**GONZAGA UNIVERSITY**

**School of Engineering and Applied Science**

**Center for Engineering Design and Entrepreneurship**

**PROJECT PLAN**

**October 19, 2023**

**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**

# **Project Summary**

Online scammers and hackers are capable of causing serious damages. Whether motivated by ransom money, identity theft, or power and status, attackers are able to leverage their unauthorized access to systems, placing individuals and companies at high risk for digital security breaches and malicious attacks. Notably, these targeted attacks are strategic, and mainly aim to exploit large amounts of sensitive and private data. Thus, to be proactive, professionals in a data-driven field must allocate and invest in resources to engineer layers of security. For instance, implementing these security precautions will allow professionals in the medical industry to ensure private medical records are kept confidential and to protect client safety and data. Adhering to medical record compliance protects sensitive patient health

information from being disclosed without patient consent or knowledge and thus follows our Sponsor’s mission of helping clients meet their federal digital security standards, known as *HIPAA* standards.

Our software inventory tool implements security precautions with an authentication process, requiring that an administrator or client enter correct credentials to access the tool. Next, the tool will send out a special agent that scans the client’s network and adds the information discovered to our database. The agent is looking for server identification addresses, information about devices on the network, location of the server, and encryption methods, among other attributes. After data has been loaded into the database, we give the user access to an exportable customized spreadsheet report that details the information discovered. Identifying devices and software within the network domain will

demystify connections that may be unsafe to allow into the network. To ensure clients will only be able to access their own information, the tool will link client information to only their accounts, reinforcing authentication and checking the user has access to their information and no one else’s.

# **Project Objectives**

The main desired outcome of the Medcurity Network Inventory Team is to develop a new tool that complies to HIPAA standards and can be used by the sponsor to help in their goal of organizing, managing, and controlling the numerous tasks healthcare organizations are responsible for. The tool must be able to scan a client’s network for devices that are currently connected to it and identify the client’s needs and diagnose issues. While the solution is a standalone product, it should work seamlessly with the sponsor’s current environment and should be easy to understand, access, and deploy. The resulting product should be able to output customized reports, either through a CSV file or Microsoft Excel

spreadsheet, that have filtering capabilities and be easy to read. Documentation is necessary in order to allow future developers to understand the code and sufficient testing is necessary to ensure the project works as expected.

# **Project Stakeholders**

The main stakeholders of the project include our Liaisons, Rachel Kunkel and Amanda Hepper, who are both directly affiliated with Medcurity. They are responsible for communicating with the Medcurity Network Inventory Team, relaying specifications for the project, and addressing any overall questions the team may have.

The design advisory board (DAB) member, Richard Weeks, who works at F5 as a version control specialist, is responsible for giving advice and guidance. Since the DAB members have had experiences in past projects and qualifications, they are able to provide a different perspective to the rest of the team and are able to help solve specific technical issues the team may run into.

Similar to the DAB member, the advisor, Mike Mudge, who is a senior manager at Avista, provides guidance to the team. Unlike the DAB member, the advisor is responsible for guidance relating to the high-level aspects of the project. They are also responsible for evaluating the team’s progress, routinely checking up with them to ensure they are on pace to complete their project.

The main developers are the students at Gonzaga University, Brandon, Colleen, Artis, and Jack. They are responsible for the majority of the project work and are expected to collaborate through good communication, organization, and providing unique perspectives to any given scenario. Their work and their process aligns with what their future career will potentially look like.

Along with the DAB member, the Medcurity IT department members are the ones likely to provide the team specific information regarding the content of the project. Since they are responsible for upholding the current software systems at Medcurity, they will be able to give tips and warnings on how the team will integrate the new software tool that will become a part of their current system.

The network management departments are a part of the general industry. The team’s work is

going to contribute to Medcurity’s systems, which are involved in the network management systems. This means it is important to consider the general industry standards relating to this subject when the team is assembling their software solution as it may lead to unforeseen circumstances. Medcurity also has a lot of clients so they must be accounted for when developing the project.

# **Project Deliverables**

The main deliverable will be the software tool as described in 1.2 and 1.5. This includes a network crawler agent, a database, an Excel spreadsheet, and a UI. The network crawler agent and the UI will be packaged to be executed as a bash command. When the command is executed, the UI will pop up, triggering the network crawler agent; when the agent is complete, it will export its findings to the Excel spreadsheet. The database will be hosted in the cloud on AWS. In addition, documentation for users and future developers, testing methods and results, and a report describing the future work that could be added will be provided. All of this will be submitted via a Git repository with a README.md file.

# **Project Scope**

This project is a stand-alone tool, and as such there are minimal out-of-scope aspects to consider. It has four in-scope features: the agent that scrapes the network for devices and software, the database that stores the devices and software that the agent finds, the UI that displays status and results and offers minor user interaction functionality, and the Excel spreadsheet that will be generated from the database. There is not any existing software that our tool will need to be integrated into. Perhaps noteworthy, our database will be MariaDB using MySQL hosted on AWS, all of which could be considered out-of-scope.



# **Figure 1**

**1.6 Related Work**

As a main component of our project, working with security and IT management, there are many software development companies that invest in and offer network scanning and device-tracking applications along with supporting management software.

Auvik1 —

The network management system offered by Auvik Networks provides real-time network discovery and mapping by collecting data from various sources allowing users to visualize their network's structure and understand how devices are connected, mirroring our project plans.

Additionally, Auvik automates inventory and documentation, capturing information including the device’s make, model, serial number, IP address, and switchport connections. These additional features lie outside of our current project plans but could potentially be implemented after initial designs with sponsor’s permission. Lastly, the Auvik program assists in hardware lifecycle management by identifying devices that may need upgrades or replacements. It retrieves data about support contracts, available software updates, security patch eligibility, and device availability for purchase. This feature will also not be included in our project plans.

Famatech2 —

Provided by the Famatech corporation, the “Advanced IP Scanner” shows all the connected network devices and provides remote control of computers via RDP and Radmin. Additionally, the program also gives the user access to shared folders. Comparing this software to our project, the only major difference is that the Advanced IP Scanner uses Radmin, a third-party company, for remote connection.

SolarWinds3 —-

The “User Device Tracker” offered by SolarWinds shares some close similarities with the plans we have for our project and has some shared functionality. Altogether, shared functionalities include the ability for the client to identify users and devices on their network with a comprehensive network topology and store that in a database for further analysis. Additionally, the SolarWinds program offers an active directory integration with an interactable interface allowing the user to whitelist hosts, track sensitive or suspicious devices and remotely turn switch ports on and off. As for current plans, this additional functionality offered by the SolarWinds program is not planned for our project, however, can be added as potential features to be added after initial development.



