**GONZAGA UNIVERSITY**

**School of Engineering and Applied Science**

**Center for Engineering Design and Entrepreneurship**

**PROJECT STATUS REPORT**

**December 7, 2023**

**Medcurity Network Inventory**

**Prepared by:**

**** 

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

Brandon Huyck Colleen Lemak

\_

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

Artis Nateephaisan John Nealon

**1 Project Overview**

In our modern age, companies are valuable targets for digital security breaches and malicious attacks. This is especially important to be aware of for companies that need to adhere to medical record compliance standards to protect sensitive patient health information from being disclosed without patient consent or knowledge. 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 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.

The major features of our project are outlined in the table below. We will deliver our product with code in a GitHub repository, README documentation outline what things are and how to execute it, and a bash script that executes the tool when run. Changes to Major Features are:

* Crawler-Database Feature is consolidated into Crawler and Database
  + Having Database, Crawler, and Crawler-Database was redundant, with tasks in Crawler-Database instead being handled by Crawler and Database separately as appropriate.
* Software Testing has been updated to better reflect tasks
  + A test network to test the crawler is required and is the main component of this feature. Testing of code of individual components is relegated to the component, itself. There is also a need for integration testing that remains unchanged under this feature.
* Scripting is added as a major feature
  + A bash script was decided on as the main way that the project will be launched and integrated.

# **Table 1: Major Features**

| *Feature* | *Description*  *Brief summary describing the feature and significance (as appropriate)* | *Team Member Responsible for Feature* |
| --- | --- | --- |
| *DatabaseSystem* | 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. 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. There will be an API for the crawler to easily plug into. | Brandon H. |
| *AdminPrivilege* | 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. |
| *SoftwareTesting* | 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. |
| *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. 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. | Colleen L. |

| *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. | Brandon H. and Jack N. |
| --- | --- | --- |
| *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. | 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 is important to collaborate with our project stakeholders in order to produce a high- quality, easily accessible and comprehensive inventory tool. | 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. |
| *Scripting* | A bash script that integrates and launches all components. | Brandon H. |

