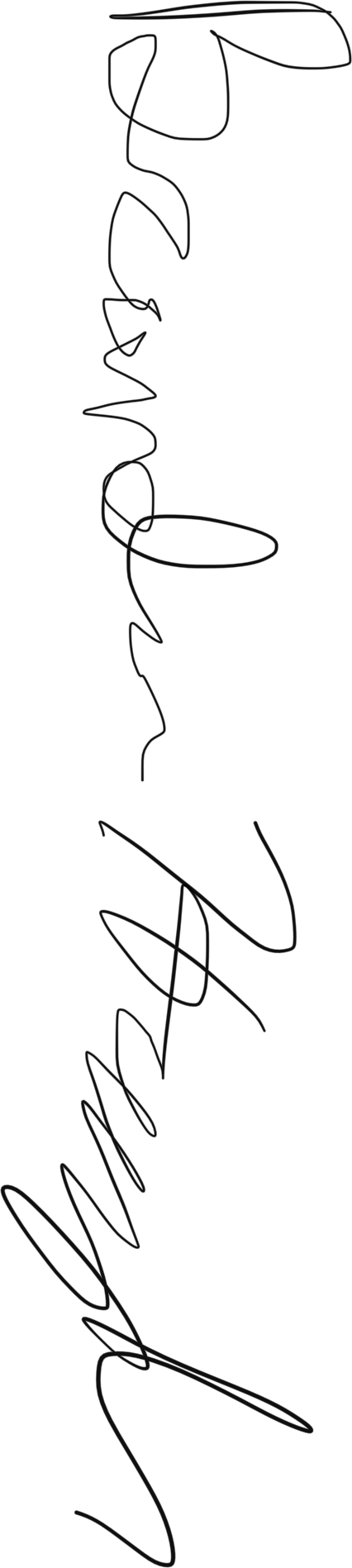
**School of Engineering and Applied Science**

**Center for Engineering Design and Entrepreneurship**

**FINAL PROJECT REPORT**

**May 3, 2024**

**<Device and Software Inventory Tool>**

**Prepared by:** 



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Brandon Huyck Colleen Lemak



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Artis Nateephaisan John Nealon

**Reviewed by:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Mike Mudge Rachel Kunkel

Faculty Project Advisor Project Sponsor/Liaison

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Richard Weeks Amanda Hepper

Design Advisory Board Member Project Sponsor/Liaison

**1 Project Overview**

In our modern age, companies are valuable targets for digital security breaches and malicious attacks. This is especially pertinent for companies that need to adhere to medical record compliance standards to protect sensitive patient health information. The mission of our sponsor, Medcurity, is to ensure that companies handle such data with care and mitigate its risk of being exploited by attackers.

Our tool assists in that mission by scanning the networks of Medcurity’s clients and automating the compilation of an inventory of software and devices connected to it. The tool will crawl the client’s network and add the information discovered to our database. This information is then displayed to the client through a user interface (UI) that lets them update and insert incorrectly gathered and missing information. After review, they can export the data to a spreadsheet for ease of access.

Our project features three main deliverables: the software tool outlined above, accompanying documentation, and all reports and presentations produced during the course of development. For each feature of the software, there is a file associated with it that outlines the intricacies of how it was developed, how it should be used, how it can be maintained, and how it integrates with the other features. There is also documentation that provides a project-wide overview.