1 Auvik Network Management - <https://www.auvik.com/features/>

2 Famatech Advanced IP Scanner - <https://www.advanced-ip-scanner.com/>

3 SolarWinds User Device Tracker - <https://www.solarwinds.com/user-device-tracker>

1. **Project Requirements**
   1. **Major Features**

# **Table 1: Major Features**

| *Feature* | *Description*  *Brief summary describing the feature and significance (as appropriate)* |
| --- | --- |
| *DatabaseSystem* | The database management system will be secure, easily queryable, and organized into tables for efficient medical record 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. Sponsor is hopeful for columns to detail Software System/Medical Devices, Server Info, Electronic Patient Health Information (ePHI), Authentication Methods, Location, Purchase, Quantity/Value, and Asset Information. |
| *AdminPrivilege* | Administrator account(s) 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. Client records must be accessed by licensed professionals to properly access client needs. Thus, these privileged accounts will be able to add, update, and remove client information to best fit the situation at hand. |
| *SoftwareTesting* | The software inventory tool will be tested thoroughly to assert functionality and reliability. Furthermore, fully integrating the tool with the Sponsor’s software calls for extensive testing of compatible software, usability, security, and accuracy. This integration testing will utilize the Crawler Agent to transfer data to the database and confirm every component functions as engineered and intended. |
| *CrawlerAgentTraversal* | 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. This is in response to the business objectives to provide the client with an inventory of software and devices to assist with HIPAA regulations. |
| *CrawlerAgentDatabase* | The crawler agent will need to integrate with the database. While traversing the network, the crawler agent will find software and devices that are also connected to the network. To report the software and devices found, it will store them as it finds them into a database, building it up 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. |

| *Report(CSV)* | A report CSV file will be 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 part of the UI, allowing the user to access the data and view it in a filtered and hassle-free way. |
| --- | --- |
| *ManualInput* | 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 client information, choose when to export the data into a csv file, and more. |
| *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 is important to collaborate with our project stakeholders in order to produce a high- quality, easily accessible and comprehensive inventory tool. |
| *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. |

* 1. **Initial Product Backlog**

# **Table 2: Initial Product Backlog**

| *Requirement* | *Description*  *Brief summary describing requirement including acceptance criteria.* | *Major Feature From Table 1* | *Priority* | *Estimate* |
| --- | --- | --- | --- | --- |

| *DatabaseType* | Use a relational database with querying abilities and enough capacity for millions of entries, such as SQLite (140 TB) or MySQL. | DatabaseSystem | High | Low |
| --- | --- | --- | --- | --- |
| *DatabaseCapacity* | Document the maximum needed size of the database to account for all client information and software associated. | DatabaseSystem | Medium | Low |
| *DataStorage: Name of Software/Device* | Dedicate column to the name of the specific software or device being recorded in the database. | DatabaseSystem | High | Low |
| *DataStorage: Type of Software/Device* | Dedicate column to the type of specific software or device being recorded in the database. | DatabaseSystem | High | Low |
| *DataStorage: AppVersion* | Dedicate column to the application version in place at the time of data retrieval. | DatabaseSystem | High | Low |
| *DataStorage: Operating System & Version* | Dedicate column to the type of operating system and version of software or device being recorded in the database. | DatabaseSystem | High | Low |
| *DataStorage: ServerName* | Dedicate column to the server name of the specific software or device being recorded in the database. | DatabaseSystem | High | Low |
| *DataStorage: ServerIP* | Dedicate column to the server IP address of the specific software or device being recorded in the database. | DatabaseSystem | High | Medium |
| *DataStorage: Server On Cloud or Premise?* | Dedicate column to storing if the specific software or device is on the Cloud or on premise. | DatabaseSystem | High | Medium |
| *DataStorage: ServerLocation* | Dedicate column to storing the server’s accurate location in the database. | DatabaseSystem | High | Medium |
| *DataStorage: EncryptionApplied* | Dedicate column to storing if the server is encrypted or | DatabaseSystem | High | Medium |

|  | not. This will be stored as a YES/NO |  |  |  |
| --- | --- | --- | --- | --- |
| *DataStorage: EncryptionMethod* | Dedicate column to storing the server’s encryption method if encryption applies to the software or device. | DatabaseSystem | High | Medium |
| *AdminList* | Create a list of authorized personnel to access the database and resources. | AdminPrivilege | Medium | Low |
| *AdminUsernames* | Establish test cases of admin accounts and assign unique usernames to each. | AdminPrivilege | Medium | Low |
| *AdminPasswords* | Craft secure passwords for test cases of associated admin accounts. | AdminPrivilege | Medium | Low |
| *AdminAbilities* | Document the privileges and software admin users will be able to update and access. | AdminPrivilege | Medium | Low |
| *AdminUserManagement* | Ensure and determine how administrators are able to create, update, and delete user accounts and permissions within the tool. | AdminPrivilege | Medium | Low |
| *AdminDataMaintence* | Ensure and determine how administrators are able to monitor and maintain stored data. | AdminPrivilege | Medium | Low |
| *AdminBackup* | Ensure and determine how administrators are able to backup and recover data in the case of system failure. | AdminPrivilege | Medium | Low |
| *AdminReports* | Ensure and determine how administrators are able to generate reports on software inventory in the database. | AdminPrivilege | Medium | Low |
| *TestingEnvironment* | Determine ideal environment to host extensive testing on software, and if this will be on a VM or local computer. | SoftwareTesting | Medium | Low |

| *TestingAccuracy* | Create methods or scripts to confirm entries are as expected to prevent tampered data or inaccurate information. | SoftwareTesting | Medium | Medium |
| --- | --- | --- | --- | --- |
| *TestingCrawlerAgent* | Verify and test the datasets used by the crawler agent to ensure correct investigation and reports. | SoftwareTesting | Medium | Medium |
| *TestingDefects* | Establish a process to report and manage errors or issues that arise with various inputs to the database, crawler agent, and UI among other major features. | SoftwareTesting | Medium | Low |
| *TestingSecurityCompliance* | Adhere and ensure compliance with HIPAA standards in all areas of the inventory tool. | SoftwareTesting | Medium | Low |
| *TestingSecureLogin* | Ensure UI and login system is free from cyberattacks like unauthorized access. | SoftwareTesting | Medium | Medium |
| *TestingManualInput* | Proactively check for invalid input to prevent attacks or inaccuracies. | SoftwareTesting | Medium | Low |
| *TestingIntegration* | Confirm and document seamless integration abilities between Sponsor software, CSV reports, UI, database, and crawler agent. | SoftwareTesting | Medium | Low |
| *CrawlerAgentStyleInvestigation* | Investigate various crawler agents, seeing how they work to assist in designing our own | CrawlerAgentTraversal | High | Low |
| *CrawlerAgentUsableInvestigation* | Investigate various crawler agents, seeing if there is an existing one that we could use | CrawlerAgentTraversal | High | Low |
| *CrawlerAgentMultipleNetworks* | Figure out if the crawler agent needs to crawl multiple networks sequentially, and if so how | CrawlerAgentTraversal | Medium | Low |
| *CrawlerAgentTargetRecognition* | The crawler agent needs to recognize the software and | CrawlerAgentTraversal | High | Medium |