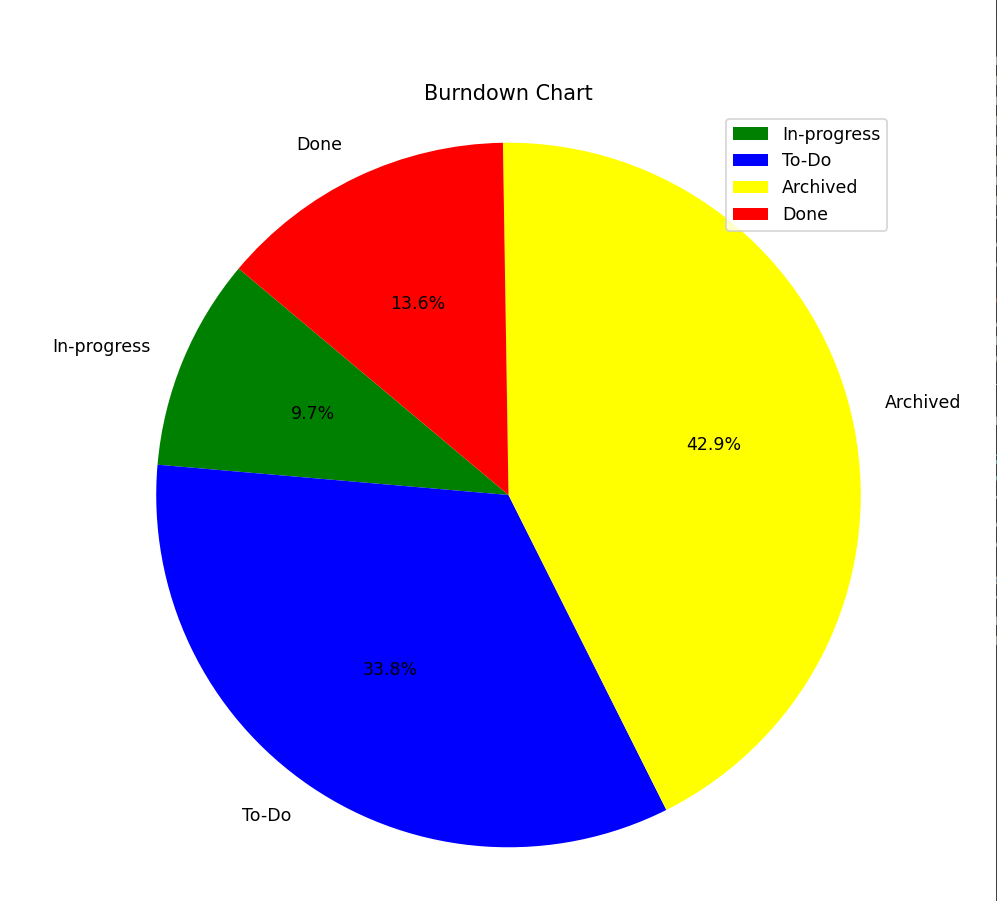
**2 Summary of Work Accomplished**

Provide a brief summary of the work you have accomplished so far in your project. Describe how the work you have done follows your previous release plan for the reporting period (highlighting any significant deviations from your release plan or additions to the plan). Either in this section or the next provide a burndown chart of your progress to date.

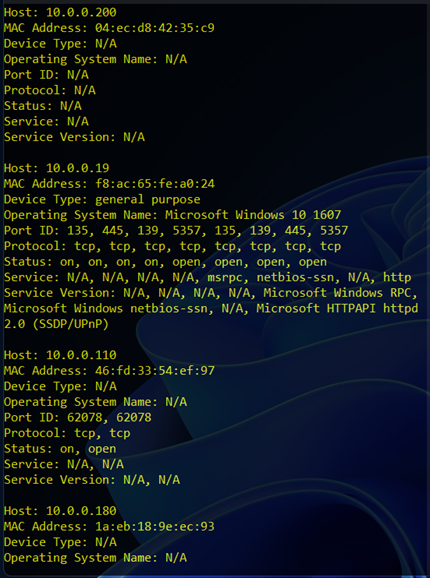
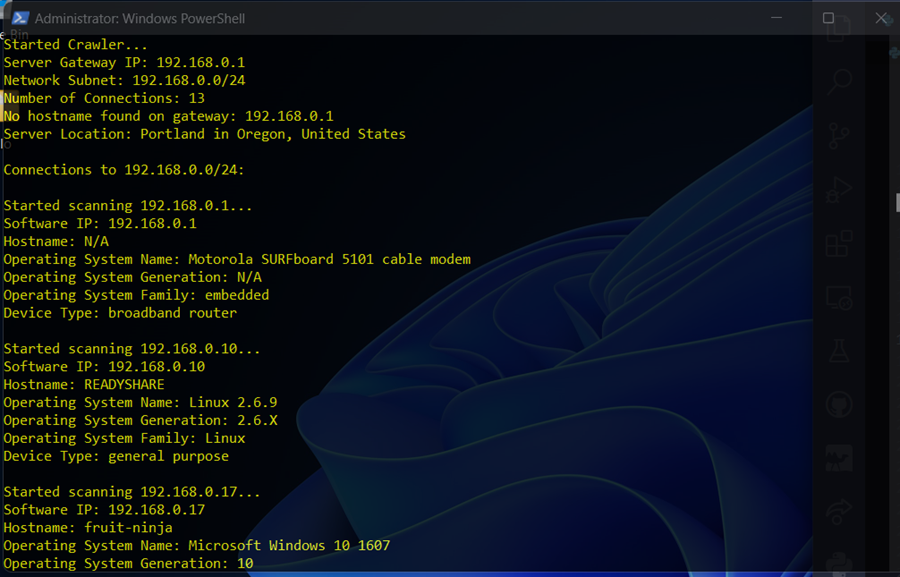
Our team is continuing to follow the release plan, and we are making progress on our major features implementation. A large part of our project purpose is to scan the network and log all targeted information.