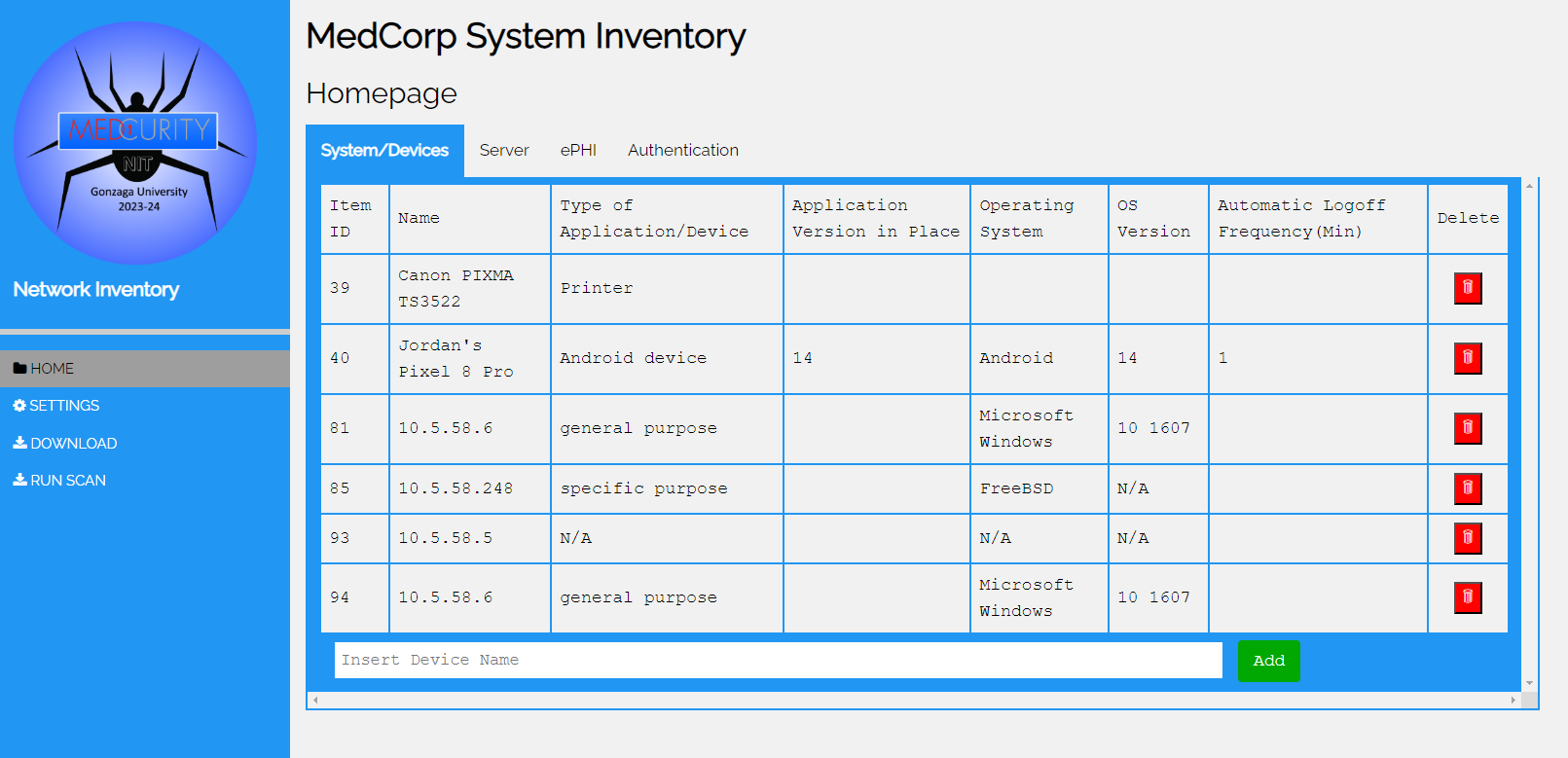
Our sponsor provided us with a set of broad requirements that allowed the team a lot of agency in deciding what and how to implement. As such, the main requirements of the system were identified to be as follows:

* a database management system that is secure, easily queryable, organized into tables for efficient inventory retrieval, developed based on a list of attributes requested by the sponsor, and created alongside an API for the crawler and export Python scripts to easily plug in to;
* a crawler agent that will traverse the network it is connected to, searching for devices and software that are also connected to the network, and add them to the database through its API
* an export CSV script that allows for the client users ease of portability that can be launched through the UI and accesses the data in database through the API
* an intuitive and visually appealing user interface for the means to view, analyze and control the program’s behavior; the ability for the user to interact with the UI, logging in with their credentials, add, update, and remove client device information, choose when to export the data into a csv file, and more
* two types of user accounts–client and admin–to allow for different view-scopes of the data when accessing the tool for client users and Medcurity admin users
* project documentation that is clear and concise on all parts and phases of the project and its lifecycle, from planning and development to deployment and maintenance, that encompasses a comprehensive report of all aspects of the project ensuring transparency
* user acceptance testing feedback, gained from events that allowed us to gather insights and areas for improvement from mock-end-users.

**2 Work Accomplished**

Our team has completed the release plan and modified the network tool after valuable feedback from our Sponsor. A large part of our project purpose is to scan the network and log all targeted information to a secure remote database.