|  | devices we want to add to the database |  |  |  |
| --- | --- | --- | --- | --- |
| *CrawlerAgentSearchAlgorithm* | Determine the best algorithm for the crawler agent to follow when traversing the tree (e.g. depth vs breadth) | CrawlerAgentTraversal | Medium | High |
| *CrawlerAgentSearchExtent* | Determine how deep and wide the crawler agent needs to search the network for target software and devices | CrawlerAgentTraversal | Low | Medium |
| *CrawlerAgentRoot* | Determine where the crawler agent will begin its traversal | CrawlerAgentTraversal | High | Medium |
| *CrawlerAgentTargetIdentification* | Determine what software and devices the crawler agent needs to recognize and how it will be able to do that | CrawlerAgentTraversal | High | High |
| *CrawlerAgentProgress* | Determine how to report current progress status of crawler agent to the UI, informing the user | CrawlerAgentTraversal | Medium | Low |
| *CrawlerAgentLaunch* | Determine how to launch the crawler agent, through bash or the UI | CrawlerAgentTraversal | High | Low |
| *CrawlerAgentParsing* | Determine what information the crawler agent needs to parse from a target | CrawlerAgentDatabase | High | High |
| *CrawlerAgentScrubbing* | Determine how the crawler agent will scrub the data it needs to collect from the target, including the best data format | CrawlerAgentDatabase | High | High |
| *CrawlerAgentDatabaseLink* | Determine how the crawler agent will be linked to the database | CrawlerAgentDatabase | High | High |
| *CrawlerAgentDataTransfer* | Determine how the crawler agent will transfer the information it scrubbed from a target to the database | CrawlerAgentDatabase | High | Medium |

| *CrawlerAgentTransfterFail* | Determine how to handle when the crawler agent attempts to perform a data transfer but it fails | CrawlerAgentDatabase | Low | Medium |
| --- | --- | --- | --- | --- |
| *CrawlerAgentDuplicateData* | Determine how to handle when the crawler agent attempts to catalog a duplicate target (potentially out of scope for this major feature) | CrawlerAgentDatabase | Low | Medium |
| *ReportFileSizeCheck* | Ensure that CSV imports work by ensuring that the file is not too large due to too many fields or rows. | Report(CSV) | High | Low |
| *ReportEncodingCheck* | Verify that the CSV file is UTF-8 encoded otherwise data may appear nonsensical when imported. | Report(CSV) | High | Low |
| *ReportMissingData* | Check if any data fields are incomplete and must be filled out. | Report(CSV) | High | Medium |
| *ReportFileTypeCheck* | Files may be saved with the wrong extension and cause issues. Ensure they are saved with the csv extension. | Report(CSV) | High | Low |
| *ReportHeaderCheck* | Check if the header columns are not missing and formatted correctly. | Report(CSV) | Medium | Medium |
| *ManualInputLogin* | Users must know how to login with their credentials properly in order to gain access to the software. | *ManualInput* | High | Medium |
| *ManualInputEdit* | Verified users must be able to manually edit data from new and existing data from patients and organizations. | *ManualInput* | High | High |
| *ManualInputInvadidCheck* | Ensure that data that is manually inputted follows the same format and syntax so the database can function properly. | *ManualInput* | High | High |
| *ManualInputReports* | Allow users to be able to export data into a report when they choose to. | *ManualInput* | High | Medium |

| *ProgressBar* | User will see a progress bar accompanied by a status message | *UI* | Low | Low |
| --- | --- | --- | --- | --- |
| *DesignCSVDownload* | Main user program interaction will be with a downloaded CSV file opened in Excel | *UI* | Medium | Low |
| *Documentation - Gonzaga CPSC491 class requirements* | Gonzaga 491 assignment check-ins with team document submissions for:  Sec02 - Requirements Section - Draft v0.1   * Sep 17   Sec03 - System Design Section - Draft v0.1   * Sep 24   Sec01 - Overview Section - Draft v0.2   * Sep 27   Sec02 - Requirements Section - Draft v0.2   * Oct 1   Sec04-07 - Risks, Release, Management, and Maintenance Sections - Draft v0.1   * Oct 8   Full Plan Draft release v0.9  - For adviser review   * Oct 9   Project Plan Presentation - SEAS CEDE Event   * Oct 18   Project Plan Document - v1.0   * Oct 19 | *Documentation* | High | High |
| *Documentation - Backlog* | Each backlog item should be accompanied by clear and concise descriptions, priority levels, and estimates, allowing the current and future development team to understand the scope and importance of each item. This documentation helps facilitate effective communication within the team, ensuring everyone is aligned on project goals and tasks. | *Documentation* | High | High |

* 1. **Additional Features**

**Table 3: Additional Features**

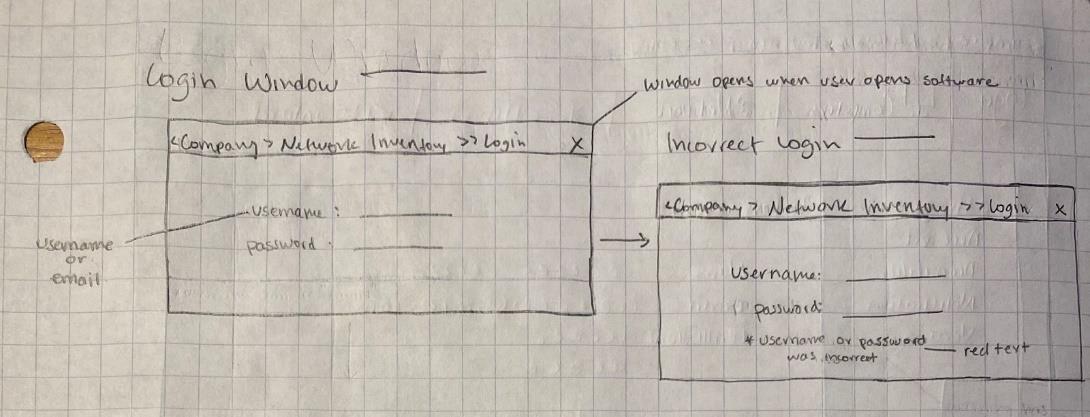
| *Requirement* | *Description* | *Priority* | *Estimate* |
| --- | --- | --- | --- |
| *PerformanceTesting* | Performance testing results showing if, how, and where the system is optimized for performance, where system performance could be improved, and an accompanying report | High | Medium |
| *Style* | Maintaining consistent style in coding and the way the user interface will look is something that must be considered in the design. The team will have to agree on a certain style to follow in order to make the code and interface as clear as possible. | Medium | Low |
| *Login* | Initial Development plans do not include interaction with confidential information that could pose potential legal action. Administrative accounts with differentiated permissions should be determined close to v1.0 release and should reflect current software abilities. | Low | Low |
| *UIWebInterface* | As a stretch goal, the user will be able to utilize a web interface to manage their network inventory. | Low | Low |

1. **Design Considerations**

# **Initial User Interface Design**

Version 1 Sketch and Storyboard —-

The login window will appear when the program starts up.