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. Currently, “dummy-data” is in place of the crawled attributes from the Python scripts. However, we plan to get crawled information in the DB by pushing to a remote API connection as part of integration; this connection is in the works. 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 will be crucial to query the database in the upcoming sprints. There exists an initial basic page with a login screen. The UI is not quite able to be navigated with input, however, attributes will be pulled from the database to display when complete.

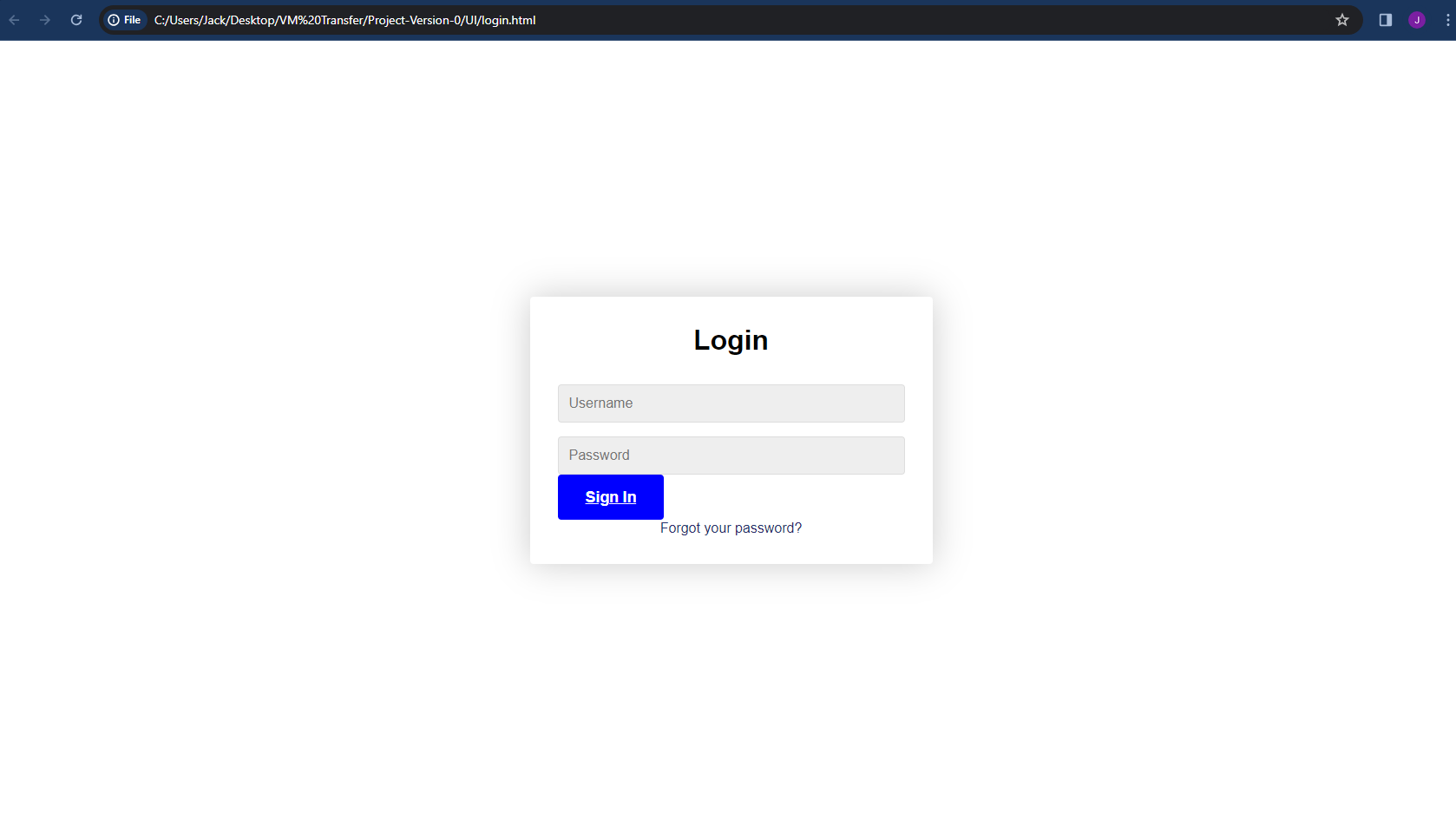
Our team documents all tasks on our Github project backlog and we use pull-requests and issues to address progress and plan for sprints. Documentation for administrators and users detailing how to use the tool will be provided as part of the second semester work. Every sprint (2-weeks) we conduct code-reviews 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. Although major features are not completely packaged up, our second semester will consist of scripting for integration, testing components, and documenting decisions. Below is our progress to-date based on our backlog tasks. This chart represents how about 57% of planned tasks are marked as done or archived from fall semester work. We have a clear view of next steps in our project, and are planning accordingly. 