The crawler agent aims to crawl applicable spreadsheet attributes in the Sponsor-provided example of system inventory. Currently, assuming the configuration of the network is online, information will be returned for the server default gateway IP, the location of the device running the script, the network subnet, and server encryption if applicable. Additionally, the Python functions are capable of returning each device IP, the number of up devices on the network, each MAC address, device type (general purpose, phone, etc.), and operating system name and version. Multithreading by creating custom classes to improve performance was another ideal goal; however, the script implements adding threads via an nmap system call because it provides output messages. Port scanning may consume lengthy amounts of time, and there is a time-out limit set for each host, 190 seconds. Our crawling-script returns port IDs, their statuses (open, filtered, etc), and the service running on the port and its version if applicable.

Medcurity connected us with an SQL database, and our team has been able to implement relations, tables, primary keys, foreign keys, data validation, and ensure 3NF execution and structure. Crawled attributes from the Python scripts are logged in the database and checked prior to insertion that the item is not a duplicate. We push the information to a remote Python integrated API connection and are able to view the changes in the database. Another project component is the user-interface which is coded in HTML, CSS, and PHP. With PHP, we are able to have HTML and database connections, which is crucial to query the database. There exists an initial login page with a login screen and options for administrator login. The user-interface offers navigation for users and allows attributes to be dynamically pulled from the database to display. The interface affords scanning the network from the UI and exporting the data associated with the logged in client to a CSV file. This file is generated by a Python script which is executed upon button-click of the UI; the data is exported in CSV format for our Sponsor’s ease of use and ability to upload into Excel.

Our team documents all tasks on our Github project backlog and we use pull-requests and issues to address progress and plan for sprints. Documentation for administrators and users detailing how to use the tool are provided as part of the Github project delivery. Every sprint (2-weeks) we conduct code-reviews to document our modifications and additions to the project–all available on the SD-2023-CS10 project. Finally, our software testing involves a test-network with a RaspberryPi, and unit tests. We have performed thorough integration with testing components and have been documenting decisions through the repository.

**3 Work Remaining**

To improve the user experience, we have discussed numerous future development ideas. With unexpected development challenges during the project, improvements that are not possible within our timeline will be passed to our Sponsor for consideration. First, we aim to improve the client experience by allowing each client to have a manager for user-account management. This way, clients do not have to contact Medcurity to manage a user’s account. We also find it important that admins are able to set their client to view, instead of being presented with all information available to them. Additionally, the user-interface would benefit from editable cells on the user-interface's Server page and user-interface filtering. Currently, users are unable to edit cells in the Server tab, so extending this functionality would allow users to modify necessary information. Adding a configuration network page and settings-accessibility page to the user-interface is another future development idea as users could view their network’s configuration, customize default settings, and receive additional context regarding their network.

In regards to the network crawl, it would be valuable to provide the user feedback and status messages based on the crawl-status in relation to completeness. This may be implemented with a progress bar on the interface, instead of the continuous circular loading symbol next to the tab’s header itself. It would also be useful to allow users to enter a range of IPs to scan in the interface. This narrows down the amount of devices the crawler must search while providing the user an interactive role in the scan. Moreover, it is recommended that an admin is able to run a scan and add devices for a set client through the interface. The administrator must have full control and access to client data, and adding this to the UI would improve the management of the inventory data. Finally, ideal future development involves packaging and live deployment for ease of use and simple product distribution.

**4 System Architecture and Design**

The software inventory tool's system architecture and design are meticulously crafted to meet the multifaceted demands of our client’s requests for a device inventory management system. At its core, the architecture revolves around a robust database system, ensuring security, efficiency, and accessibility in inventory retrieval. With the implementation of admin privileges, we are able to guarantee that only authorized personnel can access sensitive records, bolstering data integrity. Comprehensive software testing, underpins reliability and functionality, with a keen focus on network integration. The crawler agent is a pivotal component in building the inventory database. Generating report CSV files allows the user to download data in a user-friendly format. The UI prioritizes user experience, offering an intuitive interface for enhanced usability. Finally, comprehensive documentation provides transparency and support throughout the project lifecycle. Together, these components form the foundation of a sophisticated inventory management system tailored to meet the evolving needs of professionals.