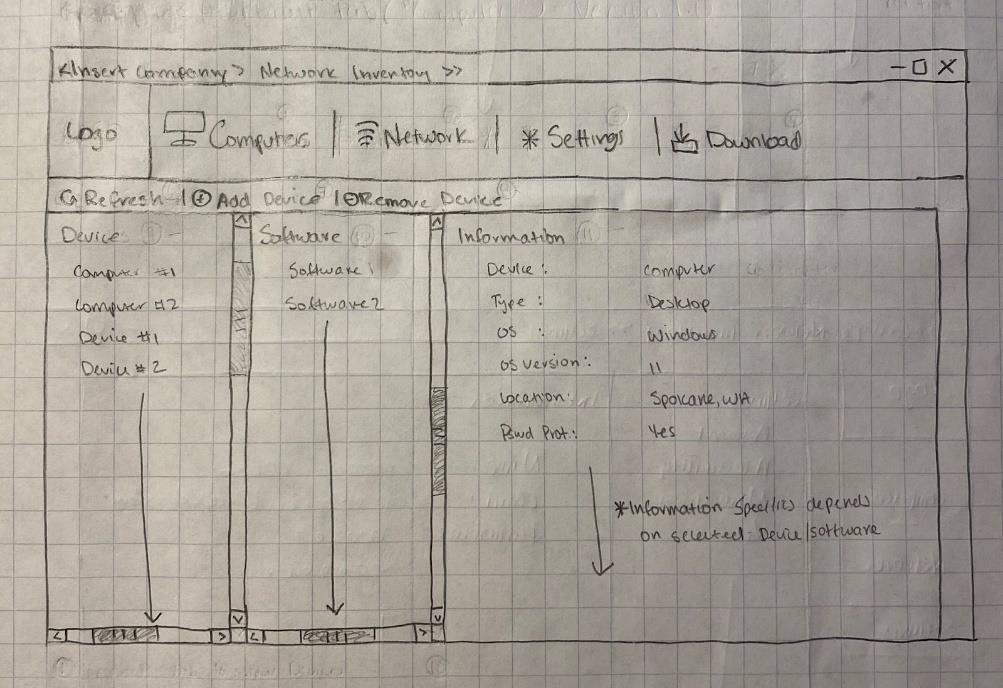


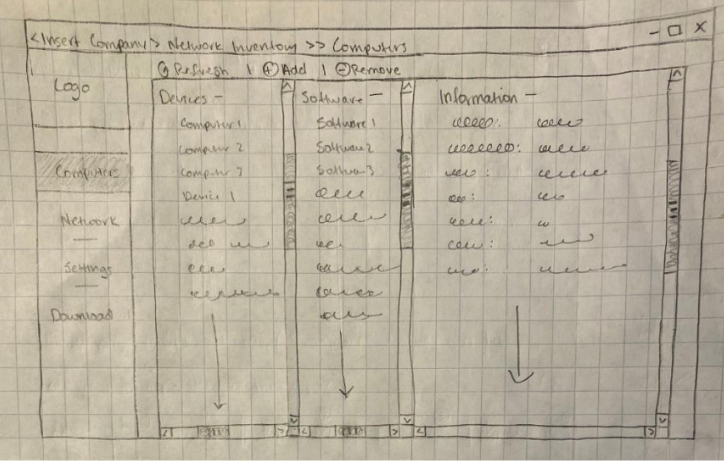
Version 1 of the main homepage which will appear after the user correctly enters their username and password in the login window. This homepage is the same as the “computers” tab in the sketch.

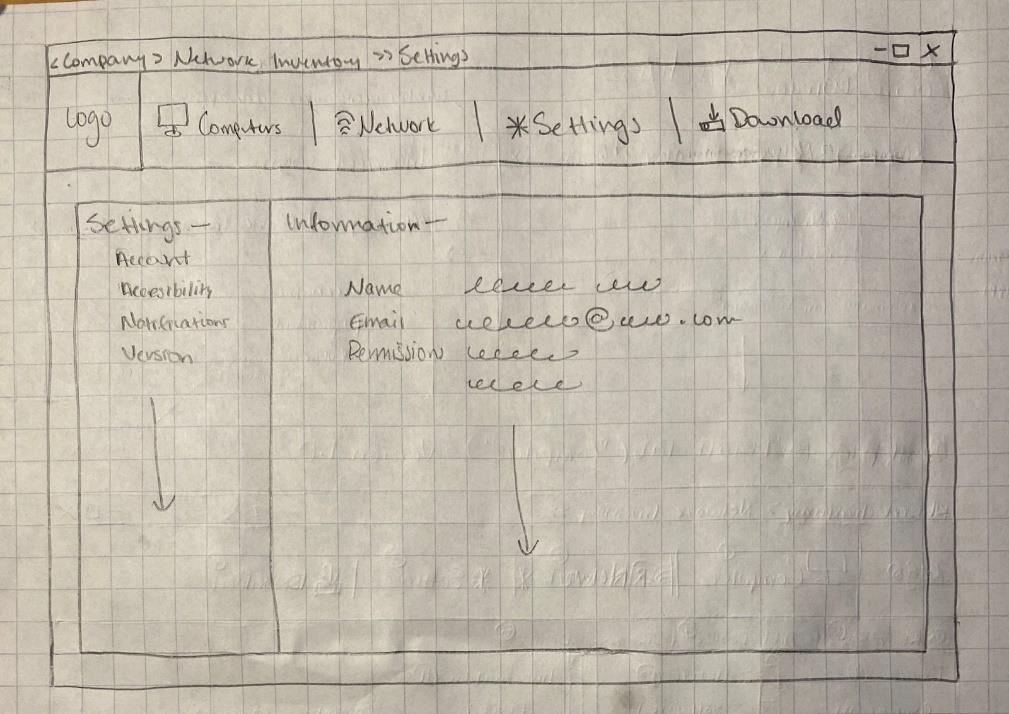
Version 2 of the main homepage which will appear after the user correctly enters their username and password in the login window. This homepage is the same as the “computers” tab in the sketch.