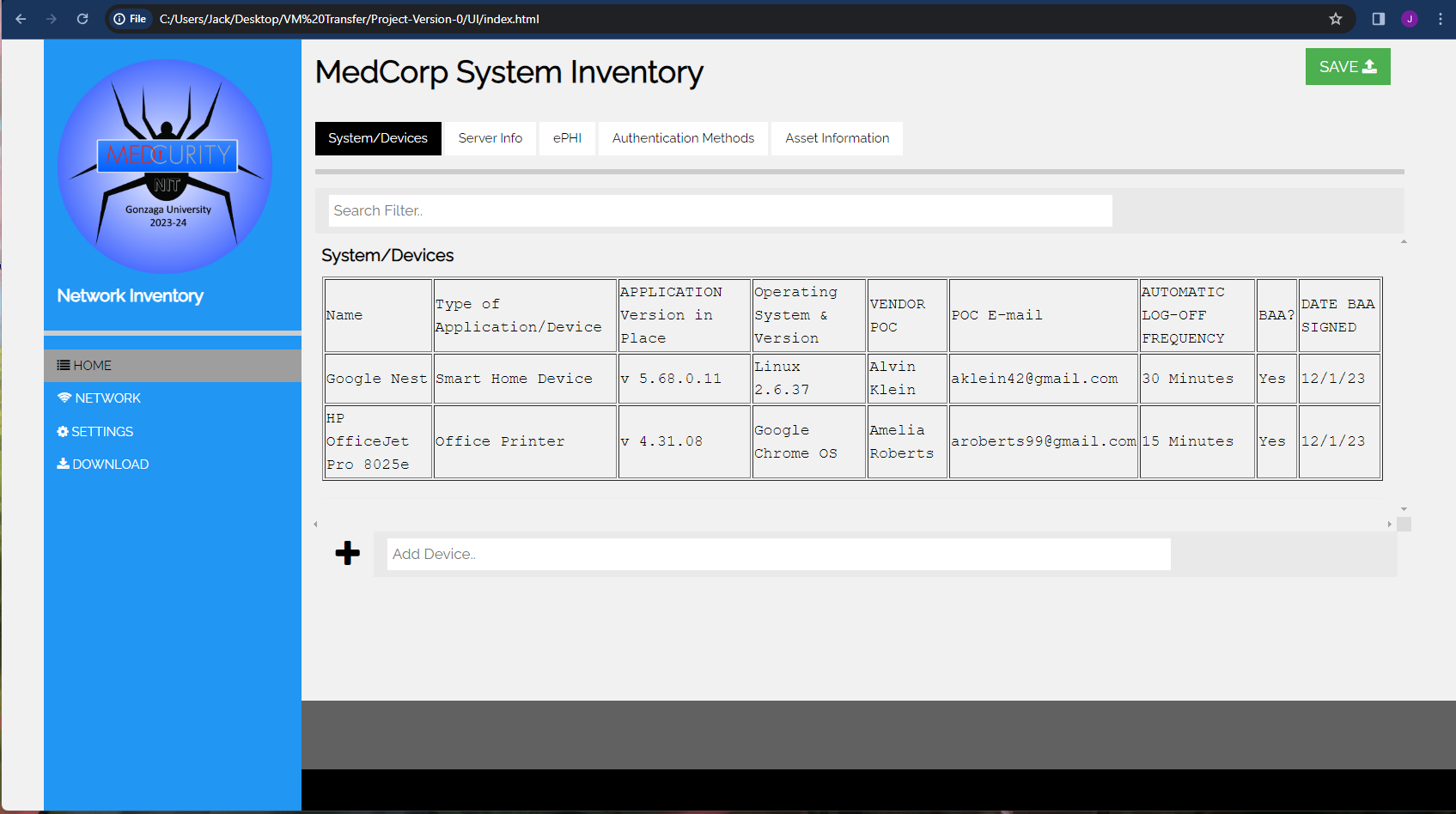
**3 Working Prototype -jack**

Crawler Agent – Using Nmap and ARP scanning we have expanded the crawler agent to include a lot more information than we previously had at the last presentation. From these scans we are able to get the following information in the following images: 

User Interface: The layout of the login and homepages have a good initial layout with structured containers written in HTML and aided by CSS. Currently, the only buttons with functionality include the left sidebar tabs as well as the homepage information tabs that detail the individual sections of the collected information. Additionally the login page’s blue sign-in button does take the user to the homepage. The following screenshots detail the login page and homepage running locally.

Login Page:



Homepage:

**4 Revised Product Backlog**

Our revised backlog contains more concrete items now that the major features and components have been finalized. It is ever growing, as we clear some items, decide on new ones, and granularize bigger ones once we start working on it.

**Table 2: Revised Product Backlog**

| *Requirement* | *Description* | *Major Feature* | *Priority* | *Estimate* |
| --- | --- | --- | --- | --- |
| *Test Database Python API* | Ensure that developed Python API for the database to be used by the crawler works | Database | High | Medium |
| *ProperUserPasswordStorage* | Securely store user passwords in database, probably as a result of salted hashing | AdminSecurity | High | Medium |
| *Database Manual Input for UI* | The database should be able to be inserted, updated, and deleted from the UI | Database | High | High |
| Database filters | The database should be able to be searched and filtered with SELECT-FROM-WHERE clauses from the UI | Database | Low | High |
| DatabaseCredentials | Database credentials need to be properly managed in deployment. Specifics regarding this need to be determined and implemented | Database | High | High |
| Crawler Information Collection | The crawler needs to be continued to be developed to collect all possible information | Crawler | High | High |
| Crawler Information Storage | The crawler needs to send the information it’s collecting to the database as it finds it using the database API in python | Crawler | High | Medium |