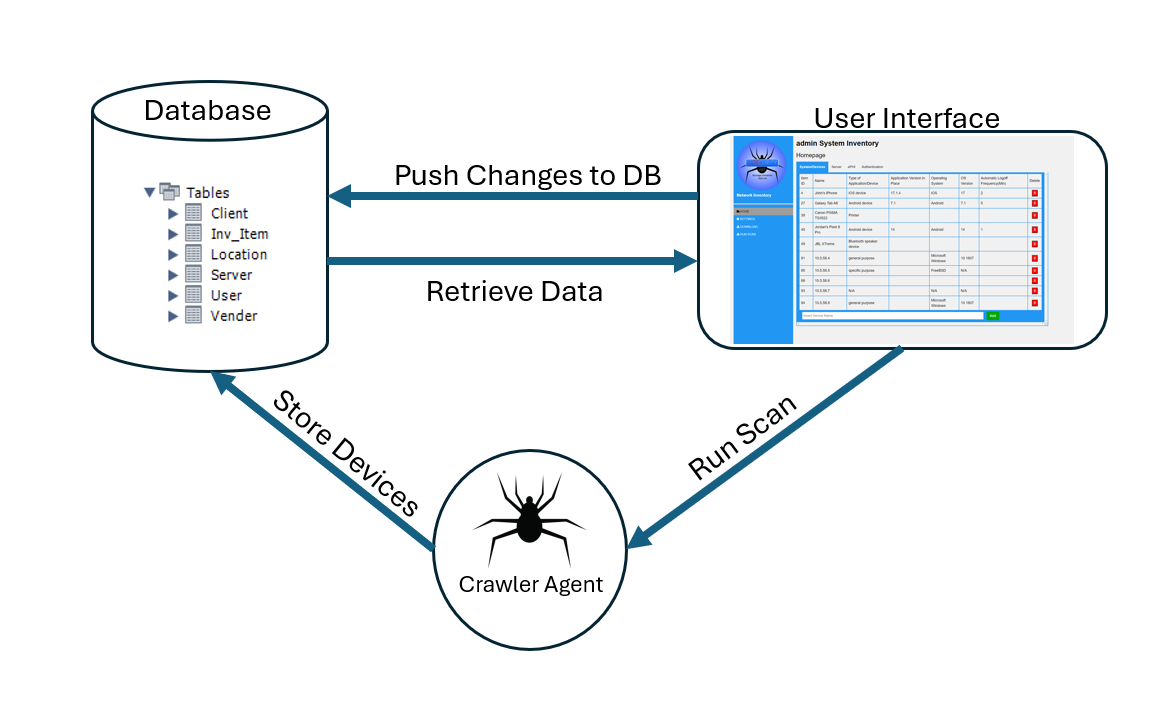
**Table 1: Major Program Components**

| *Component* | *Description*  *Brief summary describing the component and significance (as appropriate)* | *Team Member Responsible for Feature* |
| --- | --- | --- |
| *Database System* | The database management system will be secure, easily queryable, and organized into tables for efficient inventory retrieval and filtering by professionals. Rows and columns of a table are defined by the relationship between the client’s network and the software and device information connected to it. Columns detail Software System/Medical Devices, Server Info, Electronic Patient Health Information (ePHI), and Authentication Methods. There will be an API for the crawler to easily plug into. The database will use mySQL and be hosted by Medcurity on AWS. | Brandon H. |
| *Admin Privilege* | Administrator accounts owned by authorized personnel will be given access to records in the database. Together with the UI, a secure login page will prompt the admin for their credentials to proceed. Inventory records must be accessed by licensed professionals. | Artis N. |
| *Software Testing* | The software inventory tool will be tested thoroughly to assert functionality and reliability. This will include building a test network for our crawler to test functionality on. This integration testing will utilize the Crawler Agent to transfer data to the database and confirm every component functions as engineered and intended. | Jack N. and Colleen L. |
| *Crawler Agent Traversal* | A crawler agent that will traverse the network it is connected to, searching for devices and software that are also connected to the network. Traversing the network in this way is required to build up the inventory database. Building the database will be done with an API as it traverses. This is in response to the business objectives to provide the client with an inventory of software and devices to assist with HIPAA regulations. The crawler uses nmap, netifaces and various other python libraries to traverse and collect network device information. | Colleen L. |