Login Window

Homepage Version 1



Homepage Version 2

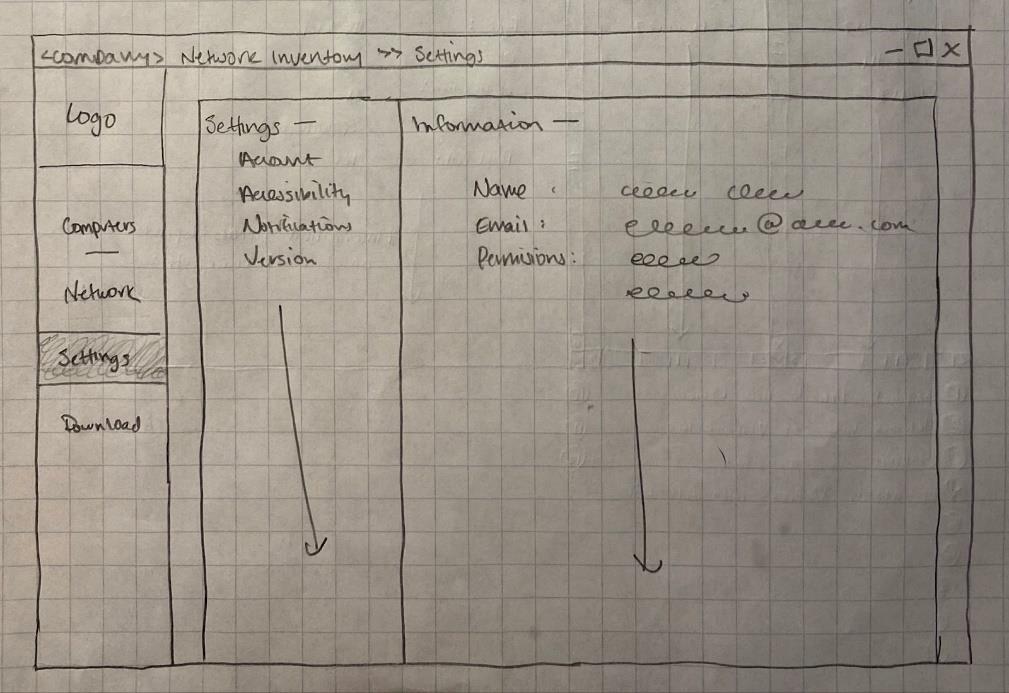
The settings versions 1 tab

will show account Information,

accessibility options, the

software version and other

miscellaneous settings.



The settings versions 2 tab

will show account Information,

accessibility options, the

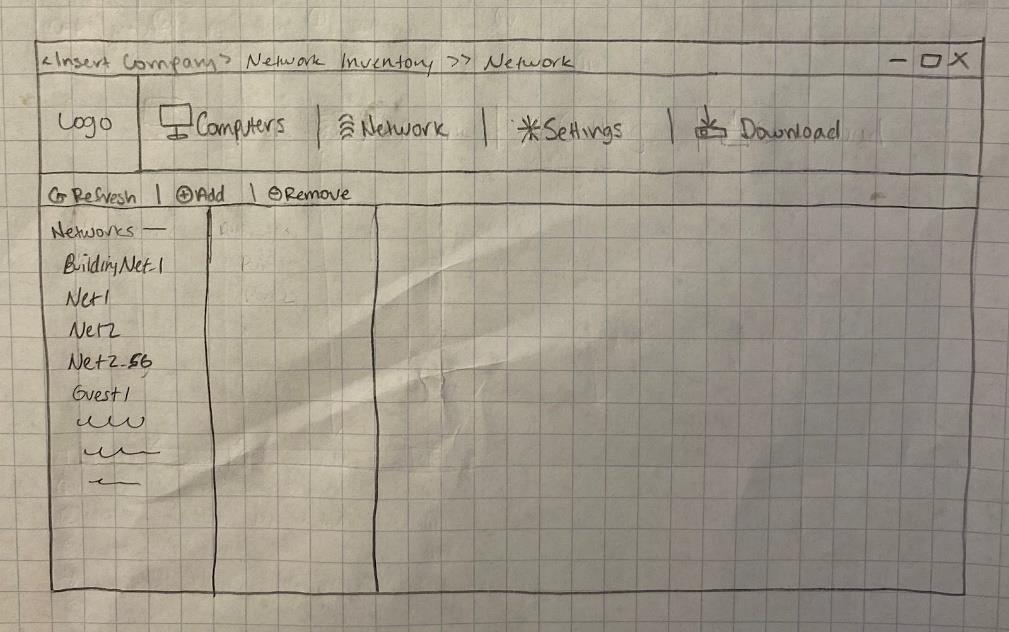
software version and other

miscellaneous settings.

Settings Version 1

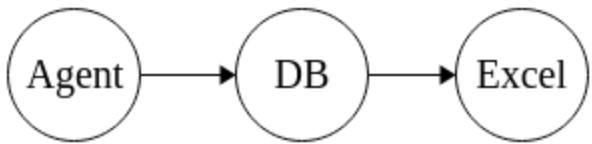
Settings Version 2

Network Version 1

The network version 1 tab Will show the setwork Configurations and other Information regarding the Network and network Technologies themselves

# **Initial Software Architecture**

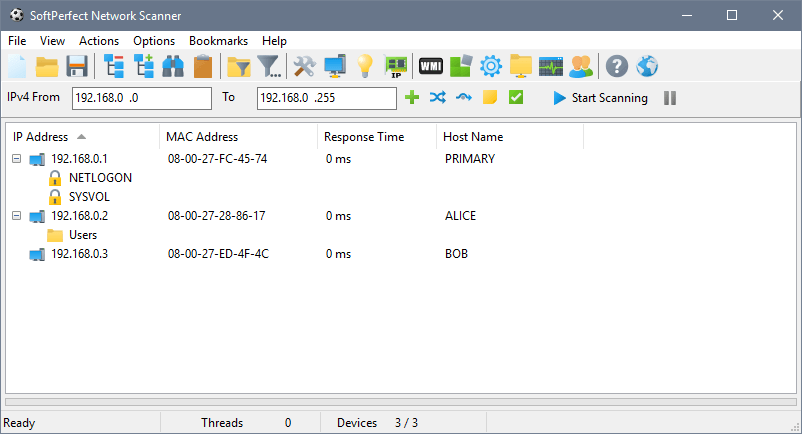
Our system consists of two main components: the network crawler agent and the database. The crawler agent will be launched from the terminal. As it crawls the network and finds target devices and software, it will insert them into the database, updating it. When the network crawl is complete, the database will be exported to an Excel spreadsheet, following the format of the example spreadsheet.