| Front End Development in PHP | Start using PHP in the front end development, for future connection to the database | UI | High | High |
| UI Configuration page | The UI needs a configuration page to access status information, add and modify users and passwords, and more | UI | Medium | Medium |
| UI Database connection | Add the database connection and MySQL Queries to the front end php/html code to display the information from the database | UI | High | High |
| UI Tabular Inventory Display | The UI needs to display the information from the database in a user friendly manner. The structures and design of this need to be further developed. | UI | High | High |
| User Authentication | When a user tries to log in, their account credentials need to be verified before granting them access to the tool and inventory | Admin | High | High |
| Manual input from UI | The UI needs to be designed to support manual insert, update, and delete with the database | UI | High | Medium |
| Test network set up | A test network needs to finished being set up for us to test the crawlers capabilities | Testing | High | High |
| Integration testing | All components of the project need to be integrated, and that integration needs to be tested to ensure proper functionality | Testing | Medium | Low |
| User Acceptance Testing | Once the components are integrated, the tool needs to be sent to the sponsor for UAT, to get feedback and eventual approval | Testing | Low | Low |
| Component Unit Testing | Each individual component of the project needs to be tested by its developer to ensure that it works as intended | Testing | High | Low |
| Integration and Launching script | Create the bash script to launch the tool from the terminal and integrate all the components | Scripting | high | low |
| Excel Creation script | Create the python script to create the excel file when prompted by the UI. Use python modules for excel, and the Database API in python. | Scripting | Medium | Medium |
| Document Components | Add documentation for each component upon stage completion | Documentation | Medium | Medium |
| Document Instructions | Add documentation describing how to use the tool | Documentation | High | Medium |