| *Report (CSV)* | A CSV report one of the main sources of output that the user will receive. With a CSV file, data can easily be imported into a spreadsheet or a different means of data visualization. It could also be implemented as a future item as part of the UI, allowing the user to access the data and view it in a filtered and hassle-free way. | Brandon H. and Jack N. |
| --- | --- | --- |
| *Manual Input* | Manual input is important because it is one of the main ways the user will interact with the software. The user should be able to interact with the interface, logging in with their credentials, add, update, and remove device information, choose when to export the data into a csv file, and more. | Artis N. and Brandon H. |
| *User Interface (UI)* | As a primary component of the program’s front end, users will interact with an intuitive and visually appealing user interface for the means to view, analyze and control the program’s behavior. The goal is to create a seamless and user-friendly interface that enhances the overall functionality and accessibility of our inventory tool. This feature directly impacts the usability of the program and the overall user experience so it was important to produce a high- quality, easily accessible and comprehensive inventory tool. The UI is programmed using a combination of HTML, PHP, and JavaScript languages. | Jack N. and Artis N. |
| *Documentation* | Project documentation is essential to the project lifecycle from planning and development to deployment and maintenance. The scope encompasses a comprehensive report of all aspects of the project ensuring transparency. Key components include the project guidelines, requirements, design and architecture, guides, logs, API usage, version history, and user guides that will aid in current and future development especially concerning maintenance. Proper documentation that is clear and concise will aid in our initial development as well as anyone else maintaining or building upon this software inventory tool. | Colleen L. and Jack N. |

The overall project layout can be seen in Figure 1 where we can see the general interactions between the three largest project components. Starting with the database, the schema is set such that when the crawler is ran, the information is stored into a couple different tables including client information, individual device information, locations, servers, users and venders. From this database, the UI is able to query for information to then show in a table to the user. The user is able to add a new device, delete an existing one, or modify an existing device’s information. The user can activate a new scan and download the existing database information for their respective client. Admin users have access to the entire database and is able to see all entries currently stored in the general database. A visual representation of the system structure can be seen in the following figure.

**Figure 1**



\* Figure 1 visualizes the relationship between our database, crawling agent and the user interface.

**5 System Evaluation**

During development, the system was evaluated and tested through the use of a testing network with a combination of both wireless and wired connections from a variety of devices. While it was initially a bit difficult to get technology, we were able to acquire the following devices:

• Qwest Actiontec Q1000 DSL Modem/Router 4-Port Wireless Router

o Security type: WPA2-Personal

• Raspberry pi running linux

o Connected with ethernet to the Qwest router

• Dell Laptop running Windows

o Connected Wirelessly

• HP Laptop running Windows

o Connected Wirelessly

• IPhone 5 running IOS

o Connected Wirelessly

These devices helped to create a small subnet that we could run our crawling agent on in order to develop and test the system in a controlled environment isolated from Gonzaga’s main campus wifi. The Qwest Actiontec Q1000 DSL Modem/Router served as the central hub, providing both wired and wireless connectivity. The Dell and HP laptops running Windows, the iPhone 5 running iOS, and the Raspberry Pi running Linux represented a diverse set of client devices, allowing us to assess the system's compatibility across different platforms. By leveraging these devices within a confined network environment, we could monitor and analyze the system's behavior under different network conditions and usage scenarios. This approach helped to enable us to identify and address potential issues proactively, ensuring the robustness and reliability of the system during the development process.

Additionally, university scheduled peer-evaluated testing events held on the 20th and 27th of March aided us in identifying our program's points of vulnerability and areas of weakness. From these events we were able to identify system bugs both big and small ranging from incompatible data types to flaws in scope for manual input to database relations. These testing events were a crucial part of our current program and aided us in increasing our overall test coverage.

As the final part of our system evaluation, we were able to meet with Amanda and Rachel from Medcurity to provide some additional system testing and overall project review. This helped our team to finalize the layout for how the program will be released at the end of this semester.

**6 Project Delivery, Deployment, and Maintenance**

The project is to be delivered through providing Medcurity access to the GitHub repository that contains all of the source code with our project, which will be done by sending our Medcurity sponsors an email link to our GitHub repository. Along with the GitHub repo link, we plan on packaging the product into a zip file for them to easily download and access. Thorough documentation that details the installation and setup process, the purpose of each component of the project, and comments throughout the code in each file, allows Medcurity to have an easy time understanding each part of the project, making the transition process seamless. Each file contains a header comment block that contains the author of the code and their emails, so should Medcurity have any questions they are able to contact any of the team members through those respective email addresses.

