**GONZAGA UNIVERSITY**

**School of Engineering and Applied Science**

**Center for Engineering Design and Entrepreneurship**

**Medcurity Network Inventory**

**Project Requirements**

**Plan Section 02**

**Release:**

**Draft v0.1**

**PROJECT PLAN DRAFT STAGE DOCUMENT**

**September 17, 2023**

**Medcurity Network Inventory Team**



**Brandon Huyck**

**Colleen Lemak**

**Artis Nateephaisan**

**Jack Nealon**

**2 Project Requirements**

**2.1 Major Features**

Provide a description of the major features that must be implemented for a viable and useful product. Major features include broad feature areas, constraints that must be met, and other major items that must be completed for the project to be considered successful. The major features should be identified in consultation with your project sponsor and target user communities. The major features described here should also be listed in the major features checklist, which must be approved by your sponsor and faculty advisor. Each major feature should also have rationale for inclusion. This subsection should also discuss major features that were considered nice to have but were not included as targets for your project and a discussion of why they were not included. Describe the process your team used to determine the major features, which should be determined through discussions and feedback from your target users as well as your project sponsors. Provide a summary of the major features in a table as follows (again, the table should match the items given in the major features checklist).

**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. |

**2.2 Initial Product Backlog**

Provide a description of the essential project requirements (features, characteristics, constraints of the system) that must be developed for project success. Your requirements must have unique names/titles; have priorities and estimates; be clear, concise, free of jargon, and consistently worded; be specific (each requirement should capture one aspect of the system); and be measurable (i.e., clearly state what “done/completed” looks like). Include a brief description of your priority and estimate scheme.

Note that high priority requirements (typically requirements you will start on right away or soon after) should be well defined and more precise but lower priority items can be fuzzier and less precise at this point. You can state your requirements as user stories, however, you should also include requirements for non-functional aspects of the system as well (as needed). Note that the requirements in this subsection should elaborate upon the major features listed in the previous subsection. Summarize your requirements in a table as follows or use your Kanban board (Trello or GitHub) to gather a series of screenshots that summarize your backlog items.

**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 not. This will be stored as a YES/NO | DatabaseSystem | High | Medium |
| *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 SQL injection and 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 devices we want to add to the database | CrawlerAgentTraversal | High | Medium |
| *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 |
| *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 |
| *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 |

**2.3 Additional Features**

Provide a description of non-essential, but nice to have product features, characteristics, and constraints (i.e., “stretch goals”). As with your project requirements (major needs), be sure to organize, name, prioritize, and clearly articulate each additional project feature, characteristic, and constraint. Each additional feature should be specific and measurable.

**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 |  |
| *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 |