**5 Revised Product Release Plan**

**Table 3: Major Milestones**

| *Milestone* | *Description* | *Target Completion Date* | *Progress* |
| --- | --- | --- | --- |
| *Database Design* | 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 | Complete |
| *Crawler Agent Outline* | 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 | Complete |
| *Testing Plan* | 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 | In Progress |
| *User Interface Layout* | The user interface should be fully sketched out, allowing for approval before starting to implement it in the second semester. | First week of December | Complete |
| *Database Version 0* | 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 | On Time |
| *Crawler Agent Version 0* | 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 | On Time |
| *Testing Underway* | 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 | On Time |
| *User Interface Version 0* | 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 | On Time |
| *Testing Complete* | All testing will be in place, completed, and passing | Second week of April | On Time |

| *Component*  *Integration* | 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 | On Time |
| --- | --- | --- | --- |
| *Documentation*  *Complete* | 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 | On Time |

**Table 4: Sprint Release Plan**

| ***No*** | ***Sprint Date*** | ***Sprint Length*** | ***Sprint Goal*** | ***Backlog*** | ***What we will demo*** | **Progress** |
| --- | --- | --- | --- | --- | --- | --- |
| ***1*** | ***10/24-***  ***11/6*** | **14** | **Evaluate how we handle the first sprint and to get initial designs done.** |  | **Current**  **Software**  **Versions** | **Complete** |
|  |  |  | **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** | **Complete** |
|  | ***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** | **Complete** |
| --- | --- | --- | --- | --- | --- | --- |
| ***4*** | ***12/5-*** | **14** | **Dead week + finals, presentation on** |  | **Current**  **Software**  **Versions** | **In Progress** |
|  | ***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** | **On Time** |
|  |  |  |  | **will vary** | **vary** |  |
| ***6*** | ***1/2-*** | **14** | **Winter Break sprint 2** | **Depending on** | **Depending on** |  |
|  | ***1/15*** |  |  | **schedules, work** | **schedules, work will** | **On Time** |
|  |  |  |  | **will vary** | **vary** |  |
| ***7*** | ***1/16-***  ***1/29*** | **14** | **Complete ~1/4 total development on Database Version Zero,** |  | **Current Software Versions** | **On Time** |
|  |  |  | **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** | **On Time** |
|  |  |  | **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** | **On Time** |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **Crawler Agent Version Zero,** |  |  |
|  |  |  | **Testing Underway,** |  |  |
|  |  |  | **User Interface Version Zero** |  |  |
| ***10*** | ***2/27-*** | **13** | **Database Version Zero,** |  | **Current Software** | **On Time** |
|  | ***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.** | **On Time** |
|  |  |  | **version 0 bugs and where the** |  |  |  |
|  |  |  | **software is lacking.** |  |  |  |
| ***12*** | ***3/26-*** | **14** | **Refine and optimize version zero** | **Will vary on what** | **Optimized software** | **On Time** |
|  | ***4/8*** |  | **software** | **isn’t working.** | **components.** |  |
| ***13*** | ***4/9-*** | **14** | **Final sprint before delivery week.** | **Will vary on what** | **Refined and optimized** | **On Time** |
|  | ***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** | **On Time** |
|  | ***5/3*** |  | **Senior Celebration Events all on 5/1,** | **FINAL tweaks to** |  |  |
|  |  |  | **final report due 5/3** | **software** |  |  |

Project Backlog Items and Version Control can be tracked at the following GitHub Project Link: [Link](https://github.com/orgs/SD-2023-CS10/projects/2/views/1?filterQuery=)

**6 Project Risks**

Provide an ***update*** on the risks identified in the previous report. For each risk, state whether it remains a risk for your project (and why), what actions you have taken to prevent and monitor the risk, and whether you have had to mitigate the risk (e.g., change to a “plan B”).