**7 Project Management**

Overall the team performed well and the core functionality of the project works. There were some features that the team initially wanted to implement from the start of the first semester but was later scrapped, such as limiting network scan ranges or extensive data filtering through the user interface. Conversely, there were features that were added that the team did not initially anticipate, such as having two types of users: Medcurity administrators and client users, as well as handling their respective functionality and permissions.

The way the project was split up was that Brandon primarily handled the database side of the project, Colleen handled the crawler, Artis handled the login page, and Jack handled the main user interface. These four components were not restricted to each assigned group member however, and each member helped in other components of the project as needed.

Brandon focused on implementing the database as well as writing and integrating Python scripts for various functionality. His major accomplishments were defining the database relations, specifically in regards to mapping client requirements to schemas, defining attribute data types, and normalization, as well as creating a Python API for the database for secure access for various components of the software that were implemented in Python, including the crawler and the CSV export scripts. Considerable amounts of time were also devoted to the creation of thorough documentation on each of the previously described components. Brandon typically worked eight hours per week, including various team meetings and presentations. As the project progressed, task length estimation became better, both in terms of under- and over-estimation. Places for further developments and refinements in Brandon’s work include the better handling of duplicate entries into the database and the dynamic writing of the header row in the CSV export scripts to support any changes that may be made to the database schema and script function calls in the future.

Colleen focused on implementing the crawler agent and researching the most effective ways to gather information from a network in a secure fashion. She researched ways to ping devices, send API requests, and execute system commands from Python scripts. Beyond this research, she coded numerous functions which return the desired attributes from the Sponsor-provided spreadsheet. The crawler script outputs messages in the terminal, if run outside of the localhost environment, and the script is mainly driven using Nmap, a lightweight tool for network scanning. Approximately 9-10 hours weekly were dedicated to project development, completing sprints, and extensive research. Together with Brandon, she was able to fully integrate the crawler with the Python API so that data is dynamically pushed to the database upon the scan in the user-interface. She took on backlog items using issues and pull requests on the Github repository, and documented important decisions in the various project documentation files.

A stretch goal for the project was the ability for users to input the range of IP addresses to be scanned, and although this offers the user more control over their scan, our timeline had to push this task out-of-scope. Additionally, multithreading was a personal goal of hers to improve execution time, but port-scanning took up the majority of time, so implementing functionality did not significantly improve runtime.

Artis initially spent time working on portions of the main user interface and contributed to the initial high-level rough design of the database. As soon as the team realized the project needed a secure login page and eventually the ability to support two types of users, he primarily spent his time on implementing the login page. He worked on handling what administrator users can do over what normal client users can do and adding different features for the client user in the login page. Each process had to be secure, so encryption methods were put in place for the final product. Approximately seven to eight hours a week were dedicated to every component that led to the final login product. This included research, both of the weekly team meetings, development, bug-fixing, documentation, and other important tasks that led to our final product.

One request for the login aspect of the project, that was made by Medcurity, later on in the project timeline was the desire to allow for Medcurity client users to create their own administrative user accounts, allowing them to create and delete client users themselves. Unfortunately, we were not able to implement this feature due to the amount of time left and converted it as a feature for future development.

Throughout the project, Jack served as the main architect and designer of the system user interface and was a part of the crawler and testing research teams. Originally, his workload was focused on the design and implementation of the crawler agent while developing and creating related building blocks in HTML/PHP/JS to use in the user interface. After the initial development period it was evident that the user interface was falling behind where Jack took it upon himself to bring the user interface up to speed and continue its development as its primary developer. Working on average close to 7-8 hours a week on the project, Jack put a lot of time into making sure that the user interface would work as intended with as much functionality as could be developed in our time frame. Additionally, Jack did a lot of logistical management including the creation of the github repositories and project pages, creation of our discord communication server, weekly meeting room scheduling and reservation, creation and management of our network subnet used for testing, and unofficial team scrum. Weekly time estimates were generally correct with a majority of tasks completed within our estimated time frame indicating healthy project growth.

Further development that was pushed out of scope includes the display of network information in the settings page and also to make it so that all cells on the homepage can be edited and changed by the user.