# **Development Environment, Tools, Languages, and Libraries**

We expect our major features to require development tools and dependencies. Beginning with the retrieval of data, the crawler agent will utilize the terminal to launch its search for networks and devices connected to it; this requires a well-built network scanner for detecting devices which can be used from an existing program such as SoftPerfect Network Scanner. This software scans TCP ports for internal and external IP addresses, and provides an exportable CSV (among other) formatted reports which is ideal for our Sponsor. The tool uses nmap and ssh commands, provides data on operating system discovery, and conducts vulnerability tests (<https://www.softperfect.com/products/networkscanner/>). This specific software will cover most of the crawler-needed elements of Software System/Medical Devices, but the crawler agent will also need to

scrape information about server location and if it is located in the Cloud or on the premise, and ePHI with or without encryption. Once the crawler gets this information, it will be stored into the database.



Our database will use either MariaDB or SQLite based on Sponsor’s capacity needs and which will be easiest to implement into the company's existing software/systems. The DB must store server information and device information noted above in addition to user authentication methods and credentials to login securely. *The DB will also hold information about Location: department/space location/date of last order, Purchase: vendor, purchase price per item/warranty expiry date, and Quantity/Value: condition, quality, asset value/total value, and Asset Information: model, vendor NO./remarks/photo or link. These attributes will likely be provided without having to use the crawler agent.* Database IDE could be dbForge which is compatible with a non-local stored MariaDB or MySQL DB. To validate credentials, the DB will use either a query that matches login information or it utilizes libraries like <https://mariadb.com/resources/blog/how-to-connect-python-programs-to-mariadb/>.

If UI is necessary, it will likely use HTML/CSS to create the interface and connections to DB and

agent.

# **Initial Software Test Plan**

Our software test plan will have to validate our user interface (UI), our output file, our network crawler agent, our database, security, and all business requirements. For our UI, we will test if the login menu functions correctly and that users will only be able to access the software if they successfully put in their credentials. For the output file we will need to test that the data is displayed properly (in correct columns, no missing data, etc) in the Excel spreadsheet. For the network crawler agent we will need to test if the crawler agent grabs all the target devices in a given network. The most important part of the software that must be tested in the database, since it has the majority of the important data we want to output. We must ensure that all data in the database is valid (correct data types) and are under the correct columns. We must verify that the database is properly storing all the devices and software into the database, ensuring that they are all accounted for. The naming conventions for the columns must make sense to the user. The primary and foreign keys must also be valid in order for the querying to work correctly. Just as important for our project are tests that address performance, security, and deployment are working for our project. The software must be secured and that only verified users have access to it. While we are unsure of the magnitude of data we are handling, the software should be able to perform at a reasonable level. We must also ensure that the software that we develop ends up working as expected and is natural to the users.

1. **Project Risks**
2. Data corruption
   1. Impact: System data may become corrupted, resulting in inaccurate or unreliable records; functionality and performance may also be negatively affected.
   2. Prevention: Ensure backup procedures are in place, perform regular database maintenance, and implement secure authentication upon tool execution.
   3. Monitoring: Frequently check the database and its entries for potential inaccuracies, data loss, and unexpected program crashes.
   4. Event Trigger: Any type of suspicious activity or database error messages must be mitigated.
   5. Mitigation: Recover data lost from backups and analyze the source of corruption. Next, document proactive steps to prevent future loss.
3. Declining database performance
   1. Impact: When filtering and storing large sets of inventory data, database performance may decline over time.
   2. Prevention: Update selected database to the most recent version of its releases, and install the latest version of the host operating system to avoid preventable technical issues.
   3. Monitoring: Evaluate performance statistics, record query response time, and monitor for potential overuse of capacity.
   4. Event Trigger: Observable slow performance when retrieving information or lengthy freezing during requests would require mitigation.
   5. Mitigation: Ensure queries are optimized when sent to the database and remove any entries that are not needed.
4. Data security breach
   1. Impact: Malicious actors may gain unauthorized access to sensitive data, causing data loss and extensive privacy breaches.
   2. Prevention: Verify authentication is secure and usable, with a focus on what the visible scope is for each user.
   3. Monitoring: Communicate any irregularities, suspicious login attempts, unexpected situations, or security alerts.
   4. Event Trigger: Be aware of any security alerts or unusual activity on local host machines or git repository files.
   5. Mitigation: Notify stakeholders of issues and contain systems that interact with the tool inventory system.
5. Software compatibility
   1. Impact: Performing scans of networks may not return all software devices due to compatibility issues, leading to inaccurate data or uncapturable data.
   2. Prevention: Install the most recent software for the database and ensure software between user interface, database, and authentication is compatible with various tests to verify proper connectivity.
   3. Monitoring: Especially in developmental stages, monitor inventory data for inconsistent entries or inaccurate records.
   4. Event Trigger: If inaccurate .csv reports are commonly generated, issues likely involve software compatibility.
   5. Mitigation: Test libraries and any potential crawler agent 3rd party tools early and often to verify accuracy and usability within project scope.
6. **Initial Product Release Plan**
   1. **Major Milestones**

**Table 3: Major Milestones**

| *Milestone* | *Description* | *Target Completion Date* |
| --- | --- | --- |
| *DatabaseDesign* | The database design should be complete, so we can start coding it for itself and the connecting parts. This needs to be available for review before the end of the first semester to provide time for revisions. | First week of December |
| *CrawlerAgentOutline* | Specifics of the crawler agent’s implementation (e.g. build from scratch or wrap open-source, how it will interact with the user and network, what information it will collect, how it will connect with the database, etc.) should be mapped out to facilitate success in the second semester | First week of December |
| *TestingPlan* | Decisions on how the various pieces of the software will be tested should be decided. This should include what parts need unit testing, how to go about user-acceptance testing, etc. | First week of December |
| *UserInterfaceLayout* | The user interface should be fully sketched out, allowing for approval before starting to implement it in the second semester. | First week of December |
| *DatabaseVersionZero* | Database will be built siloed. It will be hosted on a test AWS server and filled with test data emulating real-world conditions | First week of March |
| *CrawlerAgentVersionZero* | Crawler agent will be built siloed. It will be functional on a test network, traversing it, seeing all targets, and collecting appropriate data | First week of March |
| *TestingUnderway* | Testing will be complete to various extents, with multiple unit tests written for each component, and user-acceptance testing and integration tests planned out with extensive details | First week of March |
| *UserInterfaceVersionZero* | Each component of the user interface will be built siloed, without necessarily being connected together. It will not be connected to the database or other components of the project. Feedback from stakeholders will be requested and inform the final stage of its development | First week of March |
| *TestingComplete* | All testing will be in place, completed, and passing | Second week of April |

| *ComponentIntegration* | The database, crawler agent, and user interface will be connected, communicating with each other, and are functional in a development environment emulating production. | Second week of April |
| --- | --- | --- |
| *DocumentationComplete* | Documentation on each individual component, testing results, future directions, the software as a package, and end-user instruction will be written and saved in the repository | Third week of April |