*Also provide any new risks that are associated with your project. For each new risk, you must clearly and concisely describe: (1) why it is a risk to the project (e.g., what will the potential impact to the project be); (2) what actions you will take to prevent the risk from happening; (3) how you will monitor the risk; (4) what events/situations will trigger the need to mitigate the risk (i.e., when will you know to switch to “plan B”); and (5) what you will do if the risk does becomes a reality (i.e., what is your “plan B”, “plan C”, etc.). It is not enough to just list your risks; you must also have a plan to prevent, monitor, and mitigate each risk.*

1. Data corruption (Remains Risk)
   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.
2. Declining database performance (No Longer Risk)
   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.
3. Data security breach (Remains Risk)
   1. Impact: Malicious actors may gain unauthorized access to sensitive data, causing data loss and extensive privacy breaches. This risk also extends to unintentional remote credential-leaks.
   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.
4. Software compatibility (Remains Risk)
   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.
5. Protocol compatibility (New Risk)
   1. Impact: Scan does not return valid information if the communication to the device cannot be resolved.
   2. Prevention: Consider various approaches to getting information out of systems: continue reverse DNS and ARP scans if only one approach does not retrieve enough information.
   3. Monitoring: Print errors out and ensure the program can run to completion.
   4. Event Trigger: Programs crashing or systems freezing on a network scan could indicate issues with connectivity and information-retrieval. This could also be triggered by firewall or security settings configuration.
   5. Mitigation: Use try-except blocks in Python to allow one host to not have, for example, an operating system, but the next is present and reachable.

**7 Delivery and Maintenance Considerations**

Provide a brief update describing new or revised maintenance issues regarding the system you develop, focusing on any changes since the previous report. This needs to include updates on how you’re going to package and deploy your project for your sponsor.

While our delivery and maintenance plans remain mostly the same, there are some minor changes we have made. We still intend to hand Medcurity everything we worked on through giving access to our GitHub repository that we have been working on, which should be a smooth process. Along with our source code, we will also have files covering documentation in the GitHub repository so that Medcurity understands everything that we have worked on, from how each function works, to any potential concerns regarding performance or security.

We have maintained our stance that security and future software compatibility are both important aspects that we must consider, however, after talking with our DAB member, we learned that performance is not as important as we initially expected. In terms of database querying, we would have to handle much more data (industry level) than what is likely practical, so database efficiency will likely never be a factor. As long as our crawler and scripts are reasonably efficient, performance should relatively be a non-factor. On the other hand, we realized how important security matters. In order to handle this, our UI must have a way to handle login authentication. We plan on using AWS, which already has their own form of encryption, to store sensitive information from Medcurity’s clients. We are in the process of implementing hashing for user login, to ensure that login information is not compromised.

Regardless if our tool is used years down the line, we must ensure that we consider future software compatibility. We figure that the best way to address this is to provide thorough documentation throughout all of our code, providing insight and details on each component of our tool. That way should Medcurity want to look back at our tool, they understand how each part works not just on a basic understanding, but on a level where they can go through and make changes as necessary.

**8 Project Management Considerations**

Briefly describe how you have worked as a team to complete your working prototype, focusing on any changes you have made since the previous report. Discuss any issues and things you plan to change moving forward as a team.

We got together as a team and decided on what core functional components needed to be completed in order to produce a successful tool. We concluded that the following needed to be incorporated: a crawler, a database, a UI, scripting, testing, documentation. We then assigned 3 tasks to each team member, where two members are on each component, where they will then work together to complete these core tasks. So far it has been working great: work has been split up evenly and we have been progressing as expected.

Due to the two weekly meetings, one with our advisor and one with the development team, there has been no confusion on what has been assigned to who and everyone has maintained a steady pace of work. Scrum leaders, who are alternated every two weeks, lead the meetings and go over what has been accomplished throughout the week and what needs to be assigned. So far this process has been consistent and has yielded good results. We have also continued to use Discord and email to communicate with our sponsor and DAB member, asking questions whenever the team deemed necessary. In terms of communicating within the development team, we usually use text messaging to ask and answer questions and plan smaller meetings outside of our two standard meetings.

One thing that we mentioned in our previous project plan document was providing progress updates to our sponsor. We have not been keeping our sponsor in loop as much as we would like, so next semester we plan on showing them the progress we have made (even if the updates are in two or three week increments) to ensure that our project falls in line with what they envisioned.