* 1. **Initial Sprint Releases**

**Table 4: Sprint Release Plan**

| ***No*** | ***Sprint Date*** | ***Sprint Length*** | ***Sprint Goal*** | ***Backlog*** | ***What we will demo*** |
| --- | --- | --- | --- | --- | --- |
| ***1*** | ***10/24-***  ***11/6*** | **14** | **Evaluate how we handle the first sprint and to get initial designs done.** |  | **Current Software Versions** |
|  |  |  | **Complete ~1/3 total development on Database design, Crawler Agent Outline, Testing Plan, User Interface layout** |
| ***2*** | ***11/7-*** | **14** | **Complete ~3/4 total development on** |  | **Current Software Versions** |
|  | ***11/20*** |  | **Database design, Crawler Agent** |
|  |  |  | **Outline, Testing Plan, User Interface** |
|  |  |  | **layout** |

| ***3*** | ***11/27-***  ***12/4*** | **8** | **Database design, Crawler Agent Outline, Testing Plan, User Interface layout Due, Create second presentation** |  | **Current Software Versions** |
| --- | --- | --- | --- | --- | --- |
| ***4*** | ***12/5-*** | **14** | **Dead week + finals, presentation on** |  | **Current Software Versions** |
|  | ***12/18*** |  | **12/6, final project plan due 10/19** |
| ***5*** | ***12/19-*** | **14** | **Winter Break sprint 1** | **Depending on** | **Depending on** |
|  | ***1/1*** |  |  | **schedules, work** | **schedules, work will** |
|  |  |  |  | **will vary** | **vary** |
| ***6*** | ***1/2-*** | **14** | **Winter Break sprint 2** | **Depending on** | **Depending on** |
|  | ***1/15*** |  |  | **schedules, work** | **schedules, work will** |
|  |  |  |  | **will vary** | **vary** |
| ***7*** | ***1/16-***  ***1/29*** | **14** | **Complete ~1/4 total development on Database Version Zero,** |  | **Current Software Versions** |
|  |  |  | **Crawler Agent Version Zero,** |  |
|  |  |  | **Testing Underway,** |  |
|  |  |  | **User Interface Version Zero** |  |
| ***8*** | ***1/30-***  ***2/12*** | **14** | **Complete ~2/4 total development on Database Version Zero,** |  | **Current Software Versions** |
|  |  |  | **Crawler Agent Version Zero,** |  |
|  |  |  | **Testing Underway,** |  |
|  |  |  | **User Interface Version Zero** |  |

| ***9*** | ***2/13-***  ***2/26*** | **14** | **Complete ~3/4 total development on Database Version Zero,** |  | **Current Software Versions** |
| --- | --- | --- | --- | --- | --- |
|  |  |  | **Crawler Agent Version Zero,** |  |
|  |  |  | **Testing Underway,** |  |
|  |  |  | **User Interface Version Zero** |  |
| ***10*** | ***2/27-*** | **13** | **Database Version Zero,** |  | **Current Software** |
|  | ***3/10*** |  |  | **Versions** |
|  |  |  | **Crawler Agent Version Zero,** |  |
|  |  |  | **Testing Underway,** |  |
|  |  |  | **User Interface Version Zero Due** |  |
| ***11*** | ***3/18-*** | **8** | **Sprint after spring break to get back** | **Will vary on what** | **Documentation on** |
|  | ***3/25*** |  | **into the project. Start documenting** | **isn’t working.** | **what needs to be fixed.** |
|  |  |  | **version 0 bugs and where the** |  |  |
|  |  |  | **software is lacking.** |  |  |
| ***12*** | ***3/26-*** | **14** | **Refine and optimize version zero** | **Will vary on what** | **Optimized software** |
|  | ***4/8*** |  | **software** | **isn’t working.** | **components.** |
| ***13*** | ***4/9-*** | **14** | **Final sprint before delivery week.** | **Will vary on what** | **Refined and optimized** |
|  | ***4/22*** |  | **Finalize database, crawler agent,** | **isn’t working.** | **versions of project** |
|  |  |  | **testing, and UI** |  | **component** |
| ***14*** | ***4/23-*** | **11** | **Design Expo, Presentation #4 and** | **Presentation and** | **Deliver the project and deploy to clients** |
|  | ***5/3*** |  | **Senior Celebration Events all on 5/1,** | **FINAL tweaks to** |  |
|  |  |  | **final report due 5/3** | **software** |  |

1. **Maintenance Considerations**

Some aspects of our project that should be monitored throughout the product’s lifespan are the database and its performance, security issues, and compatibility with future software. We expect that the IT department of Medcurity, the most likely to be experienced with this line of work, will be able to, with the use of our thorough documentation, be able to understand and maintain all aspects of the product in the future.

While the database should be fairly simple and straightforward, there is a chance that issues may arise if invalid entries for the database appear or improperly formatted data. Tests will be developed to account for these issues. Should Medcurity seek to change fundamental aspects of the database, such as what users are allowed to enter or how the data should be formatted, the IT department should be able to clearly find where they should be able to do that. If the performance of the database dips, there should be systems in place for the IT department to evaluate performance statistics and pinpoint where the issue may be.

Security issues should be accounted for with a verification system for the product and should remain the same throughout the product’s lifespan. If Medcurity wants to create additional fields in

authentication to ensure user-sensitive data is secure, the documentation should allow the IT department to seemingly add more forms of security for the system.

Software updates could potentially render our system incompatible or outdated as time goes on.

In order to prevent this, all sections of the code should be clearly and thoroughly documented. The IT

department should be able to understand every section’s logic and code through documentation, allowing them to modify it should the time come.

1. **Project Management Considerations**

Our team plans to meet every Tuesday at 4:30 p.m. at PACCAR 203. Our faculty advisor will accompany the team in most meetings, and our primary DAB member and sponsors will attend whenever they must be present for discussion or updates. We have been using email as our primary form of communication, but everyone has joined a Discord server where brief communication can be made. We will plan to give our sponsor, DAB member, faculty advisor, and relevant stakeholders frequent updates of our progress, potentially weekly or bi-weekly, allowing them to be informed about details of the project. This is how we plan to break up the project, but it is not set in stone yet:

Brandon Huyck: Network Crawler Agents Colleen Lemak: Database

Jack Nealon: User Interface Artis Nateephaisan: Testing

We plan to use email to give progress reports, schedule meetings, ask questions, and receive feedback from liaisons and end users. If email is unnecessary for things such as quick questions, people may also send messages through Discord as an easy and informal way of communication.

**Team Member Bios**

Brandon Huyck

LinkedIn: linkedin.com/in/brandon-huyck

Skills

* Python, C/C++, Java, MySQL, R, Jupyter, Git, GitHub, Linux, Splunk, OOP, unit testing
* Problem solving, critical thinking, attention to detail, organization, big-picture thinking, brainstorming, teamwork, leadership, written and verbal communication, flexibility

Work Experience

* Starbucks, Software Engineering Intern in Tech Ops, Seattle, WA, May-August 2023
  + Workplace meetings, adaptability, solution design
  + Created a workflow using Microsoft Suite that increased automation and ease for team leadership, reducing fallibility of system; generated research report on technical transition for management; integrated into, and adapted, team culture.
* Mastercard, Technical Intern in BizOps, St. Louis, MO, June-August 2022
  + Big data, data analytics, data visualization
  + Created dashboards in Splunk that displayed the health of various applications using KPIs, determined the extent of impact of issues, and communicated to executives.
* Gonzaga University, Teaching Assistant, Spokane, WA, September 2022 – Present
  + Code review, explaining technical concepts
  + Data Structures and Algorithms, Introduction to Data Science, and Operating Systems

Education

* MyPL Interpreted Programming Language Project
  + Lexer, syntax parser, semantic checker, code generator, virtual machine, AST, visitor pattern
* GUI Yahtzee Application
  + OOP, Model-View-Controller, class and method design, SOLID, team project
* Computer Science Coursework
  + Databases, Computing Ethics, Theory of Computation, Programming Language Design, Machine Learning, Operating Systems, Computer Security, Introduction to Data Science
* Mathematics Coursework
  + Linear Algebra, Number Theory, Combinatorics, Statistics, Proofs, Calculus
* B.S. Computer Science, GPA 3.98, Gonzaga University, Spokane, WA, graduating May 2024

Interests

* Software Engineering, Testing, Databases, and Data Analysis
* Math tutor for K-12 students, foodie and cook, coffee nerd, card and board game player

Colleen Lemak

LinkedIn: linkedin.com/in/colleen-lemak/

Skills

* Proficiency in C++, Java, Python, R, MyPL, Kotlin, SQL, HTML.
* Consistent work-ethic and resilience
* Focused on a growth-mindset, a collaboratively oriented workplace, and a communicative atmosphere.
* Deeply values and strives for a diverse and inclusive work environment.

Publications

* Co-author of *IEEE Cyber Security and Resilience (CSR)* paper, 08/2023
  + *New Version, New Answer: Investigating Cybersecurity Static-Analysis Tool Findings*
  + [*https://ieeexplore.ieee.org/document/10224930*](https://ieeexplore.ieee.org/document/10224930)

Work Experience

* iD Tech Lead Instructor, 06/2023 to 08/2023

Lewis and Clark College – Portland, OR

* + Responsible for organizing and instructing computer science courses for 13-17 year olds and dispensing all medications necessary.
  + Courses include Artificial Intelligence & Machine Learning, Java Coding for Applications and Game Development, and Introduction to Python for Machine Learning.
  + A great opportunity to lead and engage with the next generation of programmers.
* NSF Cybersecurity Summer Researcher, 05/2022 to 08/2022

Montana State University Bozeman – Bozeman, MT

* + Produced analytic cybersecurity findings by programming scripts in Python and R.
  + Investigated static analysis tools, cwe\_checker and cve-binary-tool, tracking their output consistency over time.
  + Dedicated extensive time to creating and presenting a research poster, in addition to conducting a thorough literature review prior.
  + Navigated GitHub for code efficiency and proper file management.
* Computer Science Teaching Assistant, 08/2021 to 12/2021

Gonzaga University – Spokane, WA

* + Provided tutoring aid to freshman and sophomore students in their computer science courses.
  + Graded exams and projects in course CPSC 122 with Dr. Paul De Palma.

Education

* Senior at Gonzaga University pursuing a BS Computer Science major, an Applied Mathematics minor, and a Software Security concentration – 3.62 GPA.
* Looking to build upon cybersecurity skills and develop tools and solutions in accordance with reliable security standards.
* Received high school diploma – 4.2 GPA.

Interests

* Beyond the computer, I love camping, hammocking, and enjoying nature.
* I am also passionate about mental and physical health!

Artis Nateephaisan

LinkedIn: <https://www.linkedin.com/in/artisnateephaisan/>

Skills:

* C, C++, Python, Java, SQL, OCaml, Assembly, Markdown
* Calculus 1, Calculus 2, Calculus 3, Ordinary Differential Equations, Discrete Math
* Eager learner, excited to learn new topics
* Works well individually or as a team

Work Experience

* No relevant work experience

Education:

* Shorewood High School
  + 3.6 GPA unweighted
  + Student of the Semester 2019
* Courses pertaining to this project
  + CPSC 341 Internet of Things
  + CPSC 348 Computer Security
  + CPSC 491 Software Engineering
* Ring Doorbell Project
  + Contributed to a Ring Doorbell Project in a GU CS course, utilized Firebase for cloud storage, interacted with Raspberry Pi, created scripts in Python, and created an Android app UI to interact with the IoT.
* Spotify Data Gatherer
  + Created a program in Python that could gather user’s Spotify data and display a summary of stats including favorite artists, listening time, and predicting which genres the user may enjoy.

Interests:

* Other than computer science, I enjoy reading, hanging out with family and friends, and shooting hoops.

John Nealon

LinkedIn - <https://www.linkedin.com/in/jack-nealon/>

# Skills

A majority of my technical experience comes from my classes that I’ve taken while in school and I’ve acquired a number of skills to aid in my future career.

* **Programming** – Relevant experience with developing software written in C, C++, Java, Python, Netlogo, and x86 assembly languages.
* **Networking** – Experience working with networking technologies including Wi-Fi, ethernet, LAN/WAN and other connection systems.
* **Data Science** – Experience working with large scale data sets and produced analysis, optimization, and UI/UX software.
* **GitHub** – From my experiences throughout my college career, I have had a lot of experience with the systems of GitHub.

# Education

* High School: Silverton High School | Silverton, OR | 3.7 GPA | Honors Graduate
* University: Gonzaga University | Spokane, WA | Current Student
* B.A Computer Science and Computational Thinking - Biology Concentration
* Applicable courses pertaining to this project:
  + CS 341 – Internet of Things
  + CS 222 – Introduction to Data Science
  + CS 224 – Software Development
  + CS 223 – Data Structures
* Emphasis on technical application in biological industries

# Work Experience

* No relevant technical work experience.

# Interests

* In my personal life I am an avid chef who loves to study and create beautiful dishes with compelling flavors that create a memorable experience. I also read, write, watch movies and tv shows, and play the occasional video game. I am passionate about connecting technology with biology to develop optimized research strategies, secure information transfer, and UI/UX to improve communication.